



Robotik Zusammenfassung

Prof. Schiedermeier

SS 2019

Robin Atherton

Contents

1	Geschichte	1
1.1	Industrieroboter	1
1.2	Serviceroboter	1
1.2.1	Definition	1
1.2.2	Klassen	2
2	Software-Architekturen für mobile Robotersysteme	3
2.1	Probleme und Anforderungen	3
2.1.1	Definition mobile Roboter	3
2.1.2	Umgebung mobiler Roboter	3
2.1.3	Roboterkontroll-Architekturen	4
2.1.4	Anforderungen an das Kontrollsystem eines autonomen Roboters	4
2.2	Mögliche Modelle	5
2.2.1	Klassisches Modell - der funktionale Ansatz	5
2.2.2	Verhaltensbasiertes Modell	6
2.2.3	Hybrider Ansatz	7
2.2.4	Probabilistische Robotik	8
2.2.5	Subsumption-Architektur in Bezug auf die Anforderungen des Robotersteuerungssystems	8
2.3	ROS - Robot Operating System	9
2.3.1	Entwicklung	9
2.3.2	Design Prinzipien	10
2.3.3	Publish - Subscribe	11
2.3.4	Parameter Server und Konfigurationsdateien	12
3	Lokalisation autonomer mobiler Robotersysteme	13
3.1	Abgrenzung: Lokalisation - Mapping - SLAM - Navigation	13
3.2	Varianten der Selbstlokalisierung	13

Contents

3.3	Relative Lokalisierung versus Absolute Lokalisierung	14
3.4	Transformation von Koordinatensystemen lokale \leftrightarrow globale	15
List of Figures		17
Listings		18

1 Geschichte

1.1 Industrieroboter

Nach Definition der VDI-Richtlinie 2860 sind Industrieroboter universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen bzw. Winkel frei programmierbar und sensorgeführt sind.

- Zeichnen sich durch **Schnelligkeit, Genauigkeit, Robustheit** und eine hohe **Traglast** aus.
- Einsatzgebiete: Schweißen, Kleben, Schneide, Lackieren

Zunehmend **kollaborative** Roboter, Cobots:

- Industrieroboter, die mit Menschen gemeinsam arbeiten
- Nicht mehr durch Schutzeinrichtungen im Produktionsprozess von Menschen getrennt
- Nimmt Menschen wahr, verursacht keine Verletzungen

1.2 Serviceroboter

1.2.1 Definition

- Ein **Serviceroboter** ist eine **frei programmierbare Bewegungseinrichtung**, die **teil- oder vollautomatisch** Dienstleistungen verrichtet.
- **Dienstleistungen** sind dabei Tätigkeiten, die nicht der direkten industriellen ERzeugung von Sachgütern, sondern der Verrichtung von **Leistungen für Menschen und Einrichtungen** dienen.
- Einteilung in zwei Klassen

1.2.2 Klassen

- Roboter, die für professionellen Einsatzbereich: **Rettung, Landwirtschaft, Medizin**
- Roboter für den Privaten gebrauch: **Staubsauger, Rasenmäher, Pfleger**

2 Software-Architekturen für mobile Robotersysteme

2.1 Probleme und Anforderungen

2.1.1 Definition mobile Roboter

'Unter einem Roboter verstehen wir eine frei programmierbare Maschine, die auf Basis von Umgebungssensordaten in geschlossener Regelung in Umgebungen agiert, die zur Zeit der Programmierung nicht genau bekannt und/oder dynamisch und oder nicht vollständig erfassbar sind.' ⇒ **Joachim Herzberg, Mobile Roboter**

2.1.2 Umgebung mobiler Roboter

Bei **mobilen Robotern** ist die Umgebung im Detail **nicht bekannt und generell nicht kontrollierbar**

- Alle Aktionen sind von der aktuellen Umgebung abhängig
- Details sind erst zum Zeitpunkt der Ausführung der Aktionen bekannt
- Mobile Roboter müssen in einer geschlossenen Regelung
 - die Umgebung mit Sensoren erfassen
 - die Daten auswerten
 - Aktionen daraus planen
 - Aktionen mittels Koordination der Aktuatoren umsetzen

2.1.3 Roboterkontroll-Architekturen

Herausforderungen

- Robotersystem besteht aus den Gebieten **Wahrnehmung**, **Planung** und **Handlung**
- Herausforderungen an eine Roboterkontroll-Architektur, sie muss:
 - Sensorwerte erfassen und auswerten
 - Pfade planen
 - Hindernisse vermeiden
 - Komplexe Algorithmen in langen Zeitzyklen ausführen

Probleme bei der Software-Erstellung zur Roboterkontrolle

- Roboter sind eingebettete Systeme, die in geschlossener Regelung laufen und die Sensorströme in **Echtzeit verarbeiten** müssen
- Unterschiedliche Aufgaben → Unterschiedliche Zeitzyklen
- Unterschiedlicher Zeitskalen → kein standardisierter Kontroll- oder Datenfluss den die Architektur abbilden könnten
- Für etliche algorithmische Teilprobleme sind **keine effizienten Verfahren** bekannt
- **Prozessorkapazität ist begrenzt**

2.1.4 Anforderungen an das Kontrollsystem eines autonomen Roboters

Robustheit

- Die Umgebung des Systems kann sich ständig ändern
- Auf eine Umgebungsänderung sollte der Roboter sinnvoll reagieren und nicht verwirrt stehen bleiben.
- Verwendete Modelle der Umgebung sind ungenau.

2.2 Mögliche Modelle

Unterschiedliche Ziele

- Der Roboter verfolgt zu einem Zeitpunkt eventuell Ziele, die im Konflikt zueinander stehen.
- **Beispiel:** der Roboter soll ein bestimmtes Ziel ansteuern, dabei aber Hindernissen ausweichen.

Sensorwerte von mehreren Sensoren

- Sensordaten können verrauscht sein
- Sensoren können fehlerhafte oder inkonsistente Messwerte liefern, weil der Sensor z.B. außerhalb seines Bereichs misst für den er zuständig ist und dies nicht überprüfen kann.

Erweiterbarkeit

- Wenn der Roboter neue Sensoren erhält, sollte dies leicht in das Programm integriert werden können.

2.2 Mögliche Modelle

2.2.1 Klassisches Modell - der funktionale Ansatz

Das **klassische Modell** wird auch als hierarchisches Model oder funktionales Model bezeichnet. Ist ein Top-Down Ansatz, besteht aus drei Abstraktionsebenen

- Die unterste Ebene: **Pilot**
- Mittlere Eben: **Navigator**
- Oberste Ebene: **Planer**

Sense-Think-Act-Cycle oder **SMPA** (Sense - Model - Plan - Act).

- Sensordaten, die vom Fahrzeug geliefert werden, werden in den zwei unteren Ebenen vorverarbeitet.
- Konstruktion oder Aktualisierung eines Weltmodells
- **Planer** ist die Basis aller Entscheidungen basieren auf dem zugrundeliegenden Weltmodell

2.2 Mögliche Modelle

- Tatsächliche Fahrbefehle werden durch unterste Ebene ausgeführt

Zyklus wird ständig wiederholt \Rightarrow wenn alle Ebenen richtig funktionieren resultiert daraus ein intelligentes Verhalten und die Erfüllung der Aufgabe.

Nachteile

- **Sequentieller Ansatz, lange Kontrollzykluszeit**
- Gesamtsystem anfällig, \Rightarrow fällt ein Modul scheitert das Gesamtsystem
- Die Repräsentation der Umgebung muss alle notwendigen Informationen enthalten, damit ein Plan entwickelt werden kann. Planer hat nur Zugriff auf das Weltmodell \Rightarrow während Planer aktionen ausarbeitet, könnte sich die Umwelt schon wieder geändert haben.

2.2.2 Verhaltensbasiertes Modell

Grundlegender Gedanke Intelligentes Verhalten wird nicht durch komplexe, monolithische Kontrollstrukturen erzeugt, sondern durch das Zusammenführen der richtigen einfachen Verhalten und deren Interaktion.

Definition

- Engere Verbindung zwischen **Wahrnehmung** und **Aktion**
- Jede **Roboterfunktionalität** wird in einem **Behavior** gekapselt
- Alle **Behaviors** werden **parallel ausgeführt**
- Jedes Behavior Modul operiert unabhängig von den anderen
- Alle Behaviors können auf alle Fahrzeugsensoren zugreifen und gewissermaßen die Aktuatoren ansteuern.

Beispiel Subsumption Architektur nach Brooks

2.2 Mögliche Modelle

2.2.3 Hybrider Ansatz

- Nutzt die Vorteile der **Subsumption Architektur** und der **SMPA-Architektur**
- Der verhaltensbasierte Anteil ist nicht geeignet, auf längere Sicht zielgerichtet Aktionen zu koordinieren \Rightarrow SMPA-Anteil

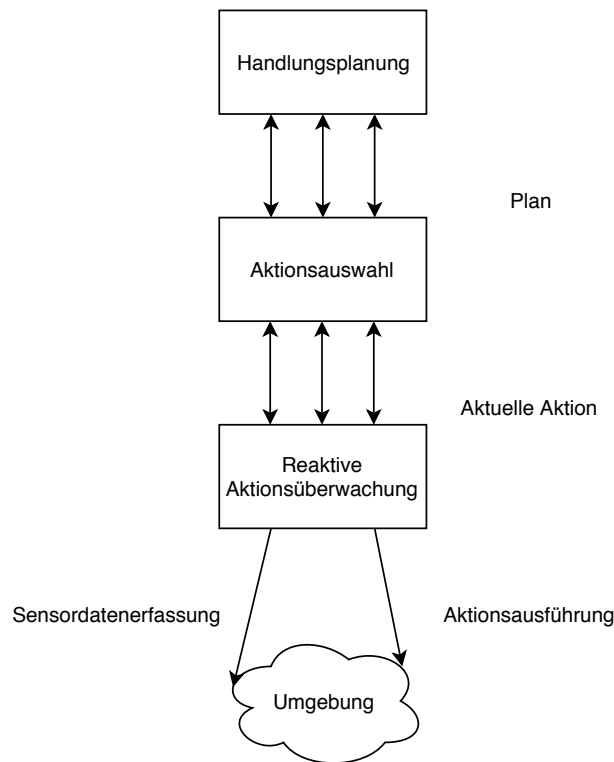


Figure 2.1: Schema der Hybridmodell Schichten

- Die **Handlungsplanung** arbeitet auf hoher, strategischer Stufe in langen Zeitzyklen
- Die **reaktive Aktionsüberwachung** enthält die Verhaltensbausteine auf operativer Ebene, die in schnellen Zeitzyklen die physische Roboteraktion anstoßen und überwachen
- die **mittlere Kontrollebene** hat die taktische Aufgabe, die jeweils **nächste Aktion aus dem Plan auszusuchen** zu instanzieren und auf die Ebene der Verhaltensbausteine zu zerlegen. Des Weiteren muss die Rückmeldung von der Aktionsüberwachung interpretieren und entscheiden, ob eine Aktion erfolgreich abgeschlossen ist. \Rightarrow entscheiden, ob die Handlungsplanung einen anderen Plan erstellen muss

2.2 Mögliche Modelle

Kritik

- Mittlere Komponente benötigt den größten konzeptuellen und programmiertechnischen Aufwand
- Das mittlere Teilproblem ist deutlich komplexer als die beiden anderen

2.2.4 Probabilistische Robotik

- Probabilistische Robotik berücksichtigt die **Unsicherheit der Wahrnehmung und der Aktionen**
- **Schlüsselidee** Information in Form von Wahrscheinlichkeitsdichten repräsentieren
- Eine Lokalisierung der Roboter wird unter Verwendung von Wahrscheinlichkeitstheorie oder einer Wahrscheinlichkeitsverteilung eine Aussage über die Umgebung treffen
- **Probabilistische Wahrnehmung:** wenn man Sensorwerte schätzen kann, dann kann man mit Wahrscheinlichkeitstheorie/Wahrscheinlichkeitsverteilung eine Aussage über die Umgebung treffen
- **Probabilistisches Handeln:** aufgrund der Unsicherheit über die Umgebung ist auch das Handeln mit Unsicherheit behaftet. Mit probabilistischen Ansätzen besteht die Möglichkeit Entscheidungen trotz Unsicherheit zu treffen

Vorteil probabilistische Verfahren können auch mit weniger präzisen Umgebungsmodellen angewandt werden.

Nachteil weniger effizient wegen komplexer Berechnungen, Approximation erforderlich

2.2.5 Subsumption-Architektur in Bezug auf die Anforderungen des Robotersteuerungssystems

Robustheit

- Wenn einige Steuerungsmodule ausfallen, arbeiten bei der Subsumption-Architektur in die restlichen Schichten einwandfrei ⇒ **eingeschränktes, aber sinnvolles Verhalten möglich**

2.3 ROS - Robot Operating System

Unterschiedliche Ziele

- Mehrere Teilsituation können verschiedene Verhaltenselemente sinnvoll machen, die sich widersprechen können.
- Die Wichtigkeit einer Handlung hängt vom Kontext ab, d.h. höhere Ziele können niederere Ziele ersetzen.
- Alle zu einem Zeitpunkt möglichen Verhaltenselemente werden parallel bearbeitet.
- Das **resultierende Verhalten wird in Abhängigkeit von Umwelteinflüssen dynamisch** bestimmt
- Das Gesamtergebnis hängt nicht von einer übergeordneten Instanz ab

Sensorwerte von mehreren Sensoren

- Der Roboter muss auch bei inkonsistenten Informationen eine Entscheidung fällen
- Die Subsumption-Architektur sieht keine zentrale Verarbeitung und Speicherung der Umweltdaten
- Jedes Modul reagiert nur auf die Daten einzelner Sensoren, es muss **kein konsistentes Abbild der Umwelt erschaffen werden**

Erweiterbarkeit Das bestehende Verhalten kann jederzeit durch Hinzufügen weiterer Schichten um komplexere Funktionen erweitert werden

2.3 ROS - Robot Operating System

2.3.1 Entwicklung

- **Das Architekturschema für Roboterkontrollsoftware** gibt es nicht \Rightarrow deshalb heute auch Unterstützung der Roboter-Softwareentwicklung durch Middleware wie ROS
- **Zweck:** soll die Entwicklung von Software für Roboter vereinfachen und wiederkehrende Aufgaben standardisieren
- Standard für Roboterkontrollsoftware

2.3.2 Design Prinzipien

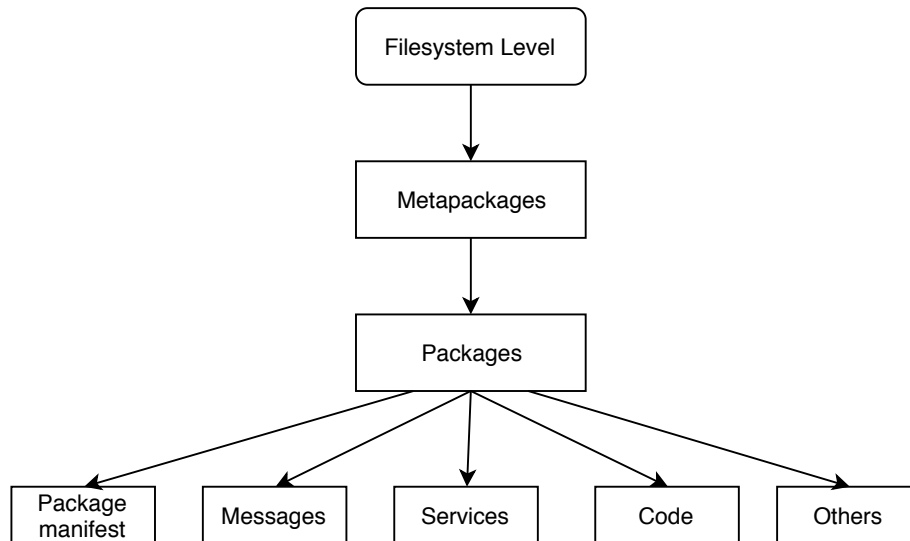


Figure 2.2: Filesystem, das ROS zugrunde liegt

- Ein Package beinhaltet die ROS Prozesse, welche auch Nodes genannt werden
- Komplexe Prozesse werden durch Netzwerke von Nodes bewerkstelligt
- Roboterkontrollprogramm besteht aus vielen Prozessen, die potentiell über mehrere Rechner verteilt sein können
- Wichtigster Knoten ist der Master \Rightarrow Abwicklung der internen Kommunikation
- Andere Knoten können nur starten, wenn ein Master existiert
- Nodes müssen sich beim Master anmelden
- Funktionalität (Kommandos, Ausführung v. Algorithmen) wird in eigenen Nodes realisiert
- Nodes sind in verschiedene Prozesse getrennt \Rightarrow fehlerhafte Knoten hat i.d. Regel wenig Auswirkungen auf die anderen
- Knoten werden über Publish-Subscribe verknüpft
- **Asynchrone Nachrichten** werden durch Topics ausgetauscht
- **Synchrone Nachrichten** werden durch Services ausgetauscht

2.3.3 Publish - Subscribe

Nodes sind Software-Module, die die Verarbeitung durchführen. Sie kommunizieren über Topics miteinander und tauschen dabei Nachrichten.

Kommunikation Die Kommunikation basiert auf einem **Publish Subscribe Pattern**

- Wenn Daten weitergegeben werden sollen, wird ein Publisher erzeugt
- Publisher registriert sich beim Master und gibt Topics an
- Daten können in anderen Knoten abgerufen werden – dazu wird ein (oder mehrere) Subscriber angelegt
- Subscriber fragt beim Master bezüglich gewünschten Topics an
- Daten werden über TCP/IP Sockets übertragen

Topics

- Themen, zu denen die Nodes Messages versenden
- Topics sind einfach Strings
- Verschiedene Nodes können zu einem bestimmten Topic Nachrichten versenden
- Ein Node kann sich prinzipiell zu mehreren Topics einschreiben und mehrere Topics publizieren

Services

- Nachteil von Publish Subscribe wird durch Services geschlossen
- Sind eine weitere Art, wie Nodes kommunizieren können
- Synchroner Nachrichtenaustausch mithilfe von Requests, welche von anderen Nodes mit einer Response beantwortet werden
- Ein Knoten registriert eine Aktion (Service), namentlich beim Master
- Ein Service Caller kann die Ausführung eines Services anstoßen, sobald dieser verfügbar ist
- geeignet für RMI oder einmalige Anfragen

2.3 ROS - Robot Operating System

Messages

- werden von Nodes bei der Kommunikation
- Messages sind streng typisierte, möglicherweise verschachtelte Datenstrukturen, die aus den primitiven Typen int, float, bool bestehen
- Eine Message kann andere Messages oder Felder von Messages enthalten

2.3.4 Parameter Server und Konfigurationsdateien

- Der ParameterServer auf dem Master Knoten enthält eine Art Wörterbuch für Werte.
- Alle Ressourcen wie Knoten, Nachrichten oder Parameter existieren in einer hierarchischen Namensstruktur.
- Speichert z.B. Konfigurationsdateien

3 Lokalisation autonomer mobiler Robotersysteme

3.1 Abgrenzung: Lokalisation - Mapping - SLAM - Navigation

- **Lokalisation** Ermitteln der aktuellen Position des Roboters.
- **Kartenerstellung, Mapping oder Umgebungsmodellierung** hilft bei Entscheidung
Kartenerstellung bedeutet die Auswertung der vom Roboter mittels Sensoren erfassten Daten der Umgebung mit dem Ziel, ein Umgebungsmodell zu erzeugen oder zu vervollständigen.
- **Großes Problem** Mapping
- **Pfadplanung oder Navigation** beantwortet die Frage **Wie gelange ich dorthin?**
Bewegungsplanung oder Pfadplanung bedeutet die Berechnung der Fahrroute und der daraus abgeleiteten Bahn vom aktuellen Punkt zum Zielpunkt
- Man unterscheidet zwischen Navigation in unbekannter und in bekannter Umgebung.
- Selbstlokalisierung und Kartenerstellung bedingen sich gegenseitig.

3.2 Varianten der Selbstlokalisierung

Lokale Selbstlokalisierung (position tracking)

- Die Startposition des Roboters ist ungefähr bekannt.
- Es handelt sich um **relative** Selbstlokalisierung

3.3 Relative Lokalisierung versus Absolute Lokalisierung

- Sobald sich der Roboter bewegt, muss aufgrund neuer Sensordaten die Position neu berechnet werden.
- Bezugspunkt ist der Startpunkt.
- **Methoden** Odometrie und Trägheitsnavigation

Globale Selbstlokalisierung

- Die Startposition ist unbekannt.
- Es handelt sich um **absolute Positionierung**
- An welcher Position der Roboter befindet, entscheidet er durch Auswerten seiner Sensordaten und durch erkennen von **signifikanten** Umgebungsmerkmalen
- **Mögliche Methode:** Triangulation

Kidnapped Robot Problem

- Die Position des Roboters ist anfangs bekannt
- Der Roboter wird willkürlich mit temporär deaktivierten Sensoren an eine beliebige andere Position versetzt, ohne darüber informiert zu werden.
- Auch dann muss das Verfahren robust die Position wiederfinden, zunächst muss der Roboter dies erkennen und sich dann relokalisieren
- Es muss eine erneute globale Lokalisierung durchgeführt werden

3.3 Relative Lokalisierung versus Absolute Lokalisierung

Relative Lokalisierung

- auch: **lokale, inkrementelle Lokalisierung** oder 'tracking'
- Relativ zu einer Startpose wird sukzessiv die Änderung der Pose an diskreten, aufeinanderfolgenden Zeitpunkten ermittelt und integriert

3.4 Transformation von Koordinatensystemen lokale \leftrightarrow globale

Absolute Lokalisierung

- auch als **globale Lokalisierung** bezeichnet
- Die Pose wird in Bezug auf ein externes Bezugssystem ermittelt, z.B. einer Karte oder einem globalen Koordinatensystem

Ziel Bestimme oder schätze die Position und Orientierung des Roboters in seiner Umgebung basieren auf

- der Eigenbewegung
- durch Messungen der relativen Position zu unterscheidbaren Objekten in der Umgebung in Roboterkoordinaten (Ultraschall, Laser, Kamera)

3.4 Transformation von Koordinatensystemen lokale \leftrightarrow globale

Kinematik Die Kinematik ist die Lehre der Beschreibung von Bewegungen von Punkten im Raum. Dabei werden die Größen Weg, Geschwindigkeit und Beschleunigung betrachtet. Die Kinematik ist ein Teilgebiet der Mechanik.

Kinematische Robotermodell

- kreisförmiger Roboter
- Zweiradantrieb
- Bewegung in der Ebene

Lokales Koordinatensystem

- mit dem Roboter verbunden
- Ursprung in der Mitte der Antriebsachse
- x-Achse zeigt in Richtung des Roboterfrontteils

3.5 Karten für statistische und dynamische Umgebungen

Festlegung der Roboterposition im globalen Koordinatensystem

- durch die Koordinaten $M(x_M, y_M)$ im globalen Koordinatensystem
- durch den Winkel θ zwischen der lokalen x-Achse und der globalen x-Achse
- Pose p gegeben durch: $p = (x_M, y_M, \theta)^T$

Transformation von lokalen in globale Koordinaten

- Koordinaten von P im lokalen Koordinatensystem: $p_l = (x_l, y_l)^T$
- Koordinaten von P im globalen Koordinatensystem: $p_g = (x_g, y_g)^T$
- Transformation von p_l nach $p_g(m = (x_M, y_M)^T) : p_g = R(\theta)p_l$

Transformation von globalen in lokale Koordinaten

- Transformation von globalen in lokale Koordinaten $p_l = R(\theta)^{-1}(p_g - m) = R(-\theta)(p_g - m)$

3.5 Karten für statistische und dynamische Umgebungen

- Generell gilt: **Karten** sollen eine explizite Repräsentation des Raumes sein.
- Die Karten sind auf die Sensorik des Roboters zugeschnitten
- Die Karten sind nicht vorrangig für den menschlichen Betrachter bestimmt, sondern der Roboter soll sie effizient nutzen können.

Statische Umgebungen

- basierend auf der Annahme, dass sich zwar der Zustand des Roboters innerhalb der Umgebung, nicht jedoch die Umgebung selbst ändert.
- Karte spiegelt die wirkliche Umgebung wider.

Dynamische Umgebungen

- Objekte können ihre Lage oder ihren Zustand ändern

3.5 Karten für statistische und dynamische Umgebungen

- In der Karte registrierte Objekte können verschwinden, nicht registrierte Objekte auftauchen
- 'Lernende' Karten sind ein fundamentales Problem in der mobilen Robotik

3.5.1 Mapping Methoden

Weltzentriert

- Die Pose aller Objekte einschließlich des Roboters werden in der Umgebung in Bezug auf ein festes Koordinatensystem repräsentiert.
- **Indoor**: Ursprung kann eine Zimmerecke sein
- **Outdoor**: Benötigung eines globalen Koordinatensystems, wie z.B. die Längen- oder Breitengrade, i.d.R. nutzen von **WGRS**(World Geographic Reference System)

Roboterzentriert gebraucht um bspw. Kollisionen zu vermeiden

- Durch Koordinaten-Transformation kann zwischen den verschiedenen Referenz-Frames konvertiert werden

3.5.2 Arten von Modellen

List of Figures

2.1	Schema der Hybridmodell Schichten	7
2.2	Filesystem, das ROS zugrunde liegt	10

Listings