

---

# **ROBOTER-ROBOTER INTERAKTION ZUR AUTONOMEN OBJEKTÜBERGABE**

**Master-Abschlussarbeit,**

eingereicht am Fachbereich Technik  
der FH Bielefeld

von Robin Rasch,  
geboren am 05.02.1991 in Minden

3. Januar 2016

---

Mit meiner Unterschrift bestätige ich, dass ich die Arbeit selbstständig und nur mit den zugelassenen Hilfsmitteln erstellt habe.

Minden, den .....

## Zusammenfassung

Diese Arbeit befasst sich mit der Entwicklung und Implementierung eines autonomen Roboter Systems. Dieses System bildet das Szenario einer Objektübergabe zwischen zwei selbstständigen Manipulatoren ab. Bei diesen beiden Manipulatoren handelt es sich um den Typ-gleichen Roboterarm des YouBot der Firma Kuka. Einer der Arme ist dabei stationär auf einem Tisch fixiert, der andere auf der mobilen YouBot Plattform.

Nach einer kurzen Einführung über das Nutzen von Robotern im Haushalt, sowie im intelligenten Gebäude, werden zunächst die Grundlagen und folgend ein aktueller Stand der Technik zusammengefasst. Anschließend wird sich diese Arbeit mit den zentralen Aspekten der Entwicklung und Implementierung befassen. Bei der Entwicklung wurden verschiedene Thematiken und Probleme der Robotik und Computer Vision berücksichtigt und bearbeitet. So wird in den folgenden Kapiteln auf das Setup der Roboter eingegangen, sowie den Themen der inversen Kinematik, der Segmentierung und Objekterkennung. Da das System in einem intelligenten Gebäude zum Einsatz kommen soll, ist die Thematik der Vernetzung einzelner Subsysteme und deren Zusammenspiel ebenfalls ein Bestandteil dieser Arbeit. Die Implementierung befasst sich mit der Algorithmik der einzelnen Probleme. Dabei wird Code exemplarisch vorgestellt, die vollständige Implementierung befindet sich im Anhang. Den Abschluss der Arbeit bilden eine Beurteilung und ein Fazit der umgesetzten Lösung, auch wird ein Ausblick über mögliche zukünftige Weiterentwicklungen gegeben.

## Abstract

This paper considers the development and implementation of an autonomous robotic system. This system describes a scenario from a handover between two independent manipulators. These two manipulators are equally robot arms from the Kuka YouBot. One is stationary fixed at top of a table, the other one is on a mobile YouBot base.

After a short introduction about the harness of service robots at smart houses, this work outlines the principles and the state of the art. Following this paper attends to the key aspects of the development and the implementation. The development observes different topics and set of problems for robotic and computer vision. There will be a variety of subjects like robover setup, inverse kinematic, segmentation and object recognition in the following chapters. Based on the operation at smart houses, networking and distributed systems are a topic in this paper, too. The implementation sink in algorithmic and solutions for single problems. There will be snippets of programm code, the complete code is at the appendix. A valuation and a conclusion of the implemented solutions, and also a prospect of possible future projects, form the ending of this work.

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Grundlagen</b>	<b>5</b>
2.1 Nomenklatur . . . . .	5
2.2 ROS: Robotic Operating System . . . . .	6
2.2.1 Package und Nodes . . . . .	6
2.2.2 Master und Parameter Server . . . . .	7
2.2.3 Topics und Messages . . . . .	8
2.2.4 Services . . . . .	9
2.2.5 Action-Lib . . . . .	9
2.3 Kinematik . . . . .	10
2.3.1 Mechanismus . . . . .	10
2.3.2 Kinematik . . . . .	11
2.3.3 Forward Kinematics . . . . .	11
2.3.4 Inverse Kinematik . . . . .	12
<b>3 Stand der Technik</b>	<b>14</b>
3.1 Multi-robot system - Konzepte zur Verwaltung mehrerer Roboter . . . . .	14
3.1.1 Klassifizierung der Koordinierung . . . . .	14
3.1.2 Probleme der Koordinierung . . . . .	15
3.1.3 Zusammenfassung . . . . .	18
3.2 Physically Embedded Intelligent Systems - PEIS . . . . .	19
3.2.1 Konzept . . . . .	20
3.2.2 Implementation . . . . .	20
3.2.3 Zusammenfassung . . . . .	21
3.3 Grasping und Handover - Arbeiten zu Roboterarmen . . . . .	21
3.3.1 Grasping - Greifen mit einem Roboterarm . . . . .	21
<b>4 Laboraufbau</b>	<b>23</b>
4.1 Aufbau Teststand . . . . .	23
4.2 YouBot . . . . .	23
4.2.1 Manipulator . . . . .	23
4.2.2 Mobile Plattform . . . . .	25
4.2.3 Rechner . . . . .	27
4.3 Sensoren . . . . .	28
4.3.1 Asus XTion Pro . . . . .	28
4.3.2 Hokuyo URG-04LX-UG01 . . . . .	28
4.3.3 Argos 3D - P100 . . . . .	29
4.3.4 Netzwerk- und Sensoranbindung . . . . .	30
<b>5 Zusammenfassung und Ausblick</b>	<b>32</b>

<b>6 Danksagung</b>	<b>33</b>
<b>7 Literaturverzeichnis</b>	<b>34</b>
<b>A Erster Anhang</b>	<b>39</b>
<b>B Noch ein Anhang</b>	<b>39</b>

## **Abbildungsverzeichnis**

1	Roboter in Seniorenheimen . . . . .	2
2	Beispiel Szenario 1 . . . . .	3
3	Erweiterung Szenario 1 . . . . .	3
4	Master Service . . . . .	7
5	Netzwerk Entlastung durch Peer-to-Peer Verbindungen . . . . .	8
6	Topic Beispiel . . . . .	8
7	Three-Layer Architektur . . . . .	17
8	Kompetenzen PEIS . . . . .	19
9	YouBot Arm Kinematik . . . . .	24
10	Arbeitsraum YouBot. Bilderquelle:Florek-Jasinska [2015] . . . . .	25
11	YouBot Base . . . . .	26
12	Mobile YouBot Plattform mit Sensorplatte. Bilderquelle:Kuka [2015] . . . . .	27
13	Asus Xtion Pro Live . . . . .	28
14	Hokuyo URG-04LX-UG01 . . . . .	29
15	Argos 3D - P100 . . . . .	30
16	Schematische Darstellung der Verbindungen . . . . .	30

## **Tabellenverzeichnis**

1	Nomenklatur . . . . .	5
2	YouBot Arm Joints . . . . .	24

# 1 Einleitung

Der Demographische Wandel in Deutschland stellt die Gesellschaft und die Politik vor ein großes Problem. Nicht nur der fehlende Nachwuchs, der ein Arbeitnehmerloch hinterlässt, sondern auch eine immer älter werdende Gesellschaft sorgen für viele Fragezeichen. Neben kommunalen Problemen, wie teure Infrastruktur, ist eins der größten Anliegen die Altersarmut. Ein sinkendes Rentenniveau und steigende Kosten werden es in Zukunft unmöglich machen für eine gute Pflegeversorgung zu bezahlen. [Zandonella, 2013] Neben den hohen Kosten in der Pflege sind auch andere Faktoren die zu einer nicht akzeptablen Situation führen. So ist der Pflegeberuf zu unattraktiv für viele junge Menschen, wodurch auch hier ein großes Loch an Arbeitnehmern wahrzunehmen ist. Körperlich anstrengende Arbeit, die in kurzer Zeit ausgeführt werden muss, führen zu vielen physischen und psychischen Erkrankungen der Pfleger.[Badura, 2005]

Einen alternativen Weg bringt die Technik. Einfache zu bedienende Systeme können den Alltag vereinfachen und so auch älteren Menschen ein selbstständiges Leben ermöglichen. Alltägliche Aufgaben müssten nicht mehr von Pflegern übernommen werden, sondern könnten durch Maschinen erledigt werden. Schon heute können einzelne Kleinsysteme Haustätigkeiten wie Staubsaugen und Rasenmähen übernehmen. Ein Vorreiter auf diesem Gebiet ist das Roboterland Japan. Dort überlegte Kobayashi Hisato, Professor für Maschinenbau und Robotik an der Hosei-Universität in Tokio, sich schon 1999 Konzepte für Senioren-Service Roboter.[Wagner, 2009] Nach seinen Vorstellungen sollten dabei Roboter nicht autonom arbeiten, sondern von Familienangehörigen ferngesteuert und überwacht.[Kobayashi, 1999] Neben diesen Konzepten kann man aber auch schon angewandte Robotik in Japans Seniorenpolitik finden. Zwei Musterbeispiele zeigen dabei die unterschiedlichen Anwendungsszenarien für Roboter im privaten Umfeld. *Sinére Kōrien* der Firma Matsushita ist ein digitales Seniorenheim. Neben einer Smarthouse Anbindung gehört auch der Roboterteddy Kō-chan zur Ausstattung der einzelnen Zimmer. Dieser dient als Unterhaltungsroboter und Kommunikationsgerät mit den Pflegern. So kann mit den Kameraaugen im Teddy eine Aufnahme vom Raum gemacht werden und im Notfall den Pflegern eine Alarmmeldung geschickt werden. Weitere Sensoren, zu Beispiel Gewichtssensoren unter den Betten, geben Informationen über die Abwesenheiten von Patienten.[Wagner, 2009] Ein weiterer Anwendungsbereich für Roboter ist die *robotto serapi* (Robotertherapie). Dabei beschäftigen sich die Senioren mit tierähnlichen Robotern, wie Hunden, Seeroben oder Katzen. Im Zentrum der Therapie steht die Interaktion zwischen Patient und Roboter. Die Robotertherapie soll die Patienten aktivieren und deren Tagesabläufe abwechslungsreich gestalten. Außerdem steigert es die Kommunikation zwischen zwei Patienten, die am selben Roboter arbeiten. [Wagner, 2009]

Nicht nur in Japan, sondern auch in Deutschland wird sich mit dem Thema *Care(rob)bots* befasst. So finden sich unter dem Stichpunkt *Mensch-Maschine-Entgrenzungen* Studien zu der Thematik. Andere Untersuchungen befassen sich mit der Gegenseite, der Akzeptanz der Senioren für Roboter. So ergab eine Befragung der VDE-Studie "Mein Freund der Roboter", dass eine Mehrheit (56%) der Senioren Robotern im Haushalt offen gegenüber stehen und diese einem Pflege-/Altersheim vorziehen würden. Neben den bekannten Staubsauger- und Rasenmärobotern, sind es auch zukünftige Anwendung, wie ein roboterisierter Rollstuhl, die hohe



(a) Roboterteddy Kō-chan Quelle: [Panasonic, 2005]



(b) Roboterkatze zur Therapie Quelle: [Wagner, 2009]

Abbildung 1: Roboter in Seniorenheimen

Akzeptanzwerte erreichen. Die Studie zeigte aber auch, dass Senioren zunächst Robotern skeptisch gegenüberstehen. Spontan lehnten 40 Prozent der Senioren Roboter in ihrem privaten Umfeld ab, 60 Prozent empfanden Robotik sogar als unheimlich. jedoch zeigte sich, dass der Wunsch nach einer selbstständigen Lebensführung ein starker Faktor für die Akzeptanz ist. Dadurch ergibt sich eine Beliebtheit für Serviceroboter. So sind Roboter, die abgrenzbare Tätigkeiten im Haushalt selbstständig erledigen, sehr beliebt. Wichtige Kriterien für die Akzeptanz waren zudem die intuitive Bedienbarkeit, die Robustheit und die Flexibilität gegenüber unterschiedlicher Handicaps. Auch menschliche Faktoren wie Geduld, Verständnis, Höflichkeit und Achtung der Intimsphäre waren den Anwendern wichtig.[Meyer, 2011]

Neben den sozialen Forschungen gibt es in Deutschland auch ingenieurwissenschaftliche Ergebnisse auf diesem Gebiet. So beschäftigt sich das Cluster of Excellence Cognitive Interaction Technology (CITEC) in Bielefeld, zusammen mit der Stiftung Bethel, in der Forschung auf dem Gebiet der Unterstützung für Demenzkranke. Die dort entwickelten Assistenzsysteme helfen den Patienten unter anderem beim Zähneputzen und geben ihnen so ein Stück Selbstständigkeit zurück.

Viele Forschungen und Entwicklungen beschäftigen sich momentan mit der Mensch-Maschine/Roboter Beziehung. Dabei geraten die Roboter-Roboter-Interaktionen in den Hintergrund. Da aber heutige Roboter keine Allround-Lösungen bieten, sondern meist hochgradig spezialisiert sind und nur eine Tätigkeit ausführen können( zum Beispiel Saugroboter, Fensterwischroboter und weitere), ist das Zusammenspiel und die Komposition der Roboter wichtig. Das Szenario in Abbildung 2 zeigt, wie solch eine Komposition aussehen könnte.

Dabei führt jeder Roboter selbstständig und unabhängig von den anderen seine Arbeit aus. Die Kommunikation geht immer von der Zentralen Steuereinheit, dem *Master*, aus. Zwischen den einzelnen Robotern findet kein großer Informationsaustausch statt. Die Änderung an dem Szenario in Abbildung 3 ändert jedoch die Art der Zusammenarbeit und der Kommunikation. Nun ist es nötig, dass die Roboter notwendige Informationen, wie die Übergabepose oder eine Synchronisierung der Gripper, austauschen.

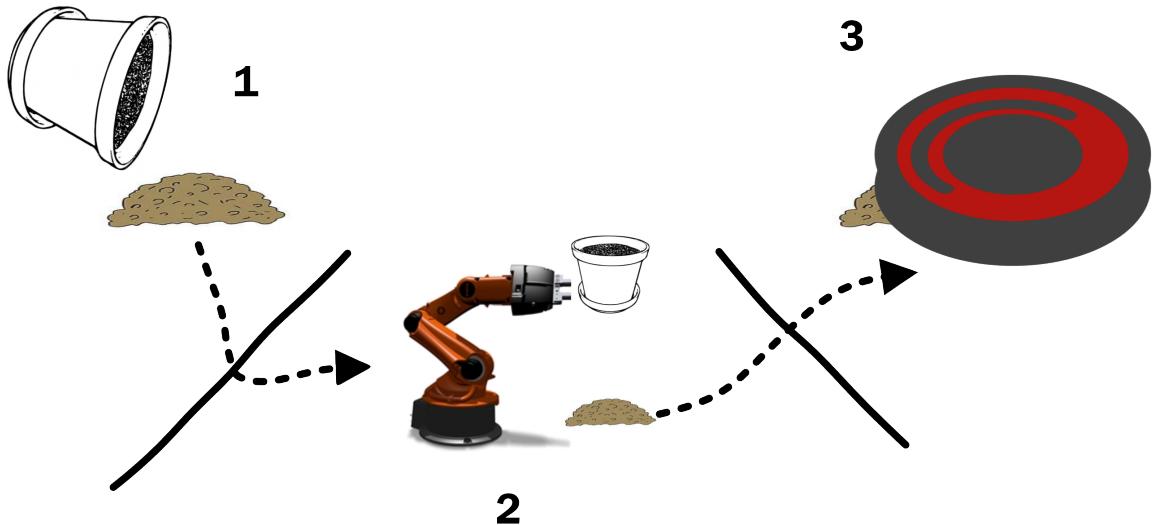


Abbildung 2: Mensch A stößt einen Blumentopf um (1). Ein Kamerasystem dediziert das der Topf umgefallen ist und Erde auf dem Boden liegt. Ein Roboter mit Arm wird gerufen, der zunächst die Vase aufhebt und an ihren Platz zurückstellt (2). Anschließend wird ein Saugroboter aktiviert, der die Erde weg saugt (3). Bei schwereren Verschmutzungen wird noch ein Wischroboter bestellt.

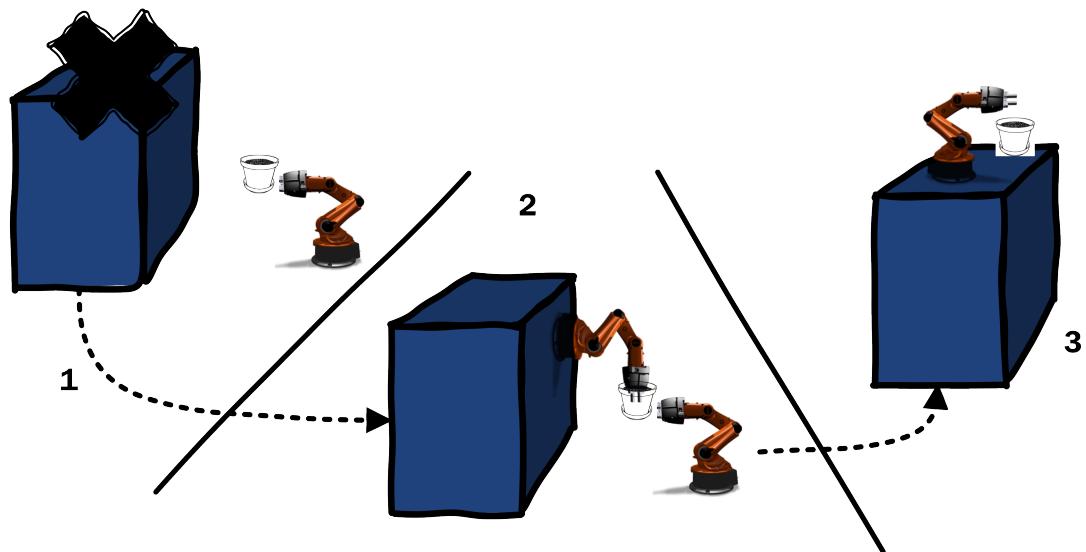


Abbildung 3: Beim Zurückstellen der Vase stellt der Roboter fest, dass er die gewünschte Position nicht erreichen kann(1), weil sein Arm zu kurz ist. Da auf der Anrichte noch ein weiterer Arm steht nimmt er die Kommunikation mit diesem Manipulator auf. Sie kommunizieren einen Übergabepunkt und führen ein Übergabemanöver durch. Arm 2 stellt nun die Vase an die gewünschte Position.

Der zentrale Aspekt dieser Arbeit befasst sich mit der Abstraktion von Szenario 2. Roboter 1 übergibt ein Objekt an Roboter 2. Dabei werden verschiedene Thematiken aufgegriffen, untersucht und erläutert. Das Folgende Kapitel befasst sich mit den Grundlagen dieser Arbeit, es werden einzelne Begriffe aus der Robotik und der Middleware **R**obot **O**perating **S**ystem erläutert, sowie der genutzte Laboraufbau dargestellt. Dabei wird die verwendete Hardware aufgelistet und im Detail betrachtet. Kapitel 3 zeigt schon bestehende Arbeiten im Bereich Robotik, die mit dieser Arbeit in Zusammenhang bestehen. So wird unter anderem der Begriff PEIS (Physically Embedded Intelligent Systems) erläutert und die Verbindung zur Thesis gezeigt. Des Weiteren werden Arbeiten für Multirobot Systeme und Handover-Methoden erklärt. Im darauf folgenden Kapitel

## 2 Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen für die folgenden Kapitel. Es soll dem geneigten Leser Informationen zum Verständnis der Arbeit geben. Die anschließenden Unterkapitel beschreiben zunächst die verwendeten Mathematischen Ausdrücke und Bezeichner um die Lesbarkeit der späteren Formeln zu erleichtern. Nachfolgend wird die Middleware ROS genauer erläutert und unterschiedliche Funktionen sowie Einschränkungen aufgezeigt, die Auswirkungen auf die entwickelte Softwarearchitektur hatten. Darauf folgt eine Definition des Begriffs der Kinematik, welcher im späteren Verlauf der Arbeit für die Entwicklung der inversen Kinematik benötigt wird. Das abschließende Unterkapitel beschreibt die verwendete Hardware und den Laboraufbau im Detail. Weitere Informationen bezüglich der Robotik und grundlegender Mathematischer Begriffe sind im Buch "Robotics, Vision and Control" von Peter Corke zu entnehmen.

### 2.1 Nomenklatur

Die folgende Tabelle erläutert die genutzten mathematischen Symbole und orientiert sich an dem Buch "Robotics, Vision and Control" Corke [2011]. Von links nach rechts sind das Symbol, die Einheit und die Beschreibung gegeben. Einige Symbole werden mehrfach verwendet und haben je nach Kontext eine andere Bedeutung.

Symbol	Einheit	Beschreibung
$T$	s	Messintervall
$T$		homogene Transformation, $T \in SE(2)$ oder $SE(3)$
${}^A T_B$		homogene Transformation zur Darstellung von Koordinatensystem $B$ betrachtet von Koordinatensystem $A$ . Koordinatensystem 0 bezeichnet das Weltkoordinatensystem und muss nicht explizit angegeben werden ${}^0 T_B = T_B()$ . Die inverse kann auf zwei Arten betrachtet werden: $({}^A T_B)^{-1} = {}^B T_A$
$\theta$	rad	Winkel im Bogenmaß. Wenn nichts weiteres gegeben ist sind Winkel immer im <b>Bogenmaß</b> zu sehen.
$\boldsymbol{\theta}$	rad	Vektor von Winkel im Bogenmaß.
$\theta_r \theta_p \theta_y$	rad	Roll-Pitch-Gear Winkel, beziehungsweise Rollen
$\alpha$	°	Winkel in Grad. Wenn nichts weiteres gegeben ist sind Winkel immer im <b>Bogenmaß</b> zu sehen.
$v$	$m s^{-1}$	Geschwindigkeit.
$\boldsymbol{v}$	$m s^{-1}$	Geschwindigkeitsvektor. Meist $\in \mathbb{R}^3$ .
$\omega$	$rad s^{-1}$	Drehwinkel-Geschwindigkeit.
$\boldsymbol{\omega}$	$rad s^{-1}$	Drehwinkel-Geschwindigkeitsvektor. Meist $\in \mathbb{R}^3$ .
$\boldsymbol{v}$		Geschwindigkeit-Drehung (engl. Twist, Screw). $\boldsymbol{v} \in \mathbb{R}^6$ , $\boldsymbol{v} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$
X,Y,Z		Kartesische Koordinaten.
$\xi$		Abstrakte Darstellung einer 3-dimensionalen Kartesischen Pose $\xi \in \mathbb{R}^6$
${}^A \xi_B$		Abstrakte Darstellung einer <b>relativen</b> 3-dimensionalen Kartesischen Pose, Pose $B$ betrachtet von Pose $A$
$\oplus$		Hintereinanderausführung (Addition) zweier Posen
$\ominus$		Unärer Operator zur Invertierung von Posen

Tabelle 1: Nomenklatur

## 2.2 ROS: Robotic Operating System

Das Robotic Operating System, kurz ROS, ist ein Middleware-Framework zur Steuerung und Verwaltung von Personal-Robotern. Es verwendet eine Paket und Node Struktur um einzelne Aktoren und Sensoren einzelner Robotersystem zu betreiben. Das ROS dient dabei als Abstraktionsebene für die Entwickler und versucht die Anwendungsebene von der Hardwareebene zu trennen. Das Framework wird am Robotikinstitut Willow Garage in Zusammenarbeit mit der Open Source Robotics Foundation entwickelt, stammt aber ursprünglich vom Stanford Artificial Intelligence Laboratory. Dort wurde 2007 im Rahmen des Stanford-AI-Robot-Projektes die Entwicklung aufgenommen. ROS ermöglicht während der Entwicklung und des Betriebs ein Cross-Plattform System, da es kompatible Versionen für Windows (experimentell), Linux und Mac OS X gibt. Außerdem steht ein Multi(programmier)sprachen System zur Verfügung. So ist es möglich eigene Nodes mit C++, Python oder Java zu entwickeln. Quigley et al. [2009] Neben den Standardkonzepten und -paketen gibt es eine Community, die eigene Pakete auf der ROS-Homepage zur Verfügung stellt. Die folgenden Abschnitte erläutern die ROS-Kernkonzepte, welche für die Entwicklung eingesetzt wurden.

### 2.2.1 Package und Nodes

*Packages* dienen der strukturellen Verwaltung aller Dateien im ROS-System. Nodes, Launch-Files, Services, Actions und Messages sind immer an ihr Paket gebunden und können nur unter deren Namen erreicht werden. Für die Verlinkung und die Erreichbarkeit gebauter Pakete ist das ROS-interne Buildsystem *Catkin* zuständig. Es benötigt ebenfalls den **eindeutigen** Package-Namen für den Build und Verwaltungsprozess. Neben den ROS bezogenen Daten können in einem Paket aber auch ROS unabhängige APIs angelegt und verwaltet werden. So werden unter anderem einzelne Pakete für Treiber genutzt. Ein ROS Paket folgt dem "Goldilocks"-Prinzip: Genug Funktionalität um nützlich zu sein, aber nicht zu viel, damit die Verwendbarkeit nicht zu komplex wird. Der Aufbau eines ROS Pakets findet sich in Anhang

Neben der strukturellen Verteilung gibt es auch die Funktionale Trennung. Dies wird durch einzelne *Nodes* erreicht. Jeder Node hat einen bestimmten Aufgabenbereich. Erst das Zusammenspiel mehrerer Nodes ergibt ein Robotersystem. So ist ein Node zum Beispiel für die Berechnung der Inversen Kinematik zuständig, ein anderer für die Lokalisierung des Roboters im Raum und ein dritter für das User-Interface. Diese Verteilung der Funktionalitäten hat mehrere Vorteile. Da jeder Node in einem eigenen Prozess gestartet wird ist das komplette System robuster. Jeder Node muss so entwickelt werden, dass er eigenständig weiterlaufen kann, auch wenn ein benötigter Node durch einen Fehler abgestürzt ist. Jeder Node wird mit einem System weit einmaligen Namen aufgerufen unter dem er vom System erreicht und angezeigt werden kann. Nodes kommunizieren untereinander mit Hilfe von Topics (siehe 2.2.3). Soll ein ROS Node gestartet werden kann dieser mit Hilfe des ROS-Befehls aufgerufen werden:

```
rosrun <paket-name> <nodetype-name>
```

Der Paketname und der Nodetype sind erforderliche Felder. Darüber hinaus können Parameter

angegeben werden die für den Node notwendig sind. Eine weitere Möglichkeit einen und mehrere Nodes zu starten ist ein Launchfile (siehe Anhang). In diesem können Nodes, Parameter und Argumente aufgelistet werden. Zu beachten ist jedoch, dass die Nodes in einer zufälligen Reihenfolge gestartet werden. Ein Launch-File kann mit folgendem Befehl gestartet werden:

```
roslaunch <paket-name> <launchfile-name>
```

Damit ein Node gestartet werden kann muss vorher ein ROS-Master aktiv sein, mit dem sich der Node verbinden kann. Ist der Node verbunden kann er auf die Topics, Services, Actions und den Parameter Server des Master zugreifen.

### 2.2.2 Master und Parameter Server

Der ROS Master ist ein zentraler Dienst für ganze System. Er beinhaltet einen Namen- und Registrierungs-Service an dem sich andere Nodes, auch Services, anmelden können. Der Master wird benötigt, damit die Nodes sich gegenseitig im System verbinden können. Der Nachrichten Transport geht nicht direkt über den Master er verbindet nur die beiden Nodes, welche anschließend via einer Peer-to-Peer Verbindung kommunizieren.

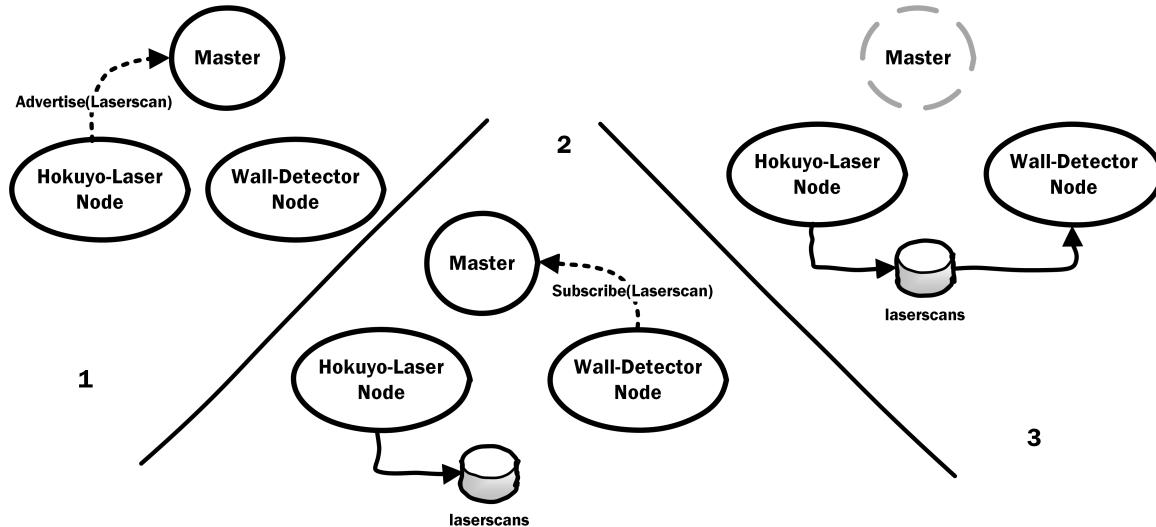


Abbildung 4: Master Service: (1) Der Hokuyo Laser Node meldet sich beim Master an, das er Laserscans zur Verfügung stellt. (2) Der Laser Node stellt seine Daten zur Verfügung und der Wall Detector meldet sich beim Master, dass er Laserscans benötigt. Der Master stellt die Verbindungsinformationen bereit und zieht sich dann zurück (graue Markierung). (3) Die Verbindung zwischen Laser Node und Detector Node ist hergestellt.

Dieses Verfahren reduziert die Netzwerklast stark im Vergleich zu einem Broadcasting-Verfahren. Tests zeigten, dass ein Kamera-Node (openni) ein Netzwerk mit seinen 3D-Punktwolken vollständig auslastet. Dabei wurde die Punktfolge auf der physikalischen Maschine erst erzeugt und auf eine zweite Maschine via WLAN übertragen. Die Delay-Zeit der Wolken war dabei sehr hoch  $> 1$  Sekunde und wurde, je länger die Übertragung dauerte höher. Werden die Daten jedoch auf der Maschine vorverarbeitet und nur noch die wichtigen Daten übertragen, sowie die Übertragungsrate reduziert, ist die Übertragung der Daten unkritisch. Nun gehen nur

noch die Verbindungsdaten zwischen Nodes und Master über das Netzwerk und die reduzierte Datenmenge.

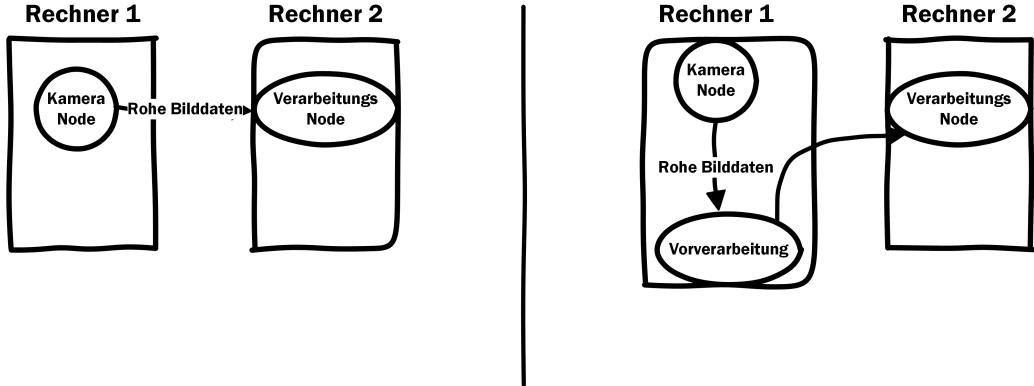


Abbildung 5: Netzwerk Entlastung durch Peer-to-Peer Verbindungen. Links: Ohne Datenvorverarbeitung, Rechts: Mit Datenvorverarbeitung

Neben dem Namen-Service verwaltet der ROS Master auch den Parameter Server. Dieser besteht aus einem Dictionary, einer großen Key-Value-Map, welches global erreicht werden können. Nodes haben so die Möglichkeit vordefinierte Daten ab zugreifen oder selber welche zu schreiben. So können Sensoren oder Aktoren kalibriert und konfiguriert werden. Der Zugriff auf diese Daten hat keine hohe Performance. Er ist für einen statischen Zugriff gedacht und kann mit einer Baum-Struktur geordnet werden, so lassen sich zusammenhängende Daten mit einer Abfrage am Server abrufen.

### 2.2.3 Topics und Messages

ROS bietet für die Kommunikation zwischen Nodes ein *Topic*-System. Jeder Topic entspricht dabei einem benannten Bus, auf welchen anonyme *Publisher* und *Subscriber* zugreifen können. Die Daten können dabei als TCP-Pakete (TCPROS) oder als UDP-Pakete(UDPROS) gesendet werden. Ein Topic besitzt einen eindeutigen Namen im System und einen eindeutigen Typen, der anhand des übersendeten Message-Typen definiert wird. Jede Topic kann nur einen Message-Typen versenden. Dieser Type ist nicht im Master bekannt, jedoch können sich Subscriber nur mit dem entsprechenden Typen anbinden. Die Anzahl der Subscriber und Publisher für einen Topic ist nur durch die Systemleistung beschränkt.

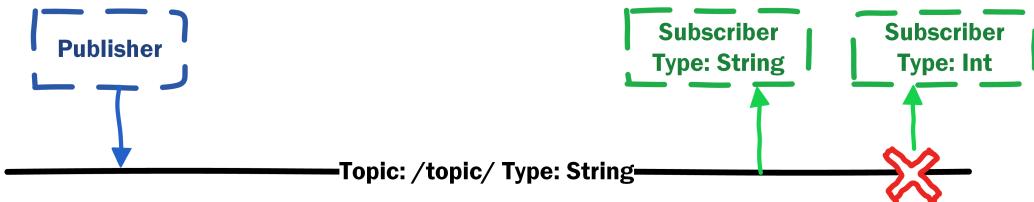


Abbildung 6: Beispiel für eine Topic mit dem Namen `/topic` und dem Type String. Ein Publisher schreibt Daten in den Bus, ein Subscriber liest diese aus. Die Verbindung eines zweiten Subscribers schlägt auf Grund des falschen Types fehl.

Publisher veröffentlichen ihre Daten in den Bus und haben nur einen schreibenden Zugriff. Sie

bekommen kein Feedback an wen die Daten gesendet worden sind oder ob die Daten überhaupt empfangen worden sind. Der erste Publisher, der sich auf einem Topic verbindet gibt den Typen vor.

Subscriber lesen die Daten vom Bus und geben sie für die Weiterverarbeitung weiter. Sie bekommen normalerweise keine Informationen über den Absender der Daten. Subscriber können sich nur bei einem Topic registrieren, wenn der Message-Type stimmt.

Subscriber und Publisher haben jeweils eigene Cache-Systeme die Nachrichten vor- oder zurückhalten können. Ein Topic ist ein einfachergerichteter Streaming-Dienst ist. Da in einem Roboter-System auch Remote Procedure Calls benötigt werden, um zum Beispiel Antworten auf Anfragen zu erhalten oder synchronisierte Aufgaben zu erledigen, wurde das Service-Paket als weiteres Kernkonzept angelegt.

#### 2.2.4 Services

*Services* ermöglichen es einen Remote Procedure Call Node übergreifend durchzuführen. Dazu muss Node A seinen Service am Master registriert haben und einen Service Server gestartet haben. Node B kann nun mit einem Service Client auf den registrierten Service zugreifen und eine Anfragen starten. Während die Anfrage verarbeitet wird, kann Node B nun je nach Anforderung blockieren oder weiter arbeiten. Node A verarbeitet nun die Anfrage und antwortet Node B mit dem Ergebnis. Der Service selbst, sowie die Parameter und die Rückgabewerte, sind für jeden Service eindeutig und werden vorher via zwei Messages, eine für die Request und eine für die Response, festgelegt. Analog zu den Topics werden Services am Master mit einem eindeutigen Namen angemeldet unter dem sie erreichbar sind. Dies ermöglicht auch einen einfachen Austausch von verschiedenen Implementierungen.

#### 2.2.5 Action-Lib

Ein aktiver Service schickt erst eine Rückmeldung an den Aufrufer, wenn er seine Aufgabe erledigt hat. Um auch einen Zwischenstand schicken zu können wurde die Actionlib eingeführt. Dieses Paket ist eigentlich keine Entwicklung der ROS-Entwickler sondern ein Community-Package. Inzwischen wurde es aber von der OSR-Foundation übernommen und als eins der Kernkonzepte eingestuft.

Analog zu den Services besitzt auch die Action-Lib einen Server- und einen Client Teil. Der Client fordert vom Server eine Aktion an, dabei übergibt er ein *Goal*. Dieses beinhaltet die benötigten Parameter für den Server. Der Server arbeitet die Anfrage ab, dabei kann er auch *Feedback*, Zwischenstände, zurückschicken. Ist die Anfrage abgearbeitet schickt der Server ein *Result* an den Client zurück. Wie auch beim Service ist die Art der Aktion eindeutig. Die Anzahl der Parameter bei Goal, Feedback und Result sind fest definiert und können zur Laufzeit nicht geändert werden.

## 2.3 Kinematik

Dieses Unterkapitel befasst sich mit dem Begriff der Kinematik und den dazu gehörigen Begriffen des Mechanismus und der inversen Kinematik. Außerdem werden einige mathematische Lösungswege für verschiedene Probleme innerhalb der genannten Themen genannt.

### 2.3.1 Mechanismus

Ein Mechanismus ist der Grundbegriff von beweglichen Körpern. Dabei wird zwischen *Gliedern* (engl. Link,  $L = \text{Set of Links}$ ) und *Gelenken* (engl. Joint,  $J = \text{Set of Joints}$ ) unterschieden. Glieder sind die starren Körper eines Mechanismus und sind durch Gelenke miteinander verbunden. Ein Glied ist nicht auf ein Gelenk beschränkt, sondern kann mehrere haben ( $N_{joint} \in \mathbb{N}_0$ ). Auch die Verbindung zwischen zwei Gliedern kann durch mehrere Gelenke realisiert sein. Ein Gelenk wiederum ist auf genau zwei Glieder beschränkt ( $N_{link} = 0$ ) und ermöglicht so eine eingeschränkte relative Bewegung zueinander. Gelenke werden nicht als eigenständige physikalische Körper gesehen. Gelenkhälften, zum Beispiel die Schenkel eines Kippscharniers, werden als Bestandteil des damit verbundenen Glied betrachtet.

Glieder können mit zwei Parametern definiert werden, der Länge  $a$  und der Verdrehung  $\alpha$ :

$$l_i \in L$$

$$a_i = \text{Länge von } l_i \quad (1)$$

$$\alpha_i = \text{Verdrehung von } l_i \quad (2)$$

Gelenke werden ebenfalls mit 2 Parametern definiert. Der Gelenk-Offset beschreibt den Abstand zwischen den zwei Gliedern eines Gelenks auf der Gelenkkachse. Der Gelenkwinkel beschreibt die Rotation zwischen den beiden verbundenen Glieder im Bezug zur Gelenkkachse. Corke [2011]

$$j_i \in J$$

$$d_i = \text{Gelenk-Offset von } j_i \quad (3)$$

$$\theta_i = \text{Gelenkwinkel von } j_i \quad (4)$$

Im Anschluss werden nur noch die englischen Begriffe Link und Joint benutzt.

### 2.3.2 Kinematik

Die *Kinematik* eines Mechanismus beschreibt die Bewegungsmöglichkeiten der Links relativ zueinander. Es wird dabei abstrahiert, ob ein Joint motorisch angetrieben oder passiv bewegt wird. Die Kinematik betrachtet bei der Bewegung auftretende *Geschwindigkeiten* und *Beschleunigungen*. Auftretende Kräfte werden nicht in der Kinematik, sondern in der *Dynamik* untersucht. Zentrale Aspekte dabei sind die Trägheits- und Schwerkraft.

Die *Kinematische Kette* beschreibt den Aufbau eines speziellen Mechanismus. Dabei ist die Anzahl der Joints pro Link auf maximal 2 beschränkt ( $N_{joint} \leq 2$ ). Besitzt jeder Link genau zwei Joints gilt die Kette als geschlossen, ansonsten als offen.

$$\text{Kinematische Kette} = \begin{cases} \text{geschlossen} & \forall l \in L | l_{joints} = 2 \\ \text{offen} & \text{für Rest} \end{cases} \quad (5)$$

Für diese Arbeit ist nur die offene Kette interessant, da die Manipulatoren der meisten Roboter an beiden Enden (Base und End-Effektor) frei sind. Dadurch ergibt sich für  $N_{link} = N_{joint} + 1$ . Typischerweise wird der erste Joint mit dem Index 1 versehen und der erste Link mit dem Index 0. Link 0 ist die *Base* des Manipulators und Link N beinhaltet den *End-Effektor*. Durch die Bezeichnung ergibt sich, dass Joint j die Links j-1 und j mit einander verbindet und den Link j bewegt. Corke [2011]

Die Transformation eines Link-Koordinatensystems j-1 zu Link j wird durch elementare Rotation und Translation beschrieben:

$${}^{j-1}A_j(\theta_j, d_j, a_j, \alpha_j) = T_{Rz}(\theta_j)T_z(d_j)T_x(a_j)T_{Rx}(\alpha_j) \quad (6)$$

Durch die Vereinigung der einzelnen Matrizen ergibt sich:

$${}^{j-1}A_j(\theta_j, d_j, a_j, \alpha_j) = \begin{pmatrix} \cos\theta_j & -\sin\theta_j \cos\alpha_j & \sin\theta_j \sin\alpha_j & a_j \cos\theta_j \\ \sin\theta_j & \cos\theta_j \cos\alpha_j & -\cos\theta_j \sin\alpha_j & a_j \sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (7)$$

### 2.3.3 Forward Kinematics

Die Vorwärts-Kinematik (engl. Forward Kinematic) beschreibt die "vorwärts" Rechnung der Kinematik. Aus den gegebenen Joint-Informationen wird die Pose des End-Effectors berechnet. Hier reichen die Daten für den Gelenkwinkel  $\theta_j$  bei Drehgelenken und über den Winkel-Offset  $d_j$  bei Schiebe-/ Schubgelenken, da die restlichen Werte für den Mechanismus als konstant angesehen werden können. Die Forward Kinematic wird häufig als Funktion  $K$  angegeben Corke [2011]

$$\xi_E = K(\mathbf{q}) \quad (8)$$

$q$  entspricht dabei der aktuellen Joint-Konfiguration und  $\xi_E$  der Pose des End-Effectors. Durch die Kombination der Transformation aus Gleichung 6 ergibt sich für einen Manipulator mit N-JointsCorke [2011]

$$\xi_E \sim {}^0T_E = {}^0A_1 {}^1A_2 \dots {}^{N-1}A_N \quad (9)$$

### 2.3.4 Inverse Kinematik

Die Inverse Kinematik berechnet die Inverse der Forward-Kinematic. Sie bestimmt die möglichen Joint-Konfigurationen  $\mathbf{q}$  für eine Zielpose. Im Gegensatz zur Forward-Kinematic ist die Inverse Kinematik nicht eindeutig. Eine einfache Veranschaulichung kann man am menschlichen Körper sehen. Fixiert man das Handgelenk an einer Position kann man den Ellbogen in verschiedene Positionen bewegen ohne die Position der Schulter zu ändern. Für die Inverse Kinematik ergibt sich folgende FunktionCorke [2011]

$$\mathbf{q} = K^{-1}(\xi) \quad (10)$$

Aus Gleichung 8 und 10 ergibt sich auf Grund der Eindeutigkeit:

$$\xi = K(K^{-1}(\xi))$$

Aber **nicht**

$$\mathbf{q} = K^{-1}(K(\mathbf{q}))$$

Um eine Inverse Kinematik zu lösen gibt es drei Methoden: geometrisch (analytisch), numerisch und heuristisch.Steidl [2011]

Die geometrische Lösung nutzt die einfache geometrische Abstraktion des Mechanismus und versucht diesen mit trigonometrischen Funktionen abzubilden. Damit auch hier mehrere Lösungen bestimmt werden können, werden verschiedene Konfigurationen für bestimmte Abstraktionsmodelle geschaffen und alle Konfigurationen berechnet. Am Beispiel des Menschen wären mögliche Konfigurationen: Ellbogen oben oder Ellbogen unten. Bei der Verwendung der trigonometrischen Funktionen muss beachtet werden, dass die Umkehrfunktionen in bestimmten Bereichen nicht eindeutig, ungenau oder gar nicht definiert sind. Empfohlen wird dafür die Verwendung von atan2, der in den meisten Programmiersprachen vorhanden ist. Diese Methodik funktioniert nur für einfache Mechanismen bei denen möglichst alle Joints die selben Achsen haben.

Die numerische Lösung versucht durch Iterationen eine Näherung an die gewünschte End-Pose zu erreichen. Das ganze fällt in die Thematik der Optimierung. Für die numerische Lösung gibt es unterschiedliche Algorithmen. Dazu gehören Jacobian Transpose, Pseudo Inverse und Damped Least Square. Die numerischen Lösungen sind im Vergleich zur analytischen langsamer und ungenauer, jedoch lassen sich mit ihr größere und komplexere Manipulatoren berechnen.Steidl [2011]

Die heuristischen Lösungen nutzen abstrakte Bezüge und Vorhersagen für Veränderungen der End-Pose im Bezug zur Joint-Konfiguration um mögliche Konfigurationen zu bestimmen. Eine anschließende Güteberechnung (zum Beispiel euklidische Distanz) gibt an, ob eine Konfiguration verworfen oder angenommen wird. Wie für die numerische Lösung gibt es auch bei der heuristischen Lösung mehrere Algorithmen. Diese sind unter anderem Cyclic Coordinate Descent und Lagrange-Multiplier. Steidl [2011]

### **3 Stand der Technik**

In diesem Kapitel werden Themen bezogene Forschungen vorgestellt und untersucht. Da es zu der genauen Thematik kaum Forschungen gibt, werden vor allem Teilaspekte der Arbeit aufgezeigt. Dabei liegt der Blick unter anderem auf der Arbeit von Mathias Broxvall, in dieser wird die Einbindung eines Roboters in eine intelligente Umgebung untersucht. Des Weiteren werden Arbeiten zu Multirobot-Systeme, sowie verhaltensbasierten Steuerungen vorgestellt.

#### **3.1 Multi-robot system - Konzepte zur Verwaltung mehrerer Roboter**

Die Verwendung mehrerer individueller Roboter ist Forschungsthema seit 1986 und ist ein weit gestecktes Feld. Frühere Themen sind zelluläre Roboter, Motion Planning, Schwarmrobotik und Architekturen. Nach dem neuen Robotik Paradigma der behavior-based Kontrolle orientieren sich viele Forschungen an der Biologie und versuchen Abbildungen der sozialen Strukturen von Insekten und anderen Schwarmtieren zu nutzen. Parker [2003a] Dabei wurden zum Beispiel die Schwarmbildung (siehe Hayes and Dormiani-Tabatabaei [2002]), die Futtersuche (Balch [1999]) und die Spurverfolgung (Vaughan et al. [2000]) betrachtet. Viele Probleme von Multi-Robot Systemen (MRS) haben oft einen Bezug zu Problemen aus anderen Forschungsgebieten und können durch die Anzahl an Robotern meist robuster und schneller gelöst werden. Weitere große Forschungsbereiche sind die verteilte künstliche Intelligenz und der Multi-Agent Ansatz. Beide befassen sich mit der Kooperation zwischen Systemen. Erste Ergebnisse dieser Bereiche wurden mit einer Prioritätsordnung der Subtasks erreicht Durfee et al. [1987]. Spätere Arbeiten betrachteten das Gruppieren von Systemen, die Aufgaben miteinander bearbeiteten Shehory and Kraus [1998] und Lau and Zhang [2003]. Diese Gruppen, auch Koalitionen genannt, lösen Aufgaben, die nicht an einen einzelnen Roboter gestellt werden können. Die Arbeit Vig and Adams [2005] überführte dann die bekannten Lösungen der Koalitionen in die Probleme der MRS. Weitere Themen aus dem Bereich Multi-Agent, Team-Work (Pynadath and Tambe [2003]), Potenzial-Management (Timm and Woelk [2003]) und Normen (Boella [2002]), wurden für die Koordinierung von MRS angewandt. Lundh et al. [2006]

##### **3.1.1 Klassifizierung der Koordinierung**

Die Koordinierung von Robotern beschreibt die Organisation von Bewegungen, welche die Roboter zusammen ausführen sollen. Potenzielle Probleme sind dabei die Fragestellungen wie die Koordination organisiert werden soll, wer organisieren soll oder wie viel Koordination nötig ist um eine Aufgabe zu erledigen.

Bei den Organisationsmöglichkeiten unterscheidet man zwischen zentralisiert und verteilt. In einem stark verteilten System sind die einzelnen Roboter unabhängig und selbstständig. Entscheidungen werden meistens zwischen den Systemen ausgehandelt. Diese Variante ist gegenüber einem zentralen System robuster, da kein Single-Point-of-Failure auftritt. Dafür ist es meist komplexer eine optimale Lösung für ein Problem zu finden. Außerdem ist der Kommunikationsaufwand wesentlich höher. In einem stark zentralisiertem System übernimmt ein System die Füh-

rung und alle Entscheidungen. Das ganze kann dann als ein Robotersystem mit ferngesteuerten Sensoren und Aktoren abgebildet werden. Nachteilig ist hier der beschriebene Single-Point-of-Failure: Fällt das zentrale System aus, kann das ganze System nicht mehr arbeiten. Außerdem sind solche Systeme meist langsamer, da die ganzen Informationen erst auf einer Recheneinheit zusammengefasst werden müssen und anschließend dort ausgewertet werden. Dadurch können aber auch einfacher optimierte Lösungen gefunden werden. Lundh et al. [2006] Weitere Informationen finden sich in den Arbeiten Farinelli et al. [2004] und Dias and Stentz [2003].

Ein weiterer Aspekt der Koordinierung von MRS ist der benötigte Grad an Koordinierung. Dabei wird in der Literatur zwischen den Begriffen der engen (*tight*), beziehungsweise eng-gekoppelt oder starken, Koordinierung und der losen (*loosely*), lose-gekoppelt oder schwache, Koordinierung unterschieden. In der Arbeit Farinelli et al. [2004] ist die starke Koordinierung als eine Koordinierung beschrieben, die auf einem Koordinationsprotokoll aufbaut. Ein Koordinationsprotokoll beschreibt eine Menge an Regeln die spezifizieren wie Roboter miteinander interagieren. Eine schwache Koordinierung baut nicht auf einem solchen Protokoll auf. Die Begriffe eng (eng-gekoppelt) und lose (lose-gekoppelt) definieren die Quantität der Koordinierung. Ein eng-gekoppeltes MRS benötigt viel Koordinierung, ein loses MRS eher wenig. Da die Begriffe viel und wenig ungenau und schwer zu definieren sind, existieren in der Literatur verschiedene Definitionen. Die Arbeit Kalra et al. [2004] unterteilt die Koordinierung in *loosely*, *moderately* und *tightly* anhand der Zerlegung der Aufgaben (Tasks) in Unteraufgaben (Subtasks). Bei einer losen Koppelung können alle Tasks in Subtasks zerlegt werden, die von einem Roboter ausgeführt werden können. Mäßige Kopplungen begrenzen die Koordination auf zeitlich abhängige Subtasks. Dabei beschränkt sich die Koordination nur auf den Startzeitpunkt ( $t_0$ ) eines Subtasks. Die Ausführung des Tasks wird nicht koordiniert. Eine enge Kopplung erfordert eine permanente Koordination und Tasks können nicht in Subtasks zerlegt werden, die von einem Roboter abgearbeitet werden. Lundh et al. [2006]

Eine weitere Klassifizierung der Literatur bezieht sich auf die Tasks. In der Definition von Gerkey and Matarić [2004] wird zwischen *single-robot tasks* (SR) und *multir-robot tasks* (MR) unterschieden. SR benötigen genau einen Roboter für die Ausführung, zum Beispiel eine Bewegung an eine gewünschte Position. MR können mehrere Roboter benötigen. Außerdem werden Roboter eingegliedert. *Multi-Task* (MT) Roboter können gleichzeitig mehrere Tasks ausführen, *Single-Task* Roboter führen immer nur Tasks nacheinander aus. Lundh et al. [2006]

### 3.1.2 Probleme der Koordinierung

Bei lose-gekoppelten MRS ist das Hauptproblem die Zuweisung der Tasks, da jeder Task ein SR ist. Dieses Problem wird als *Task allocation* bezeichnet. Darauf baut das Problem der Rollenzuweisung auf, wenn im MRS mit einem Rollensystem gearbeitet wird. Task Allocation beschreibt wie eine Anzahl von Tasks auf eine Anzahl von Systemen verteilt wird. Die einfachste Aufteilung wurde schon 1960 in Gale [1989] beschrieben und seitdem in zahlreichen Arbeiten untersucht: Ein Task kann nur einem Agenten zugewiesen werden und jedem Agenten kann nur eine Task zur Zeit abarbeiten.

Ein bekannter Lösungsansatz ist das ALLIANCE-System von Lynne E. Parker aus dem Jahr 1998 Parker [1998]. Dieser Algorithmus nutzt vier einfache Schritte zur Vergabe der Tasks. In einer Pärchen-Liste von Task-Agent sind alle möglichen Kombinationen im MRS gespeichert und werden nach der Nützlichkeit bewertet. (1) Finde das am besten bewertete Pärchen (2) weise die Task dem Agenten zu (3) entferne das Pärchen aus der Liste (4) ist die Liste nicht leer beginne mit (1), ansonsten terminiere. Der Vorteil von ALLIANCE liegt nicht in der Zuweisung, sondern in der Neuzuweisung. Dazu werden zwei Aspekte der Verhaltenssteuerung genutzt: Ungeduld (*impatience*) und Fügung (*acquiescence*). Ungeduld ermöglicht es, dass ein Agent eine Task eines anderen Agenten übernimmt. Wenn Roboter A die Eindruck bekommt, dass Roboter B die Task nicht lösen kann, wird er ungeduldig und übernimmt die Aufgabe. Die Fügung funktioniert ähnlich, nur das Roboter A feststellt, dass er selbst die Task nicht oder nicht gut genug lösen kann und diese an Roboter B abgibt. Weitere Forschungen gibt es unter anderem bei Werger and Matarić [2000].

Ein weiterer bekannter Ansatz für das Problem ist das *Contract Net Protocol* (CNP) von Davis und Smith aus dem Jahr 1983. Davis and Smith [2003]. Das CNP beruht auf einem Auktionshausprinzip. Tasks werden von einem Dealer als Broadcast angeboten. Agenten im MRS haben die Möglichkeit privat auf die Tasks zu bieten. Dabei werden meist Ergebnisse des auszuführenden Task geboten. Diese Ergebnisse können unter anderem Energie- und Zeitkosten beinhalten. Der Dealer entscheidet wer den Zuschlag bekommt. Er kann auf Grund des Gebotes den bestmöglichen Agenten für die Task bestimmen. Die Auswahl kann dynamisch, je nach Priorität (zum Beispiel Zeit vor Energiekosten), ausgewählt werden. Varianten die auf CNP basieren sind GOFER Caloud et al. [1990], M+ Botelho and Alami [1999], TraderBots Dias and Stentz [2000] und Sold! Gerkey and Matarić [2002].

Bei Rollenbasierten Ansätzen werden einzelnen Agenten Rollen zugewiesen und dadurch in ihrer Funktionalität beschränkt. Agenten mit den gleichen Rollen haben die gleichen Funktionalitäten. Ansätze dafür finden sich in den Arbeiten Frias-Martinez et al. [2005], Vail and Veloso [2003] und Stone and Veloso [1999].

Bei eng-gekoppelten MRS ist die primitive Zerlegung wie bei losen Systemen nicht möglich. Dadurch fällt die Koordinierung nicht nur in der Startphase an, sondern auch während des Prozesses. Zentrale Probleme sind nicht nur der Ausführer, sondern die Frage nach der Ausführung, also wie eine Task erledigt wird. Ein möglicher Anwendungsfall ist die Arbeit von Saffiotti et al. [2000]. Bei diesem Ansatz arbeiten Roboter in einer Formation. Jeder Roboter besitzt dabei zwei Vorsätze, das Team-Ziel und das Individualziel. Das Team-Ziel ist das übergeordnete Gemeinschaftsziel, zum Beispiel das Tragen eines Objektes. Das Individualziel sind Roboter-eigene Interessen, zum Beispiel die Kollisionsvermeidung. Typische Lösungsansätze für solche Anwendungen ist das Führer-Folger Prinzip, bei dem ein Führer bestimmt wird, dieser übernimmt Entscheidungen, die Folger folgen dem Führer. Dies ist bei komplexeren Anwendungen jedoch nicht ausreichend, sodass eine Planung notwendig wird. Dabei treten die beiden zentralen Fragen auf: **Wer macht was?** und **Wie wird es gemacht?**.

Das Robotik Institute der Carnegie Mellon University hat eine Forschungsgruppe, die sich mit der Frage nach dem *wer* beschäftigt. Dabei werden drei Ansätze verfolgt. Die erste beschäftigt

sich mit der Aufgabenverteilung bei komplexen Tasks, die zweite mit der Aufgabenverwaltung bei eng-gekoppelten MRS und die dritte mit Architekturen für enge MRS. Als Grundlagen dient den drei die schon erwähnte Arbeit von Dias und Stentz, den TraderBotsDias and Stentz [2000] die auf dem CNP aufbauen. Die Arbeit Zlot and Stentz [2005] befasst sich mit der Aufgabenverteilung bei komplexen Tasks. Dabei werden komplexe Tasks als Aufgaben mit mehreren Lösungswegen definiert. Die Erstellung eines Plans für eine komplexe Task liegt meist in *NP-hard*. Im Gegensatz dazu stehen die einfachen (simple) Tasks, die einfach vorwärts gerichtet sind. Ähnlich dem Auktionsverfahren wurde so ein Auktionsansatz für komplexe Tasks entwickelt. Anstatt eine primitive Task anzubieten werden Taskbäume angeboten. Dabei entspricht die Wurzel dem abstrakten komplexen Task, die Knoten den komplexen Subtasks, die Blätter den primitiven Subtasks und die Kanten dem Koordinationsaufwand. Die einzelnen Agenten können nun auf die Blätter bieten. Die zweite Forschungsrichtung befasst sich mit einem Marktansatz für eng-gekoppelte Systeme und wird in den Arbeiten Kalra et al. [2004] und Kalra et al. [2005] behandelt. Das Konzept trägt den Namen Hoplites und lässt sich aus dem antiken Griechenland herleiten. Dort waren die Hopliten eine der besten Infanterieeinheiten und auf komplexe Manöver spezialisiert. Bei diesem Framework werden Agenten rückwirkend bewertet. Die Bewertung erfolgt nach Beendigung eines Task und gibt an wie sehr dieser Agent dem Gemeinschaftsziel gedient hat. Des weiteren nutzt das Framework aktive und passive Koordinierung ein. Passive Koordinierung wird für simple Tasks eingesetzt und solange genutzt, bis die aktive Koordinierung bessere Vorhersagen bietet. Das Framework selbst hat keinen Planer für komplexe Aufgaben, sondern entscheidet nur, wann welcher Planer eingesetzt werden soll. Der dritte Aspekt behandelt Architekturen für eng-gekoppelte MRS und heterogenen Roboter. Die Arbeiten Simmons et al. [2001] und Simmons et al. [2002] zeigen eine drei Schichten Architektur. Diese Schichten (Planungs-, Ausführungs- und Verhaltensschicht) können innerhalb eines Roboters miteinander kommunizieren. Außerdem können die einzelnen Schichten auch Roboter übergreifend kommunizieren. Also Planungsschicht von Roboter A mit Planungsschicht von Roboter B, aber nicht mit Ausführungsschicht von Roboter B. Lundh et al. [2006]

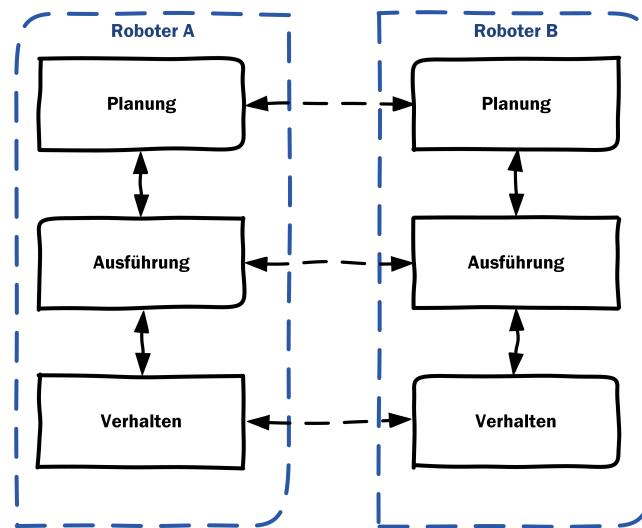


Abbildung 7: Drei Schichten Architektur zur Koordinierung von eng-gekoppelten Systemen.  
Übernommen aus: Simmons et al. [2002]

Die schon zitierte Lynne E. Parker leitet die Forschungsgruppe des *Distributed Intelligence Laboratory* an der University of Tennessee. Die zentralen Forschungen der Gruppe befassen sich mit der automatischen Koordination von Roboterteams. 2003 und 2004 präsentierte Parker zwei Arbeiten Parker [2003b] und Parker et al. [2004] in denen Netzwerke von über 70 Robotern eng koordiniert wurden. Dabei wurde zwischen sensorarmen und -reichen Robotern unterschieden. Für die enge Kopplung wurde dabei ein Führer-Folger Prinzip gewählt. Als Führer wurden die sensorreichen Roboter ausgewählt. In der Arbeit wurden zwei Tasks bearbeitet. Die erste wurde als Long-Distance-Navigation beschrieben, dabei übernimmt ein Führer die Kontrolle und berechnet einen Weg mit seinen Sensoren. Die sensorarmen Roboter folgen dem Führer. Am Ziel angekommen bauen alle Roboter ein gemeinsames Sensornetzwerk auf und überwachen den Zielraum. Die Roboter innerhalb des Sensornetzwerkes versorgen sich dabei automatisch mit den benötigten Informationen. Die Konfiguration für dieses MRS ist jedoch statisch durch die Entwickler vorgegeben. In der anschließenden Veröffentlichung Parker et al. [2005] stellte die Forschungsgruppe *ASyMTRe* (Automated Synthesis of Multi-robot Task solutions through software Reconfiguration) vor. Dieses Konzept, beruhend auf der Schemen Theorie aus Arkin [1987], ermöglicht einem MRS eng-gekoppelte Tasks durch Informationsaustausch zu lösen. Dazu werden verschiedene Schemen so miteinander verbunden, dass die Informationen von den Umweltsensoren zu den Motor-Schemen gelangen. ASyMTRe nutzt dazu einen Algorithmus, der alle Informationen für den am wenigsten fähigen Roboter sammelt und diese anschließend an alle verteilt. Anschließend wird der Roboter als Führer ermittelt, der den größten Informationsgehalt bei kleinsten Sensorik hat. Lundh et al. [2006]

### 3.1.3 Zusammenfassung

Dieses Kapitel zeigte Arbeiten und Konzepte rund um das Thema Multi-Robot Systeme auf. In dieser Zusammenfassung sollen die benötigten Informationen für diese Arbeit extrahiert und Entscheidungen für die eigene Entwicklung getroffen werden. In diesem Kapitel wurden zunächst die Begriffe Koordination und Konfiguration durch verschiedene Arbeiten definiert. Außerdem wurden die Begriffe verteilte und zentrale, starke und schwache, sowie enge und lose Koordination erläutert. In den Arbeiten stellte sich heraus, dass zentrale Steuerungen durch einen Single-Point-of-Failure durch Störungen stärker betroffen sind als verteilte Koordination. Diese wiederum benötigt einen größeren Aufwand der Algorithmen und der Konfiguration. Daraus folgt für diese Arbeit, dass ein erster Entwurf eine zentrale Koordinierung haben wird, da der Aufwand beschränkt ist. Jedoch soll eine Schnittstelle für eine verteilte Koordinierung vorgesehen werden. Aus der Literatur ergab sich, dass starke Koordination auf einem Koordinationsprotokoll aufbauen und schwache auf dynamischen Algorithmen, die beteiligte Robotersysteme auswählen. Da in dieser Arbeit nur zwei Roboter verwendet werden und diese zwangsläufig miteinander arbeiten müssen ist die Auslegung eines dynamischen Zuweisungsmodells überflüssig, soll aber auch in Form von Schnittstellen für weitere Aufgaben vorgesehen werden. Die Begriffe der eng- und lose-gekoppelten MRS wird durch verschiedene Definitionen unterschiedlich ausgelegt. Für diese Arbeit wird auf die Definition von Kalra et al. [2004] zurückgegriffen. Da in dieser die Begriffe anhand der Zerlegung von Tasks definiert werden, kann die Kategorisierung

für dieses System erst nach der Anforderungsanalyse geschehen. Auch die Eingliederung der Tasks in Single-Robot und Multi-Robot Tasks kann erst nach der Anforderungsanalyse erfolgen. Dies betrifft ebenfalls die Entscheidung für Single-Task oder Multi-Task Roboter. Auf ein Rollensystem soll zunächst verzichtet werden, da auch dies für dieses kleine System unnötiger Aufwand ist. Eine Erweiterung die für dieses System vorgesehen ist, beschreibt die Arbeit Davis and Smith [2003] mit dem CNP. Die Vergabe der Aufgaben mit Hilfe eines Auktionshauses kann in zukünftigen Entwicklungsschritten auch für komplexere, sogar eng-gekoppelte, MRS genutzt werden. Im folgenden Kapitel wird die Architektur PEIS vorgestellt, die neben Robotersystemen auch weitere Sensoren und Aktoren einschließt. Diese Architektur soll als Ausgangsarchitektur dienen.

### 3.2 Physically Embedded Intelligent Systems - PEIS

Ähnlich der Einleitung zu dieser Arbeit beschreiben auch Broxvall und Kollegen ihre Zukunftsvision von Robotern im sozialen und alltäglichen Umfeld. Im Fokus ihres Szenarios stehen dabei die unterschiedlichen Sensoren und Aktoren in einem intelligenten Gebäude, im Zusammenspiel mit Robotern. Dieses Zusammenspiel ist das zentrale Konzept der Arbeit von Broxvall und wird als PEIS-Ökologie bezeichnet. PEIS steht für Physically Embedded Intelligent Systems und umfasst die Kernkonzepte von künstlicher Intelligenz, allgegenwärtigem Rechnen und Robotik. Die Grafik 8 zeigt die Überschneidung dieser Themenbereiche, sowie die Position an denen Broxvall und Kollegen ihre Arbeit sehen. Broxvall and Saffiotti [2005]

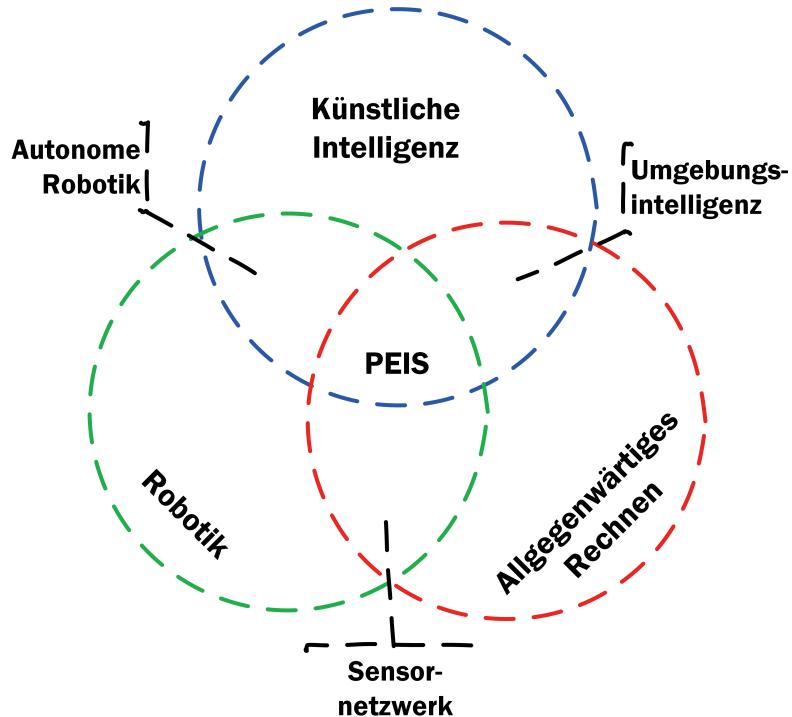


Abbildung 8: PEIS liegt in den Schnittmengen mehrerer Kernkompetenzen. Übernommen aus: Broxvall and Saffiotti [2005]

### 3.2.1 Konzept

Das Konzept der PEIS-Ökologie beruht auf drei Grundlagen. Zunächst wird die gleichmäßige Notation von PEIS (*uniform notion of PEIS*) vorgestellt. Jedes technische Gerät ist eine mögliche PEIS-Komponente, wenn es über eine Recheneinheit und eine Kommunikationsschnittstelle verfügt, sowie mit der Umwelt interagieren kann. Dies betrifft sowohl Akteure als auch Sensoren und reicht vom einfachen Toaster über den intelligenten Feuermelder bis zum komplexen Roboter. Das zweite zentrale Konzept ist das einheitliche Kommunikationsmodell (*uniform communication model*). Dieses Modell realisiert die Dynamik eines PEIS. Zu jedem Zeitpunkt können sich neue PEIS-Komponenten anmelden und bestehende Komponenten das Modell verlassen. Das Modell „ermöglicht eine einheitliche Kommunikation und versteckt die Heterogenität zweier PEIS-Komponenten unabhängig vom physikalischen Kommunikationsmedium.“ Broxvall and Saffiotti [2005] Die dritte Kernkompetenz ist das einheitliche Kooperationsmodell (*uniform cooperation model*). Dabei beschreiben Broxvall und Kollegen die Verbindung funktionaler Komponenten. Diese Verbindung ermöglicht es jeder PEIS-Komponente Funktionen anderer PEIS-Teilnehmer anzufordern. Diese drei Konzepte repräsentieren die Stärke eines PEIS: Die Zusammenarbeit mehrerer unabhängiger Systeme. Broxvall and Saffiotti [2005]

### 3.2.2 Implementation

Neben dem Konzept der PEIS-Ökologie stellen Broxvall und Kollegen in ihrer Arbeit die Implementation einer PEIS-Ökologie vor. Dabei wird zunächst nur die Abstraktion der Robotersysteme, sowie der Umwelt angestrebt. Dazu werden funktionelle Aufgaben und physikalische Systeme getrennt und zu einander zugewiesen. So werden zum Beispiel Kameras und Lasersensoren für die visuelle Raumwahrnehmung, Saugroboter für die Hindernisbeseitigung und die Reinigung eingeteilt. Neben der Funktionalität müssen die PEIS-Komponenten Schnittstellen anbieten, damit andere PEIS-Komponenten die Funktionen anfordern können. Der nächste Schritt ist die Entwicklung des Kommunikationsmodells. Broxvall und Kollegen empfehlen die Verwendung eines Hybridmodells zwischen einem Event-basierten und einem Tuple-Space Modell. Diese Empfehlung lässt sich auf die Grafik 8 zurückführen. Arbeiten in den Bereichen Umgebungsintelligenz (siehe dazu Arregui et al. [2003] und Siegemund [2004]), Sensornetzwerke (siehe Adjie-Winoto et al. [1999] und Heidemann et al. [2001]) sowie Robotik (Caceres et al. [2003]) setzen auf diese Kommunikationsmodelle, sowie auf die Hybridform. Für PEIS wird ein Eventbus-System ähnlich den ROS-Topics gefordert. Dieses ermöglicht das autonome An- und Abmelden von Komponenten. Das Tupel-Space Modell kann für die PEIS-Ökologie simple gehalten werden und wird von Broxvall und Kollegen mit der Form beschrieben:

```
<peisID, compID, key, val0, ..., valN>
```

peisID und compID identifiziert dabei die PEIS-Komponente, sowie die innere logische Einheit. key bezeichnet das Tupel selbst. Die dritte Komponente der Implementierung ist die Konfiguration der Ökologie. Diese beinhaltet alle Informationen über die Funktionalitäten des System, sowie die Verbindungen der einzelnen Komponenten untereinander. Dazu werden die Informationen ebenso in Tupeln gespeichert wie beim Kommunikationsmodell. Durch diesen Ansatz

können Methoden aus der künstlichen Intelligenz genutzt werden um Aufgabenbereiche für einzelne PEIS-Komponenten zu arrangieren. Lundh et al. [2005] Der vierte und letzte Bestandteil der Implementierung von Broxvall und Kollegen ist der PEIS-Kern. Dieser Kern ermöglicht beinhaltet das Kommunikationsmodell, sowie die Konfiguration. Der Kern verbindet sich selbst mit weiteren Instanzen von PEIS-Kernen, die er im selben Netzwerk erkennt. Dies funktioniert unabhängig vom Typen des Netzwerkes. PEIS-Kerne tauschen ihre Konfigurations-Tupel untereinander aus. Dadurch kann eine PEIS-Komponente eine Funktionalität bei einer anderen Komponente finden und anfordern. Broxvall and Saffiotti [2005]

### **3.2.3 Zusammenfassung**

Das Konzept der PEIS-Ökologie funktioniert nach dem Prinzip des teile und herrsche. Unterschiedliche Funktionalitäten werden nicht in einem allmächtigen Roboter System zusammengefasst, sondern in spezialisierte Roboter-, Aktoren- und Sensoren-Systeme verteilt. PEIS schafft eine Möglichkeit diese unterschiedlichen Systeme zusammen arbeiten zu lassen. Der Gedanke mit dem Tupel und dem Event-System ergibt sich logischerweise aus der benötigten Dynamik des Modells. Bedenklich ist dabei die lose Struktur innerhalb einer Tupels, da Daten nicht ordentlich gewartet werden können. Auch der Schritt der Konfiguration ist umständlich, da so nur starre Verbindungen zwischen einzelnen Komponenten hergestellt werden können. Durch eine Lockereitung der Konfiguration und einem CNP-System kann die Aufgabenverteilung optimiert werden. Ansonsten ist das Konzept praktikabel und zweckgemäß für das Zusammenspiel von Robotern und einer intelligenten Umgebung und soll im Rahmen dieser Arbeit verwendet werden.

## **3.3 Grasping und Handover - Arbeiten zu Roboterarmen**

Dieses Kapitel befasst sich mit Arbeiten zu dem Thema Roboterarmen. Zunächst wird das Greifen eines Gegenstandes, *Grasping*, aufgezeigt. Dabei werden Arbeiten zum allgemeinen Greifen, zur Erkennung geeigneter Greifpositionen an Gegenstände und Kamera-Unterstütztem Grasping aufgeführt. Im Anschluss wird die Thematik der Übergabe, dem zentralen Thema dieser Arbeit, aufgegriffen und Literatur zu Roboter-Mensch Übergaben betrachtet.

### **3.3.1 Grasping - Greifen mit einem Roboterarm**

Als Ausgangsarbeit für dieses Thema bietet sich die Arbeit Bicchi and Kumar [2000] an. In diesem werden die Grundlagen des Greifens erläutert. Dabei wird zunächst die menschliche Hand und ihre Verwendungszwecke betrachtet. Der Mensch nutzt seine Hand zur Erkundung, zum Festhalten und zur Manipulation von Objekten. Das Erkunden von Objekten ist der eigene große Forschungsbereich *Haptik* und kann bisher, auf Grund von fehlender Sensorik, in der Robotik nicht vollständig eingesetzt werden. Zwischen dem Festhalten, Fixieren und der Manipulation wird in der Arbeit unterschieden, sowie zwischen dem Manipulieren mit den Fingern und der Manipulation mit dem gesamten Arm. Dies liegt vor allem an den Anwendungs- und Forschungsbereichen. Während das Fixieren eines Objektes, beziehungsweise die Manipulation

mit dem gesamten Arm, in der Industrie oft eingesetzt wird, ist die filigrane Arbeit mit den Fingern noch ein Thema der Forschung. Die ersten Arbeiten zu diesem Thema der Robotik gehen auf ? und ? zurück. Neben dem Greifen mit den Fingern der Hand gibt es auch Griffe mit dem Kompletten Arm, bezeichnet mit *whole arm grasps* (?), (?) und *power grasps* (?). Bicchi and Kumar [2000]

Das Greifen unterliegt physischen Grenzen, dabei wird der Griff, beziehungsweise das Halten, unter anderem durch den Kraftvektor und dem Haftkoeffizienten beeinflusst. Ein Griff wird durch  $N$  Kontakte beschrieben. Dabei wird angenommen, dass alle Kontakte punktuell sind. Auch ein Kontakt auf einer Linie oder Oberfläche wird durch zwei oder mehr Punktkontakte abgebildet. Die Literatur unterscheidet diese Kontakte in reibungslose, reibungsbedingte oder weiche Punktkontakte. Ein reibungsloser Kontakt kann nur eine Kraft entlang der gemeinsamen Normalen erzeugen. Ein reibungsbedingter kann neben der normalen auch eine tangentiale Kraft und ein weicher Kontakt zusätzlich einen Drehmoment erzeugen. Die Kontaktart ist abhängig von den Oberflächeneigenschaften des Grippers und des Objektes. Bei einer gummiähnlichen Oberfläche des Grippers wird das Kontakt als weich modelliert. Haben Gripper und Objekt beide harte und raue Oberflächen wird ein reibungsbedingter Kontakt angenommen. Sind die Kontaktstellen auf Gripper und Objekt glatt und ist ein kleiner Reibungskoeffizient gegeben, gilt der Kontakt als reibungslos. Bicchi and Kumar [2000]

An jedem Kontakt ist das Zielobjekt einer normalen Kraft, einer tangentialen Kraft und einem Drehmoment um die Normale unterworfen. Diese werden als Drehung  ${}^i w_N$ ,  ${}^i w_T$  und  ${}^i w_\theta$  notiert. Wobei jede Drehung als ein  $6 \times 1$ -Einheitsvektor, zusammengesetzt aus Kraft und Drehmoment, definiert ist. Die dazugehörigen Intensitäten werden als  ${}^i c_N$ ,  ${}^i c_T$  und  ${}^i c_\theta$  angegeben. Die  $N$  Kontakte werden nun in den Submatrizen  $w_N$ ,  $w_T$ ,  $w_\theta$ ,  $c_N$ ,  $c_T$  und  $c_\theta$  geordnet. Die Drehungen werden nun als  $W$  und die Intensitäten als  $c$  zusammengeführt. Außerdem wird noch  $g$  als bekannte äußere Drehung eingeführt. Die kann eventuell 0 sein. Folgende Definitionen und Eigenschaften können nun für Griffe aufgestellt werden ?:

Ein gegriffenes Objekt ist im Gleichgewicht  $\Leftrightarrow$

1.  ${}^i c_N \geq 0 \mid \forall i \in N$  (1. Kontaktbedingung)
2.  $| {}^i c_T | \leq {}^i \mu_T {}^i c_N \mid \forall i \in N$  (2. Kontaktbedingung, Coulombsches Gesetz)
3.  $| {}^i c_\theta | \leq {}^i \mu_\theta {}^i c_N \mid \forall i \in N$  (3. Kontaktbedingung, abgewandeltes Coulombsches Gesetz ?)
4.  $Wc + g = 0$  (Statisches Gleichgewicht)

## 4 Laboraufbau

Dieses Kapitel befasst sich mit dem Aufbau der Versuchsanlage und der verwendeten Hardware. Dabei werden die Aspekte des räumlichen Aufbau genauso betrachtet wie die Netzwerkinfrastruktur und die genutzten Roboter und Sensoren.

### 4.1 Aufbau Teststand

### 4.2 YouBot

In dieser Arbeit werden zwei Roboter genutzt. Bei beiden handelt es sich um YouBots der Firma Kuka. Diese wurden 2010 erstmals auf der Automatica in München vorgestellt. Kuka ist ein deutscher Hersteller von Industrierobotern mit Sitz in Augsburg. Kuka hat aber auch Erfahrung im Bau von experimentellen Robotern wie etwas die Arme für Justin, einem Service-Roboter.

Die beiden YouBots unterscheiden sich beim Aufbau. YouBot 1 besteht aus einem Manipulator mit Gripper, YouBot 2 besteht aus Manipulator mit Gripper und mobiler Plattform mit omnidirektionalen Rollen. Zur besseren Unterscheidung und zur Lesbarkeit dieser Arbeit haben beide Roboter einen Namen bekommen. YouBot 1 entspricht dabei **Dummy**, YouBot 2 ist **Rose**. Im Weiteren Verlauf der Arbeit werden nur noch diese Namen genutzt um eine Verwechslung zu vermeiden.

#### 4.2.1 Manipulator

Die Roboterarme von Dummy und Rose sind eine offene kinematische Kette mit fünf Joints, die von der Basis aufsteigend nummeriert sind. An Joint 5 sind die Gripper montiert, diese bestehen aus zwei individuellen Fingern, die linear auf einer Achse bewegt werden können.

Joint 1 und Joint 5 sind zwei Drehgelenke um die Z-Achse, die bei ausgestreckter Haltung (Candle-Pose) parallel liegen. Joint 2, 3 und 4 sind Kippgelenke um die X-Achse, die immer parallel liegen. Durch diese Konstellation ergeben sich für den kompletten Arm fünf Freiheitsgrade (5-DOF). Durch die Einschränkung der Achsen auf die X- und Z-Achsen der einzelnen Joints ist ein seitlicher Griff bei  $x_{base} = 0$  oder  $y_{base} = 0$  nicht möglich. Die Höhe in Candle-Pose beträgt inklusiv Gripper 655mm. Die einzelnen Abstände lassen sich der Grafik 9 entnehmen.

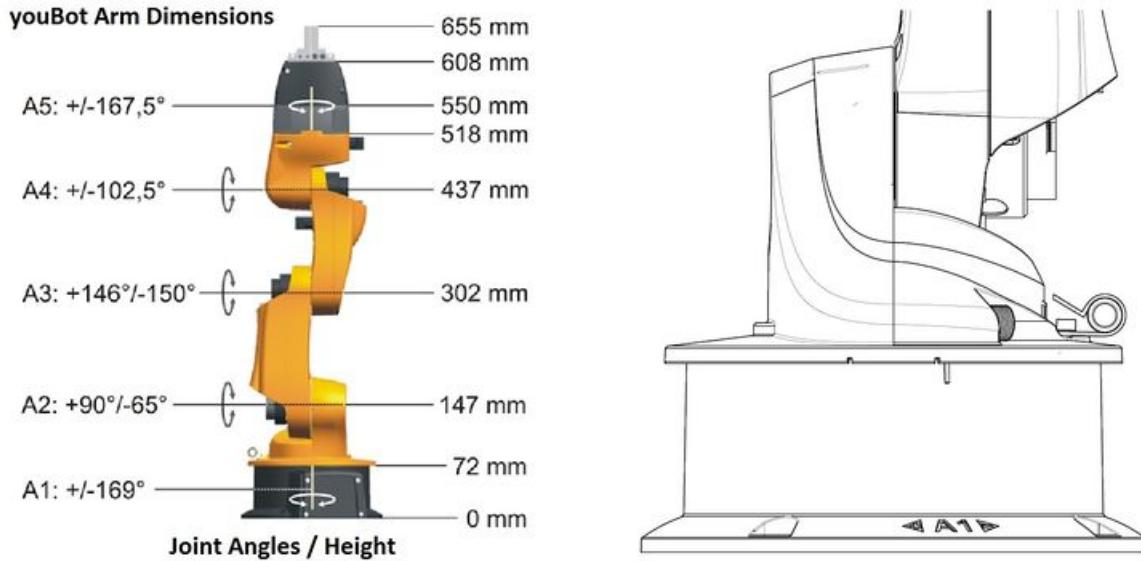


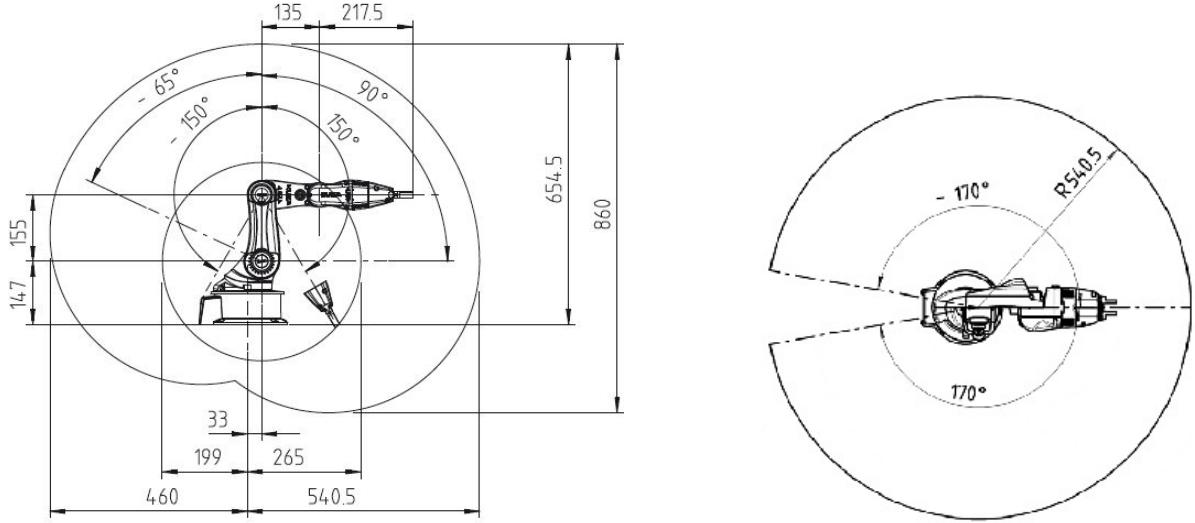
Abbildung 9: YouBot Arm Kinematik. Links: Detaillierter Mechanismus mit Joint und Link Angaben. Rechts: Vergrößerte Darstellung der Basis. Bildquelle: Florek-Jasinska [2015]

Tabelle 2 stellt noch einmal den Drehbereich aller Joints dar. Dabei sind neben Bezeichner auch die minimalen und maximalen Winkel, sowie die Winkelgeschwindigkeiten. Eine Besonderheit stellt Joint 3 dar, durch den Aufbau bedingt wurde die Achse innerhalb der Treiber gespiegelt, sodass die Winkel um den 0° Punkt gespiegelt wurden. In der Tabelle und den folgenden Zeichnungen ist diese Spiegelung nicht beachtet.

Bezeichnung	Max./Min. Winkel	Winkelgeschwindigkeit (rad / s)
Joint 1	$\pm 2.94$	$\frac{\pi}{2}$
Joint 2	$+ \frac{\pi}{2} / -1.13$	$\frac{\pi}{2}$
Joint 3	$+2.54 / -2.63$	$\frac{\pi}{2}$
Joint 4	$\pm 1.78$	$\frac{\pi}{2}$
Joint 5	$\pm 2.91$	$\frac{\pi}{2}$

Tabelle 2: YouBot Arm Joints. Quelle: Florek-Jasinska [2015]

Der Arbeitsraum des YouBot Arms beschränkt sich durch die Joints. Die Grafiken 10(a) und 10(b) stellen den Arbeitsbereich eingeschränkt dar. Die Darstellung 10(a) zeigt mit Hilfe von Konturen die einzelnen Bahnen unter Beschränkung der Joints, sowie der End-Effektor Ausrichtung. Die äußerste Kontur gibt die maximale Reichweite für den End-Effektor an. Die Z-Achse steht dabei orthogonal zur Tangente an der Konturposition. Abbildung 10(b) gibt eine Draufsicht auf den Arbeitsraum. Durch die Beschränkung von Joint 1 befindet sich ein toter Winkel im "Rücken" der Arms. Dieser kann durch einen Überschlag des ganzen Arm, insbesondere Joint 2 und 3, und einer 180 °Drehung von Joint 1 dennoch erreicht werden. Dies wird durch den linken Bereich in Abbildung 10(a) klar.



(a) YouBot Arm Arbeitsraum. Reichweite der einzelnen Gelenk und einzelne Abmessung der Links und Link-Offsets. Einheitslose Angaben sind in mm.

(b) Draufsicht YouBot Arm Arbeitsraum.

Abbildung 10: Arbeitsraum YouBot. Bilderquelle:Florek-Jasinska [2015]

#### 4.2.2 Mobile Plattform

Rose besitzt neben dem Arm noch eine mobile Plattform. Diese ist 590 mm lang, 380 mm breit und 140 mm hoch. Die Plattform hat vier Räder mit einem Durchmesser von 47.5 mm. Jedes Rad lässt sich getrennt ansteuern. Bei den Rädern handelt es sich um Mecanum-Räder, die eine Steuerung in alle Richtungen ermöglicht. Dazu sitzen auf jeder Felge sechs drehbare tonnenförmige Rollen die im Winkel von 45° zur Achse des Rades angebracht sind. Damit Omni-Direktionale Bewegungen möglich sind werden die Räder abwechselnd zum Nachbar auf +45° oder -45° angeordnet. Die minimale Geschwindigkeit der Plattform beträgt  $0.01 \frac{m}{s}$ , die maximale  $0.8 \frac{m}{s}$ . Abbildung 11 zeigt den Aufbau und Abmessungen der Plattform.

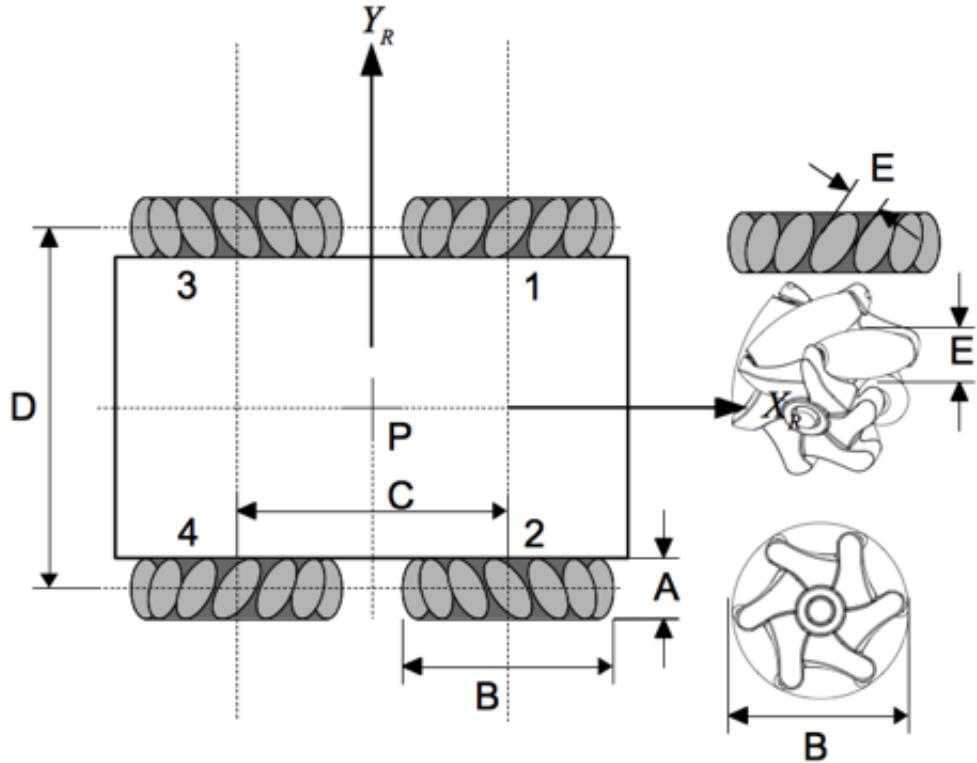
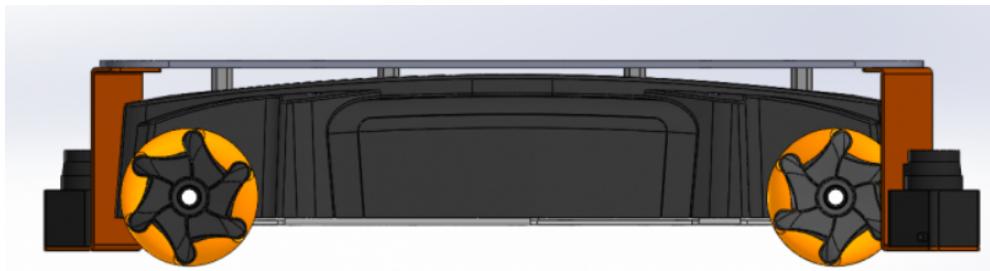
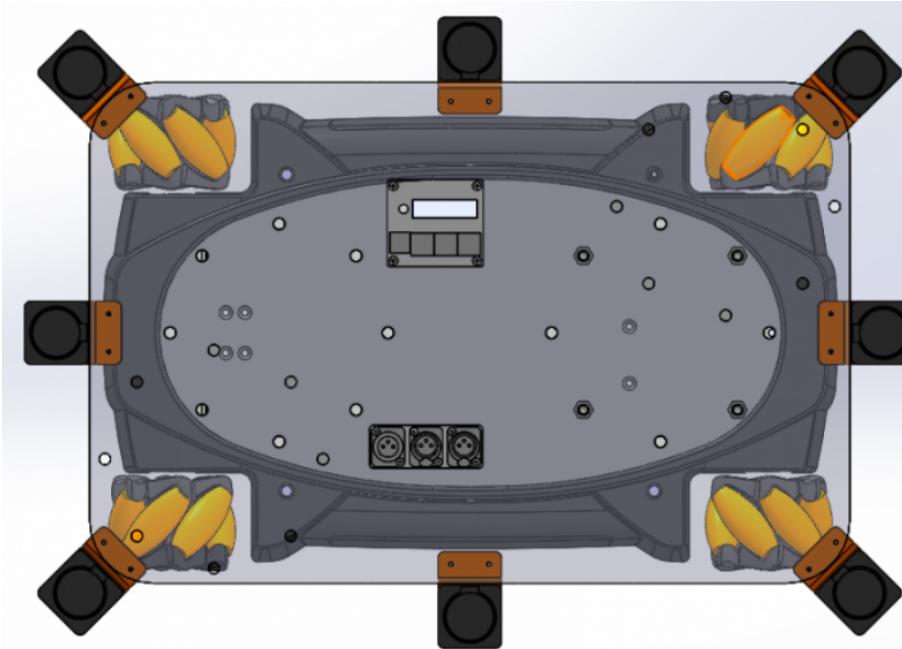


Abbildung 11: YouBot Base.  $A = 74.87 \text{ mm}$ ,  $B = 100 \text{ mm}$ ,  $C = 471 \text{ mm}$ ,  $D = 300.46 \text{ mm}$   $E = 28 \text{ mm}$ . Die Bodenfreiheit der Plattform beträgt 20 mm. Bildquelle: Florek-Jasinska [2015]

Die Plattform von Rose ist modifiziert und besitzt eine Aufsatzplatte. Diese dient zur Montage zusätzlicher Sensoren und zur Schutz der Räder bei Kollisionen. Diese ist 600 mm lang und 396 mm breit. Durch die Abstandsbolzen und die Dicke der Platte (5 mm) erhöht sich die Plattform auf 150 mm (siehe Abbildung 12(a)).



(a) Seitenansicht mobile Plattform mit Sensorplatte



(b) Draufsicht mobile Plattform mit Sensorplatte

Abbildung 12: Mobile YouBot Plattform mit Sensorplatte. Bilderquelle:Kuka [2015]

#### 4.2.3 Rechner

Für die Rechenleistung der beiden Roboter sorgen neben den verbauten, eingebetteten Boards zwei Computer mit Linux Betriebssystemen. Der für Rose zuständige PC ist ein Mini-ITX und in der mobilen Plattform eingebaut. Dieser läuft mit einem Intel Atom D510 Dual Core mit 1.66 GHz als CPU und 2 GB DDR2 RAM. Als Festplattensystem ist eine 32 GB SSD verbaut. Der Rechner stellt einige IO-Schnittstellen zur Verfügung: sechs USB 2.0, ein VGA und zwei LAN Anschlüsse sind nutzbar. Dabei ist ein LAN Anschluss für den Ethercat-Anschluss der Plattform belegt. Diese wiederum bringt nochmal zwei LAN Anschlüsse mit. Der Rechner für Dummy ist ein BLALBLA

Weitere Details zu den Armen, der Plattform oder den Rechnern befindet sich im Anhang.

## 4.3 Sensoren

Für die Erkennung von Objekten und der Positionierung im Raum sind mehrere Sensoren nötig. In dieser Arbeit wird dabei auf das Kamerasystem Asus XTion Pro Live und auf einen Laserscanner Hokuyo URG-04LX-UG01 zurückgegriffen.

### 4.3.1 Asus XTion Pro

Die XTion Pro ist ein Tiefenkamerasystem von Asus. Es besteht aus einer RGB-Kamera, einer Tiefen-Kamera und zwei Mikrofonen. Die Kamera wurde von Prime Sense entwickelt und ist eine um gelabelte Microsoft Kinect. Die XTion ist kleiner und leichter als die Kinect und damit besser für die Robotik geeignet. Angeschlossen wird die XTion mit einem USB Kabel und betrieben mit dem ROS Paket OpenNI2. Der Node published eine PointCloud mit RGB Daten, sowie eine PointCloud für die Tiefenkamera, als auch Bilder der RGB Kamera.

Das Sichtfeld der Kamera beträgt Horizontal 58°, 45°Vertikal und 70°Diagonal. Die Auflösung der Tiefenkamera beträgt bei 30 Frames pro Sekunde 640x480 und bei 60 FPS 320x240. Die Auflösung des RGB-Bildes 1280x1024. Die nutzbare Distanz der Tiefenkamera beträgt zwischen 800 mm und 3500 mm. Asus [2015] Diese Distanz schränkt die Nutzbarkeit der Kamera ein, so ist sie nicht für eine visuelle Unterstützung der Gripper geeignet, wenn sie an diesen montiert ist, da die Distanz zwischen Kamera und Gripper kleiner 800 mm ist.



Abbildung 13: Asus XTion Pro Live. Bildquelle: Asus [2015]

### 4.3.2 Hokuyo URG-04LX-UG01

Der URG-04LX-UG01 ist ein optischer Abstandsmesser. Dieser kann Entfernung in einer zirkulären Ebene messen. Entwickelt wurde er von der Firma Hokuyo, welche neben dem URG noch weitere, ähnliche Sensoren anbietet, die andere Reichweiten und Sichtfelder abdecken. Die Sensoren nutzen zur Messung der Entfernung eine Infrarot-Diode mit einer Wellenlänge von 785 nm. Die Distanz wird mit Hilfe der Phasenverschiebung berechnet. Diese Technik reduziert den Einfluss durch das Oberflächenmaterial des gemessenen Objektes. Versorgt wird der Sensor mit 5V DC, welche über den USB-Anschluss angelegt werden. Da die Startleistung des Sensors die

maximale Abgabeleistung eines einzigen USB-Anschluss überschreitet, müssen ein Y-USB-Kabel und zwei USB-Anschlüsse genutzt werden. Angesteuert wird der Sensor mit dem ROS-Paket Hokuyo-Node, welche eine PointCloud published.

Der URG-Sensor hat eine Reichweite von maximal 4000 mm und benötigt für eine Messung 100 ms, was einer Messrate von 10 FPS entspricht. Das Sichtfeld beträgt 240°. Die minimale Graduelle Auflösung beträgt 0.36°. Dadurch ergeben sich die ungefähren Disparitäten von 20 mm bei 2000 mm und maximal 40 mm bei 4000 mm Radius. Die Ungenauigkeit beträgt maximal 3 Prozent des gemessenen Wertes, was bei der maximalen Reichweite von 4000 mm einer Abweichung von 120 mm entspricht. MAEDA [2009]

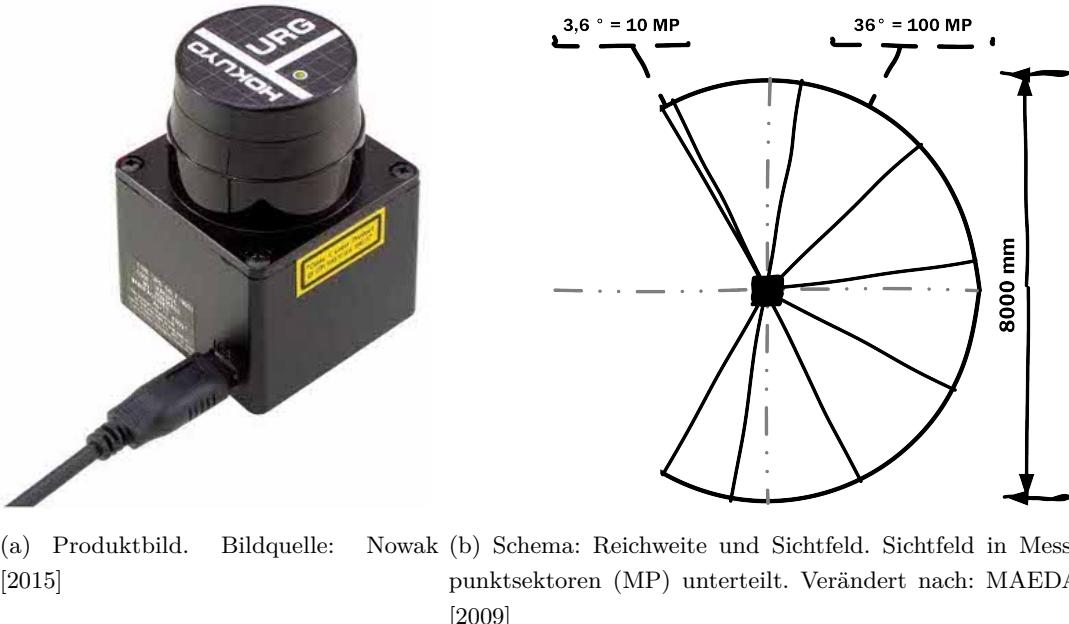


Abbildung 14: Hokuyo URG-04LX-UG01

#### 4.3.3 Argos 3D - P100

Ein weiterer Tiefensensor ist die Argos 3D – P100 von Bluetechnix. Diese arbeitet mit dem Time of Flight (ToF) Prinzip. Der Sensor wird mit einem USB-Kabel angesteuert, benötigt aber noch eine weitere Stromzufuhr. Der Sensor kann aus ROS mit einem eigenen Node angesprochen werden. Der Sensor hat eine Reichweite zwischen 100 mm und 3000mm. Die Framerate beträgt bis zu 160 FPS bei einer Auflösung von 160 x120. Das Sichtfeld beträgt 90 °.[Bluetechnix, 2015]



Abbildung 15: Argos 3D - P100. Bildquelle: Bluetechix [2015]

#### 4.3.4 Netzwerk- und Sensoranbindung

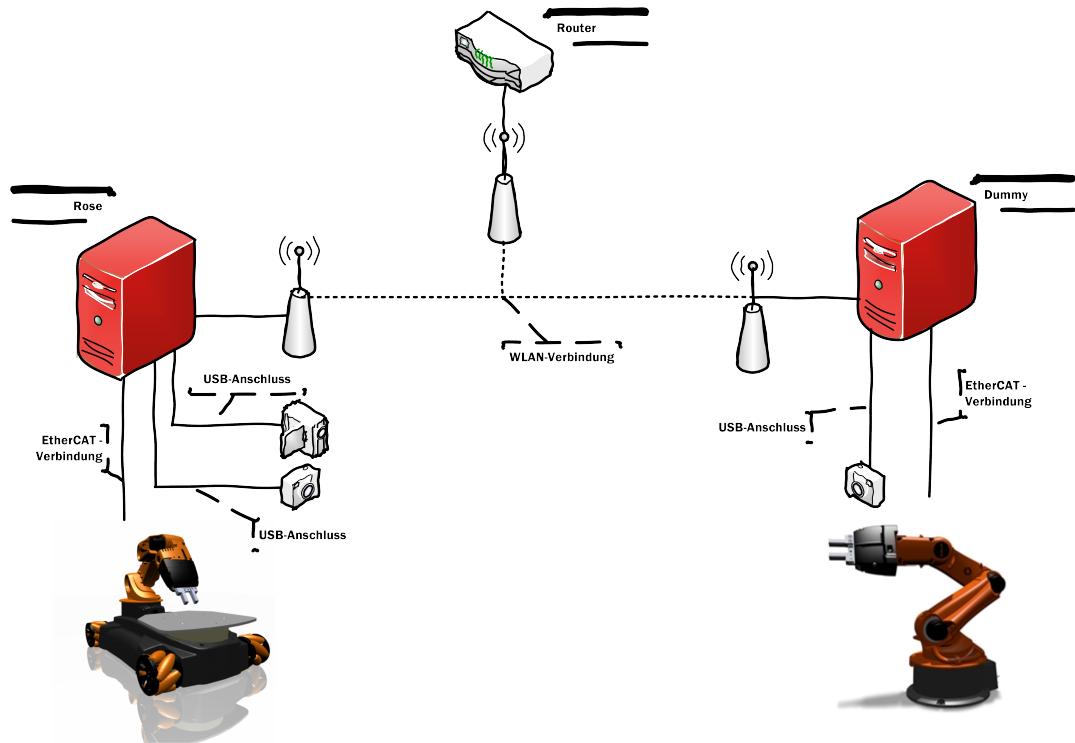


Abbildung 16: Schematische Darstellung der Verbindungen

Abbildung 16 stellt den genutzten Verbindungsplan dar. Die zentrale Komponente ist der Router, welcher mit einem externen Access-Point (AP) eine WLAN-Verbindung ermöglicht. Mit diesem WLAN verbinden sich die beiden Steuerungsrechner *Rose* und *Dummy*, welche beide unter diesen Hostnames im Netz erreichbar sind. Die Verbindungen auf beiden Seiten sind gleich gehalten. Die Verbindung zwischen Rechner und Roboter sind mit der EtherCAT-Schnittstelle (Ethernet for Control Automation Technology) realisiert. Diese Ethernet-Variante wurde für Echtzeit-Anforderungen entwickelt, indem Wert auf kurze Zykluszeiten ( $\leq 100 \mu\text{s}$ ) und niedrigem Jitter für exakte Synchronisierung ( $\leq 1 \mu\text{s}$ ) gelegt wurde. und Holger Büttner [2003] Die Verbindung zwischen den Rechnern und den Sensoren ist mit USB-Verbindungen umgesetzt. Dabei werden

alle USB-Anschlüsse auf der mobilen Plattform von Rose vollständig durch die Sensoren, sowie dem WLAN-USB-Dongle und dem Tastatur/Maus-Dongle belegt. Weitere USB-Geräte sind nur durch einen zusätzlichen HUB möglich.

Eine Alternative zu diesem Plan ist ein vermaschtes Netz, wie bei PEIS gefordet. Dadurch könnte der Router wegfallen, da alle Endgeräte direkt miteinander verbunden sind. Des Weiteren könnten einige Sensoren auf eigene Rechner ausgegliedert werden. Dies würde die einzelnen Rechner, besonders den leistungsschwachen Rechner von Rose, entlasten, aber auch die Netzwerkkomplexität und die Kosten steigern.

## **5 Zusammenfassung und Ausblick**

Hier eine Zusammenfassung und der Ausblick

## **6 Danksagung**

Ich bin allen Leuten furchtbar dankbar, vor allem den Jungs bei Google.

## 7 Literaturverzeichnis

### Literatur

William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *ACM SIGOPS Operating Systems Review*, volume 33, pages 186–201. ACM, 1999.

Ronald C Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 264–271. IEEE, 1987.

Damián Arregui, Christer Fernstrom, François Pacull, and J Gilbert. Stitch: Middleware for ubiquitous applications. In *Proc. of the Smart Object Conf*, 2003.

Asus. Xtion pro live, dec 2015. URL [https://www.asus.com/de/3D-Sensor/Xtion\\_PRO\\_LIVE/](https://www.asus.com/de/3D-Sensor/Xtion_PRO_LIVE/).

B. Badura. *Fehlzeiten-Report 2004 Gesundheitsmanagement in Krankenhäusern und Pflegeeinrichtungen*. Badura, Prof. Dr. Bernhard Schellschmidt, Dr. Henner Vetter, Christian, 2005. ISBN 978-3-540-27051-5.

Tucker Balch. The impact of diversity on performance in multi-robot foraging. In *Proceedings of the third annual conference on Autonomous Agents*, pages 92–99. ACM, 1999.

Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *ICRA*, pages 348–353. Citeseer, 2000.

Bluetchnix. Depth sensing - bluetchnix, dec 2015. URL <http://www.bluetchnix.com/de/products/depthsensing/product/argos3d-p100/>.

Guido Boella. Norms and cooperation: Two sides of social rationality. 2002.

Sylvia C Botelho and Rachid Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1234–1239. IEEE, 1999.

Mathias Broxvall and Alessandro Saffiotti. Peis ecologies: Ambient intelligence meets autonomous robotics. In *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, sOCEUSAI '05, pages 277–281, New York, NY, USA, 2005. ACM. ISBN 1-59593-304-2. doi: 10.1145/1107548.1107615. URL <http://doi.acm.org/10.1145/1107548.1107615>.

D Alonso Caceres, H Martinez, MA Zamora, and LM Tomás. A real-time framework for robotics software. In *Int Conf on Computer Integrated Manufacturing*, 2003.

Philippe Caloud, Wonyun Choi, Jean-Claude Latombe, Claude Le Pape, and Mark Yim. Indoor automation with many mobile robots. In *Intelligent Robots and Systems' 90.'Towards a New*

*Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on*, pages 67–72. IEEE, 1990.

Peter Corke. *Robotics, Vision and Control - Fundamental Algorithms in MATLAB*. Springer Science & Business Media, Berlin Heidelberg, 1st ed. 2011 edition, 2011. ISBN 978-3-642-20143-1.

Randall Davis and Reid G Smith. Negotiation as a metaphor for distributed problem solving. In *Communication in Multiagent Systems*, pages 51–97. Springer, 2003.

M Bernadine Dias and Anthony Stentz. A comparative study between centralized, market-based, and behavioral multirobot coordination approaches. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2279–2284. IEEE, 2003.

M Bernadine Dias and Anthony Stentz. A market approach to multirobot coordination. Technical report, DTIC Document, 2000.

Edmund H Durfee, Victor R Lesser, and Daniel D Corkill. Coherent cooperation among communicating problem solvers. *Computers, IEEE Transactions on*, 100(11):1275–1291, 1987.

Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Multirobot systems: a classification focused on coordination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5):2015–2028, 2004.

Monika Florek-Jasinska. Youbot detailed specifications, dec 2015. URL [http://www.youbot-store.com/wiki/index.php/YouBot\\_Detailed\\_Specifications](http://www.youbot-store.com/wiki/index.php/YouBot_Detailed_Specifications).

Vanessa Frias-Martinez, Elizabeth Sklar, and Simon Parsons. Exploring auction mechanisms for role assignment in teams of autonomous robots. In *RoboCup 2004: Robot Soccer World Cup VIII*, pages 532–539. Springer, 2005.

David Gale. *The theory of linear economic models*. University of Chicago press, 1989.

Brian P Gerkey and Maja J Matarić. Sold!: Auction methods for multirobot coordination. *Robotics and Automation, IEEE Transactions on*, 18(5):758–768, 2002.

Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

Adam T Hayes and Parsa Dormiani-Tabatabaei. Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 4, pages 3900–3905. IEEE, 2002.

John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 146–159. ACM, 2001.

Nidhi Kalra, Tony Stentz, and Dave Ferguson. Hoplites: A market framework for complex tight coordination in multi-agent teams. Technical report, DTIC Document, 2004.

Nidhi Kalra, Dave Ferguson, and Anthony Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1170–1177. IEEE, 2005.

Hisato Kobayashi. *Technolife Series. Robotto wa tomodachi da! [Technolife-Serie. Der Roboter ist ein Freund!]*. Ohmsha, Tokyo, 1999.

Kuka. Mounting and sensor plate, dec 2015. URL <http://www.youbot-store.com/accessories/youbot-extensions/mounting-and-sensor-plate>.

Hoong Chuin Lau and Lei Zhang. Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pages 346–350. IEEE, 2003.

Robert Lundh, Lars Karlsson, and Alessandro Saffiotti. Can emil help pippi. In *Proc. of the ICRA-05 Workshop on Cooperative Robotics*, 2005.

Robert Lundh, Lars Karlsson, and Alessandro Saffiotti. Plan-based configuration of a group of robots. 2006.

MAEDA. *Scanning Laser Range Finder URG-04LX-UG01 (Simple-URG) Specifications*. Hokuyo, aug 2009.

Dr. Sibylle Meyer. *Mein Freund der Roboter*. BMBF/VDE Innovationspartnerschaft AAL, oct 2011. ISBN-10: 3800733420.

Nowak. Hokuyo urg-04lx-ug01, dec 2015. URL [http://www.youbot-store.com/wiki/index.php/Hokuyo\\_URG-04LX-UG01](http://www.youbot-store.com/wiki/index.php/Hokuyo_URG-04LX-UG01).

Panasonic, may 2005. URL <http://panasonic.com.jp/museum/smilenium/kaden/robot/38/01.html>.

Lynne E Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on*, 14(2):220–240, 1998.

Lynne E Parker. Current research in multirobot systems. *Artificial Life and Robotics*, 7(1-2):1–5, 2003a.

Lynne E Parker. The effect of heterogeneity in teams of 100+ mobile robots. *Multi-Robot Systems*, 2:205–215, 2003b.

Lynne E Parker, Balajee Kannan, Fang Tang, and Michael Bailey. Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 1016–1022. IEEE, 2004.

Lynne E Parker, Maureen Chandra, and Fang Tang. Enabling autonomous sensor-sharing for tightly-coupled cooperative tasks. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pages 119–130. Springer, 2005.

David V Pynadath and Milind Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):71–100, 2003.

Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

Alessandro Saffiotti, Nina B Zumel, and Enrique H Ruspini. Multi-robot team coordination using desirabilities. In *Proc of the 6th Intl Conf on Inteligent Autonomous Systems*, pages 107–114, 2000.

Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1):165–200, 1998.

Frank Siegemund. A context-aware communication platform for smart objects. In *Pervasive Computing*, pages 69–86. Springer, 2004.

Reid Simmons, Sanjiv Singh, David Hershberger, Josue Ramos, and Trey Smith. First results in the coordination of heterogeneous robots for large-scale assembly. In *Experimental Robotics VII*, pages 323–332. Springer, 2001.

Reid Simmons, Trey Smith, M Bernardine Dias, Dani Goldberg, David Hershberger, Anthony Stentz, and Robert Zlot. A layered architecture for coordination of mobile robots. In *Multi-robot systems: from swarms to intelligent automata*, pages 103–112. Springer, 2002.

Daniela Steidl, 2011. URL [https://wwwcg.in.tum.de/fileadmin/user\\_upload/Lehrstuehle/Lehrstuhl\\_XV/Teaching/SS11/Proseminar-PIXAR/Daniela\\_Steidl\\_talk.pdf](https://wwwcg.in.tum.de/fileadmin/user_upload/Lehrstuehle/Lehrstuhl_XV/Teaching/SS11/Proseminar-PIXAR/Daniela_Steidl_talk.pdf).

Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.

Ingo J Timm and Peer-Oliver Woelk. Ontology-based capability management for distributed problem solving in the manufacturing domain. In *Multiagent System Technologies*, pages 168–179. Springer, 2003.

Dr.-Ing. Dirk Janssen und Holger Büttner. Ethercat – der ethernet-feldbus. Technical Report 23, Elektronik, 2003.

Douglas Vail and Manuela Veloso. Multi-robot dynamic role assignment and coordination through shared potential fields. *Multi-robot systems*, pages 87–98, 2003.

Richard T Vaughan, Kasper Støy, Gaurav S Sukhatme, and Maja J Matarić. Whistling in the dark: cooperative trail following in uncertain localization space. In *Proceedings of the fourth international conference on Autonomous agents*, pages 187–194. ACM, 2000.

Lovekesh Vig and Julie A Adams. Issues in multi-robot coalition formation. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pages 15–26. Springer, 2005.

Cosima Wagner. „Tele-Altenpflege“ und „Robotertherapie“: Leben mit Robotern als Vision und Realität für die alternde Gesellschaft Japans, 2009.

Barry Brian Werger and Maja J Matarić. Broadcast of local eligibility for multi-target observation. In *Distributed autonomous robotic systems 4*, pages 347–356. Springer, 2000.

Bruno Zandonella. Bevölkerungsentwicklung und Renten. oct 2013.

Robert Zlot and Anthony Stentz. Complex task allocation for multiple robots. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1515–1522. IEEE, 2005.

## **A Erster Anhang**

blah blah blah

## **B Noch ein Anhang**

spannende Sache, oder?