Entwicklung eines modularen Gestensteuerungssystems

Von der Fakultät für Maschinenbau
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation

Von
Dipl.-Ing. Habib Nasri
geboren am 20.07.1982
in Sidi Bouzid (Tunesien)

2015

Tag der mündlichen Prüfung: 06.03.2015

Erster Gutachter: Prof. Dr.-Ing. Roland Lachmayer

Zweiter Gutachter: Prof. Dr.-Ing. Ludger Overmeyer

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Produktentwicklung und Gerätebau (IPeG) der Leibniz Universität Hannover.

Herrn Prof. Dr.-Ing. Roland Lachmayer, dem Leiter des Instituts für Produktentwicklung und Gerätebau, gilt mein besonderer Dank für die vertrauensvolle und wohlwollende Unterstützung, die ich während meiner Tätigkeit am Institut erfahren habe.

Herrn Prof. Dr.-Ing. Ludger Overmeyer, dem Leiter des Instituts für Transport- und Automatisierungstechnik (ITA) danke ich für die Übernahme des Koreferates und das fachliche Interesse an der Arbeit.

Frau Prof. Dr.-Ing. Annika Raatz, der Leiterin des Instituts für Montagetechnik (match) danke ich für die Übernahme des Prüfungsvorsitzes.

Ich danke meinen Kollegen am Fachgebiet, Diplom- und Studienarbeitern sowie studentischen Hilfskräften, die einen Beitrag zur Realisierung dieser Arbeit geleistet haben.

Hannover, den 01.12 2014 Habib Nasri

Kurzfassung

Nasri, Habib

Entwicklung eines modularen Gestensteuerungssystems

Schlagworte: Gestensteuerung, Bewegungserfassung, Echtzeit, Modularität

In Zeiten der Entwicklung hin zu multimedialen Inhalten, sowie der Simulation und Visualisierung von modernen Produkten, gewinnt die Mensch-Maschine-Interaktion immer mehr an Bedeutung. Humanoid- und Teilhumanoid-Roboter werden allmählich Bestandteil des täglichen Lebens. Die Steuerung und Interaktion mit komplexen Humanoid-Robotern erfordert eine hohe Rechenleistung sowie komplexe Algorithmen zur Erzeugung einer Art künstlicher Intelligenz. Eine Alternative zur Steuerung von Humanoid-Robotern mittels künstlicher Intelligenz ist die Gestensteuerung, d.h. die Bewegung sowie die Intelligenz werden vom Menschen und nicht von der Maschine vorgegeben. In diesem Projekt wurde ein modulares Gestensteuerungssystem zur Steuerung eines Humanoid-Roboters entwickelt. Durch die hohe Flexibilität sowie die intuitive Bedienbarkeit des System kann der Roboter auch von fachfremden Benutzern bedient und für Aufgaben eingesetzt werden, welche mit normalen Industrierobotern nicht zu bewältigen sind.

Abstract

Nasri, Habib

Development of a modular gesture control system

Keywords: Gesture control, motion detection, real-time, modularity

In times of the increasing trend towards multimedia contents, the simulation and visualization of modern Products, human-machine interaction becomes more important. Humanoid and part-humanoid robots become gradually part of our daily life. The control and interaction with complex humanoid robots requires a high computing power and complex algorithms to produce a type of artificial intelligence. An alternative for the control of humanoid robots with artificial intelligence is the gesture control. Thereby the movement and intelligence are dictated by the human operator and not by the machine. In this project, a modular gesture control system for controlling a humanoid robot was developed. Due to the high flexibility and the intuitive operation the robot can be controlled by non-specialists and can be used for tasks which cannot be managed by normal industrial robots.

Inhaltverzeichnis

Abkürzungen

CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
EEPROM	Electrically Erasable Programmable Read-Only Memory
EES	Echtzeit Embedded System
GPGPU	General Purpose Computation on Graphics Processing Unit
GPU	Graphics Processing Unit
GCC	GNU Compiler Collection
I2C	Inter-Integrated Circuit
IC	Integrierter Schaltkreis (Integrated Circuit)
ICP	Iterative Closest Point Algorithm
JTAG	Joint Test Action Group
MIPS	Million Instructions Per Second
NUI	Natural User Interface
PWM	Pulsweitenmodulation
SPI	Serial Peripheral Interface
SLAM	Simultaneous Localization and Mapping
TTL	Transistor-Transistor Logik
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
RoKi	Roboter gesteuert mit Kinect
FPS	Frames per Second

Formelverzeichnis

Abschnitt 2.3

θ	Rotationswinkel von x_0-Achse auf x_1-Achse
α	Rotationswinkel von Gelenkachse z_0 um die x_1-Achse
d	Verschiebung des Koordinatensystem entlang der Gelenkachse z_0
a	seitlicher Abstand zum nächsten Gelenk

Abschnitt 2.7

$R(u)$	Bildelement
u	Bildpunkt
d	Distanz
D_s	Distanz zur Projektionsfläche
q	Quaternion
G	Gaußfilter
g	Glättungsparameter
GK	Globale Koordinatensystem
K	Intrinsische Matrix
x, y, z	Koordinaten im R^3
N	Nachbarschaft
$R(q)$	Nachbarschaftselement
W, C	Normalisierungsterm
R	Rotationsmatrix
σ	Standardabweichung

Abschnitt 4.1.1

X_n	Eingabeposition
X_{med}	Median der letzten Eingangspositionen
t	Schwellenwert
b_n	geglättete Eingabeposition
γ	Glättungsparameter
$\widehat{X}_n$	Trend der Eingabeposition
α	Konvertierungsparameter
φ_{basis}	Winkel zwischen dem Koordinatenvektor z und der Kameraachse
φ_i	Gelenkwinkel

Abschnitt 4.2

G	Raumbegrenzung der Gelenkwinkel
α	Gelenkwinkel für Raumbegrenzung
$\emptyset_{i,min}$	minimale Gelenkwinkelgrenze
$\emptyset_{i,max}$	maximale Gelenkwinkelgrenze
$\widehat{G}$	minimale Anzahl der aktiven Grenzen
L	Gelenkwinkellimit
$\Pi_{i,min}$	minimale Gelenkwinkellimit
$\Pi_{i,max}$	maximale Gelenkwinkellimit
φ_i	Gelenkwinkel für Gelenkwinkellimit
w	Geschwindigkeit
w_{max}	maximale Geschwindigkeit

a_{acc}	Beschleunigung
a_{dec}	Verzögerung
φ_i	Zielwinkel
γ_i	aktuelle Gelenkposition
ε_i	kürzester Weg zur Zielposition
$\hat{\varepsilon}_i$	Trend des kürzesten Weges
d_i	Drehrichtung
Ω_i	Geschwindigkeit nach der Zeit t
Γ_i	erreichte Position nach der Zeit t
w_{dec}	erlaubte Geschwindigkeit
f_i	Schrittfrequenz des Schrittmotors

Abschnitt 4.4.3

a	erste Speicherkomponente des Eingangswerts
b	zweite Speicherkomponente des Eingangswerts
X	Binärzahl
A	Binärzahl der ersten Speicherkomponente des Eingangswerts
B	Binärzahl der zweiten Speicherkomponente des Eingangswerts
N	1-basierte Nummer des höchstwertigen 1-Bit
T	Abgerundeter Geschwindigkeitswert
p	Periodendauer

Abschnitt 4.5.2.2

P	Projektionsmatrix
n	direkte Abstand vom Koordinatenursprung zur Nahbene

f	Abstand zur Fernebene
l	x-Koordinaten der linken vertikalen Kante der Nahbene
r	x-Koordinaten der rechten vertikalen Kante der Nahbene
b	y-Koordinaten der linken vertikalen Kante der Nahbene
t	y-Koordinaten der rechten vertikalen Kante der Nahbene

1 Einleitung

Roboter sind aus vielen Anwendungsfeldern in der heutigen Zeit kaum noch wegzudenken. War ihr Einsatz zunächst stark an das industrielle Umfeld gebunden, finden diese mittlerweile auch Einzug in die Unterhaltungsbranche und haushaltnahe Unterstützungsdienste.

Der Einsatzzweck bestimmt dabei Form und Ausprägung des Roboters, Art und Anzahl der benötigten Gelenke und Freiheitsgrade sowie die Art der Steuerung. Bei Robotern, die in der Fertigung eingesetzt werden, liegt der Fokus auf der Optimierung und Anpassung an bestimmte Bewegungsabläufe, die über einen langen Zeitraum mit gleicher Wiederholgenauigkeit ausgeführt werden sollen. Insbesondere das Thema der Energierückgewinnung und Ressourcennachhaltigkeit steht hier im aktuellen Forschungsinteresse [Web09] [Hei07].

Roboter für die Unterhaltungsindustrie oder für haushaltsnahe Dienste werden dem gegenüber mit umfangreichen Interaktionsmöglichkeiten ausgestattet, so dass veränderten Umweltbedingungen entgegengewirkt und auf Befehle des Nutzers reagiert werden kann. Der Trend geht in diesem Bereich in Richtung menschenähnlicher Roboter, genannt Humanoid-Roboter. Ein Merkmal dieser sind die beiden an einem Torso angebrachten Manipulatorarme. Die Steuerung solcher Roboter erfordert in der Regel eine hohe Rechenleistung, sowie komplexe Algorithmen zur Implementierung einer künstlichen Intelligenz, welche die Interaktion mit der Umwelt innerhalb von festgelegten Parametern erlaubt [Toy12] [Hon12].

Die Verbindung aus beidem eröffnet neue Einsatzbereiche, z.B. Montage, Teileübergabe, Maschinenbeschickung, Verpackung und andere Handlingaufgaben, die derzeit eher von Menschen ausgeführt werden. Eine der Herausforderungen hierbei ist die Steuerung der Roboter selbst. Ist eine Aufgabe in einer gegebenen Konfiguration nur wenige Male auszuführen, sind lange Einfahrzeiten oder aufwendige Programmierung nicht wirtschaftlich.

Eine Alternative zu traditionellen Programmier- und Steuerungstechniken stellt die Gestensteuerung dar. Hierbei wird durch ein Sensorsystem die Bewegung eines Operators detektiert und über einen Prozessrechner als Steuerungsbefehle an die Gelenkantriebe weitergegeben. Die Reaktion auf die Umwelt wird dabei dann nicht mehr durch die Maschine kontrolliert, sondern liegt prinzipiell in der Hand des Bedieners. Dies wiederum führt dazu, dass für die Aufrechterhaltung der Betriebssicherheit bestimmte Werte der Umwelt der Maschine bekannt sein müssen, z.B. Räume, die nicht durch den Roboter durchfahren werden dürfen, weiterhin muss sie über Sensoren verfügen, die im Fall einer Kollision einen Abbruch der

Bewegung auslösen. Darüber hinaus ist eine Echtzeitsteuerung für ein natürliches Bewegungs- und Steuerungsverhalten wünschenswert.

Je nach Art der Bewegungserfassung ist eine umfangreiche Nachbearbeitung und Transformation der Bewegungsdaten des Bedieners durchzuführen. Dies liegt unter anderem darin begründet, dass der menschliche Arm über sieben Gelenke verfügt (und damit kinematisch redundant ist), die Roboterkinematik eventuell jedoch weniger Freiheitsgrade hat [Den55] [Pau84].

Eine Gestensteuerung ermöglicht prinzipiell einen weiteren Anwendungsfall, da die Bewegungserfassung räumlich nicht an den Arbeitsort des Roboters gebunden ist, mit anderen Worten, die Maschine ist über Teleoperation steuerbar. Dies führt zu der Anforderung, dass dem Benutzer ein Feedback über die aktuelle Arbeitsumgebung des Roboters zugänglich gemacht werden muss.

Am Institut für Produktentwicklung und Gerätebau (IPeG) wurde im Rahmen dieser Arbeit das Humanoid-Robotersystem mit dem Namen ROKI entwickelt. Zielsetzung des ROKI-Projektes ist die Entwicklung eines modularen Gestensteuerungssystems zur Steuerung von Humanoid- und Teilhumanoid-Robotern mit ökonomisch vertretbarem Aufwand.

An den Demonstrator werden dabei folgende Anforderungen gestellt:

- Gestensteuerung: Diese soll auf Basis von kostengünstigen Sensoren realisierbar sein. Hierfür ist das Kinect-Kamera-System von Microsoft als bildbasierte Gestensteuerung auf seine Leistungsfähigkeit hin zu überprüfen.
- Intuitivität: Die Gestaltung der Benutzerschnittstelle ist intuitiv auszuführen, was auch bedeutet, dass das System in der Lage sein muss, die Sensordaten zu interpretieren und ggf. zu filtern um Fehler auszugleichen oder auf besondere Kommandos zu reagieren.
- Echtzeitumgebung: Um ein flüssiges Arbeiten zu ermöglichen müssen die Steuerungsbefehle in Echtzeit an die Antriebe der Maschine übergeben werden. Im Gegenzug müssen die aktuellen Gelenkstellungen und die Position des Roboters wieder zurückgemeldet werden. Dazu ist ggf. eine entsprechende Hardware zu konzipieren und umzusetzen.

- Modularität: Die Systemkomponenten sollen weitestgehend unabhängig voneinander entwickelt werden können. Ein Upgrade des späteren Systems soll durch den Austausch von Komponenten jederzeit möglich sein.
- Skalierbarkeit: Das System soll unabhängig von der Anzahl der Gelenke des Roboters funktionieren. Das bedeutet, dass die Gelenkstellungen des Benutzers auf die Gelenkanzahl der Maschine ggf. automatisch herunterskaliert werden müssen.
- Umwelt-Feedback: Der Roboter muss dazu in der Lage sein, die für die Bedienung in Teleoperation nötigen Daten der Umwelt zu detektieren und über eine Schnittstelle darzustellen, so dass der Benutzer ausreichend Informationen für Entscheidungsprozesse zur Verfügung gestellt bekommt.

Da der Fokus auf der Steuerung selbst liegt, werden für den Demonstrator zunächst keine besonderen Anforderungen an die Steifigkeit des Roboters, die zu erreichende Absolutgenauigkeit oder die zu hebenden Lasten gestellt.

Aufbau und Inhalt der Arbeit

Im folgenden Kapitel zwei „Stand des Wissens“ werden zunächst die Grundsätze der Steuerung von Maschinen erläutert. Dann folgen eine Begriffsklärung zu Roboterarten, Roboterkinematik, sowie eine Übersicht der bekanntesten existierenden Humanoid-Roboter, gefolgt von einer Zusammenfassung der Grundlagen der Bewegungserfassung. Daran schließt sich eine Erläuterung des Aufbaus und der Funktion des Kinect-Systems von Microsoft an, sowie Erläuterungen für die Bildbearbeitung zur Verfügung stehenden, CUDA-Technologie und Grundlagen der simultanen Lokalisierung und Kartographierung. Den Abschluss des Kapitels bilden die Definitionen von Echtzeitsystemen und Modularität.

Kapitel drei beginnt mit der Spezifikation des Systems, gefolgt von der Beschreibung der Architektur des modularen Gestensteuerungssystems. Abschließend werden die wichtigsten Module des Systems vorgestellt.

Kapitel vier befasst sich mit der Software. Zuerst steht eine detaillierte Beschreibung des Algorithmus der Bewegungserfassung und Winkelberechnung. Danach folgt die mathematische Beschreibung der Rückführung der Gelenkwinkel und anschließend die Regelung und Ansteuerung der Schrittmotoren. Im Anschluss werden das Kommunikationsprotokoll sowie die Firmware der Controller beschrieben. Daran anknüpfend erfolgt die Beschreibung der PC-Software sowie der Benutzeroberfläche und zum Ende des Kapitels wird das Verfahren

zur 3D Rekonstruktion der Roboterumgebung in den 3D virtuellen Raum, sowie seine Umsetzung beschrieben.

Das fünfte Kapitel verschafft einen Einblick in die entwickelte Hardware, begonnen mit dem Steuerungsmodul und seinen Komponenten. Ebenfalls wird das Design und die Mechanik des im Rahmen dieser Arbeit als Demonstrator entwickelten Humanoid-Roboters RoKi vorgestellt.

Kapitel sechs befasst sich mit den Testergebnissen und der Validierung des gesamten Systems in Bezug auf Positioniergenauigkeit, Wiederholgenauigkeit und Hubkraft.

Das siebte Kapitel beschreibt den Transfer in eine reale Praxisanwendung „Gestengesteuerter Tauchroboter“, sowie den Aufbau der ersten Tests und die Validierung.

Im Kapitel acht wird die durchgeführte Arbeit zusammenfassend beschrieben gefolgt von einem Ausblick über die Möglichkeiten für eine Weiterentwicklung des Systems

.

2 Stand des Wissens

2.1 Maschinensteuerung

Die Steuerung einer Maschine hängt stark von der Art der Maschine, dem Automatisierungsgrad sowie der auszuführenden Aufgabe ab. Man unterscheidet zwischen Direkt-, Indirekt-, Manuell-, Vollautomatik- und Überwachungssteuerung. Ein Kriterium hierfür ist die Datenmenge, die zwischen dem Bediener und der Maschine ausgetauscht wird.

Direkte und indirekte Steuerung

Bei der Steuerung von Maschinen wird zwischen indirekter und direkter Steuerung unterschieden [Hes11]. Ein klassisches Beispiel für indirekte Steuerung ist die Erkennung von Ausschuss durch den Vergleich eines Kamerabildes mit dem Bild eines Gutteils. Beim Erkennen eines Schlechtteils, wird dieses durch eine Maschine entfernt. Anhand des Kamerabildes wird nur entschieden, ob ein Teil ausgestoßen werden soll, oder nicht, der Ausstoßvorgang selbst bleibt unbeeinflusst [Hes11]. Ebenso ist das Takten eines Prozesses mittels einer Lichtschranke, die das Vorhandensein eines Rohteils anzeigt, eine Form der indirekten Steuerung.

Wird jedoch beispielweise auch die Länge eines Rohteils erfasst und dementsprechend die Verfahrwege der Maschine während der Bearbeitung angepasst, spricht man von direkter Steuerung [Bai04]. Mit dieser Form der Steuerung ist es ebenfalls möglich, dass ein System automatisch auf Hindernisse reagiert und diese umgeht.

Eine Weiterentwicklung der direkten Steuerung ist die unmittelbare Steuerung einer Maschine ohne einen zugrunden liegenden Bewegungsablauf [Cor94]. Hierzu zählt zum Beispiel, wenn ein Roboter die Bewegungen eines menschlichen Bedieners kopiert, um Arbeiten in einem für Menschen nicht zugänglichen Raum durchzuführen.

Manuelle Steuerung

Bei der manuellen Steuerung hat der Bediener die volle Kontrolle über die Maschine. Er betätigt die Bedienelemente und ist verantwortlich für das gesamte Verhalten der Maschine. Die momentanen Bearbeitungszustände können direkt durch Beobachtung oder mittels Sensorik und Anzeigen verfolgt werden [Cor94] [Hes11].

Vollautomatische Steuerung

Hier findet die Aufgabe ohne jeglichen Eingriff des Bedieners statt. Die Maschine reagiert selbständig und handelt nach ihrer eigenen Programmierung. Der Ablauf des Systems wird von außen nicht beeinflusst. Das bedeutet jedoch auch, dass die Maschine nicht selbstständig auf Ereignisse reagieren kann, die nicht in der Programmierung definiert worden sind [Pau84].

Überwachungssteuerung

Die Aufgabe wird selbständig von der Maschine wie beim Automatikbetrieb erledigt. Der Operator wird jedoch indirekt einbezogen, indem er anhand von Prozessinformationen die Vorgänge überwacht, beispielweise durch das Ablesen der Anzeigen. In Konflikt- oder Störfällen kann so sofort eingegriffen werden [Hes11] [Pau84].

2.2 Robotik

Ein Roboter ist ein komplexes mechatronisches System, das Technologie aus dem Maschinenbau, der Elektrotechnik, Softwaretechnik und Regelungstechnik enthält [Bru07].

2.2.1 Definitionen

Der Begriff „*Roboter*“ stammt aus dem tschechischen Wort „*Robota*“ was die Bedeutung von „Arbeit“ hat. Die Bezeichnung „*Robot*“ wurde von dem tschechischen Schriftstellern Karl Capek im Jahr 1920 in seinem Theaterstück „Rossum Universal Robot“ verwendet.

Definition nach VDI-Richtlinie 2860

„[. . .], universell einsetzbarer Bewegungsautomat mit mehreren Achsen, dessen Bewegungen hinsichtlich Folge und Wegen bzw. Winkeln frei programmierbar und ggf. sensorgefühlt sind“ [VDI2860].

Definition nach Robotic Industries Association

„Ein Roboter ist ein programmierbares Mehrzweck-Handhabungsgerät für das Bewegen von Material, Werkstücken, Werkzeugen oder Spezialgeräten. Der frei programmierbare Bewegungsablauf macht ihn für verschiedenste Aufgaben einsetzbar.“ [Rob10].

Definition nach Japan Robot Association (JARA)

Die JARA unterscheidet und klassifiziert Roboter anhand der folgenden Systematik:

- *Manual Manipulator:* ist ein Handhabungsgerät ohne eine Programmierung und wird direkt vom Bediener geführt [Jar72] [Arn10].
- *Fixed Sequence Robot*: ein Roboter der seiner Arbeit nach einem konstanten Bewegungsmuster wiederholt. Eine Änderung dieses Bewegungsmusters ist relativ aufwendig [Jar72] [Arn10].
- *Variable Sequence Robot*: ist ein Roboter mit ähnliche Eigenschaften wie der Fixed Sequence Robot, die Neuprogrammierung seiner Bewegungsmuster ist jedoch wesentlich einfache [Jar72] [Arn10].
- *Playback Robot*: Der Bewegungsablauf dieses Roboters wird durch den Bediener vorgeführt und im Programmspeicher gespeichert. Mit dem gespeicherten Bewegungsmuter kann die Arbeit beliebig oft wiederholt werden [Jar72] [Arn10].
- *Numerical Control Robot*: Diese Roboter hat eine ähnliche Steuerung wie eine NC-Maschine: Die Bewegungsabläufe werden über Taster, Schalter oder Datenträger eingegeben [Jar72] [Arn10].
- *Intelligent Robot*: ist ein Roboter der über verschiedene Sensoren verfügt und in der Lage ist, seine Bewegungsabläufe selbsttätig zu bestimmen und damit auf Grundlage vom Veränderungen des Werkstücks und der Umwelt zu reagieren [Jar72]. [Arn10].

2.2.2 Roboter Kinematik

Eine zentrale Aufgabe bei der Steuerung von Robotern ist die Einstellung der Gelenkwinkel des Roboters anhand einer gegebenen Endeffektorposition im Raum. Man spricht in diesem Zusammenhang auch von der Transformation zwischen Jointspace und Taskspace [Pfe87]. Hierbei wird die Bestimmung der Gelenkstellungen aus den kartesischen Koordinaten Vorwärtstransformation oder auch direkte Kinematik genannt. Die Umkehrung stellt die Rückwärtstransformation oder inverse Kinematik dar [Pfe87].

Vorwärtstransformation

Die Vorwärtstransformation wird für die Berechnung der kartesischen Koordinaten aller Gelenke bei bekannten Gelenkstellungen verwendet. Für sog. serielle Kinematiken (die Vorschubbewegungen der einzelnen Achsen addieren sich auf, dabei ist jede Achse für eine separate Vorschubbewegung verantwortlich) ist dies ein rechentechnisch einfaches Problem. Die allgemeine Vorgehensweise besteht aus dem Aufstellen von Transformationsmatrizen von einem Gelenk zum nächsten. Durch die verkettete Ausführung der einzelnen Transfor-

mationen werden die kartesischen Koordinaten von jedem beliebigen Punkt des Roboters berechnet [Den55] [Pau84].

Ein formales Verfahren zur Aufstellung der Transformationsmatrizen ist die Denavit- Hartenberg-Transformation [Den55]. Sie gibt einen Algorithmus vor, wie aus den Parametern Gelenkwinkel, Gelenkabstand, Armlänge und Verwindung für jedes Gelenk eine Matrix erstellt werden kann. Eine Denavit-Hartenberg-Transformation kann auf jede beliebige rein serielle Kinematik angewendet werden.

Bei parallelen Kinematiken (die Vorschubbewegungen beeinflussen sich gegenseitig, mehrere Achsen sind an einer Vorschubbewegung beteiligt; ein Beispiel hierfür ist der Hexapod) gibt es keine formalisierten Verfahren zur Durchführung der Vorwärtstransformation. Die möglichen Stellungen einer Parallelkinematik lassen sich durch ein Gleichungssystem aus Polynomen beschreiben, dieses ist jedoch für jede kinematische Struktur unterschiedlich. Um die kartesischen Koordinaten der Gelenke zu berechnen, muss das Gleichungssystem gelöst werden, was nur in Ausnahmefällen analytisch möglich ist. Wenn das Gleichungssystem über mehrere Lösungen verfügt, muss die richtige Lösung anhand geeigneter Nebenbedingungen (zum Beispiel minimal einzusetzende Energie) ermittelt werden [Neu04].

Rückwärtstransformation

Die Rückwärtstransformation dient zur Bestimmung der notwendigen Gelenkstellungen einer bestimmten Stellung des Roboters im Raum. Es gibt drei grundlegende Herangehensweisen [Pau84]: Wenn es für die Vorwärtstransformation analytische Lösungen gibt, werden zur Berechnung die entsprechenden Transformationsmatrizen invertiert und das so entstehende Gleichungssystem gelöst. Dabei gibt es oft mehr als eine Lösung, so dass anhand geeigneter Rahmenbedingungen die beste Lösung ausgewählt werden muss. Dies ist z. B. bei seriellen Kinematiken unter Einsatz der Denavit-Hartenberg-Transformation generell gegeben.

Außerdem können die Gelenkstellungen direkt aus der Geometrie des Roboters abgeleitet werden. Gerade bei Robotern mit nur wenigen Gelenken kann so schnell eine optimierte Gleichung gefunden werden.

Wenn die analytische Berechnung nicht möglich ist, was bei Parallelkinematiken generell der Fall ist, erfolgt eine numerische Berechnung. Hierbei wird durch iteratives Ändern der Gelenkstellungen versucht, eine Kombination zu finden, bei der der Roboter die gewünschte Stellung einnimmt [Sch07].

2.2.3 Humanoid-Roboter

Humanoid-Roboter unterscheiden sich von den in der Produktion und Fertigung eingesetzten Industrierobotern durch ihre menschlichen Bewegung und ihr menschliches Aussehen. Während Industrieroboter dafür gebaut werden, immer wiederkehrende Aufgaben zu erledigen, haben humanoid-Roboter ein weites Aufgabenfeld. Zudem sind Industrieroboter meistens ortsgebunden. Humanoid-Roboter sind dagegen mobil, werden größtenteils in der Unterhaltungsbranche, im Service oder im der Medizintechnik genutzt und bewegen sich auf unterschiedliche Arten. Während die Unterschiede bei der Umsetzung der Bewegung der Arme und des Kopfs eher gering sind, wird die Bewegung des Unterkörpers unterschiedlich ausgeführt. Je nach Arbeitsaufgabe und Funktion des Roboters, wird dabei in einbeinige, zweibeinige, und radangetriebene Roboter unterschieden [Kim09].

Seit den 1986er Jahren entwickelt die Firma *Honda Motor Co., Ltd.* Humanoid-Roboter. *ASIMO* (Advanced Step in Innovative MObility) ist das aktuellste Projekt und ist der zurzeit am weitesten entwickelte zweibeinige Humanoid-Roboter (Abbildung 1).

Abbildung 1: ASIMO [Hon12]

Dieser verfügt über jeweils 14 Freiheitsgrade in den Armen und vier in den Händen und kann bis zu 1 kg heben. Der Roboter ist 1300 mm groß und hat einen Rucksack am Rücken, in dem er einen Li-Ionen Akku trägt. Dadurch kann der Roboter sich bis zu 1 Std. alleine fortbewegen, bis er wieder an seine Ladestation muss. Dabei kann er eine Geschwindigkeit von bis zu 6 km/h erreichen. Der Körper besteht größtenteils aus einer Aluminiumlegierung und einem Polymer, was ihn sehr leicht (54 kg) macht. ASIMO kann als autonom bezeichnet werden, da er selbstständig handeln kann. Beispielsweise kann er Menschen anhand ihrer Stimme erkennen, Hände schütteln, Objekte umgehen (vorausschauend handeln) und Treppen steigen. Der Roboter wird in der Unterhaltungsbranche und im Service eingesetzt [Hon12].

Humanoid und *Robina* (Abbildung 2) wurden von der Toyota Motor Corporation (TMC) entwickelt, um den Menschen in seiner Umgebung zu unterstützen. *Humanoid* ist ein zweibeiniger, 1522 mm großer und 56 kg schwerer Roboter, der wie ein Mensch gehen und laufen kann. Er besitzt je 17 Freiheitsgrade in Armen und Händen und kann sehr präzise gesteuert werden. Beispielsweise kann er durch seine Flexibilität in den Händen auch Violine spielen [Toy12]. Zukünftiges Ziel der Entwicklung von *Humanoid* ist die Einbindung des Roboters zur Unterstützung im Haushalt, in der Pflege und in der medizinischen Versorgung.

Abbildung 2: Humanoid (Links) und Robina (Rechts) [Toy12]

Robina ist 1200 mm groß, wiegt ca. 60 kg und besitzt insgesamt 27 Freiheitsgrade. Der radangetriebene Roboter bewegt sich mit bis zu 3,6 km/h und besitzt einen Laser-Entfernungsmesser, mit dem Objekte in der Umgebung erkannt und umfahren werden. Dadurch kann *Robina* Wege mit Echtzeitinformationen planen. Zudem besitzt der Roboter eine Drei-Finger-Hand, mit der er einen Stift halten und schreiben kann. Durch die Sprach-erkennung und –wiedergabe ist *Robina* auch in der Lage Unterhaltungen zu führen. Die Einsatzgebiete des Roboters sind daher vielseitig. Durch seine Eigenschaften soll der Roboter zukünftig den Menschen in der Krankenpflege, medizinischen Versorgung und im Haushalt unterstützen [Toy12].

Seit den 1999er Jahren entwickelt die Waseda-Universität in Tokio (Japan) Humanod-Roboter. Ein aktuelles Projekt heißt TWENDY-ONE (Abbildung 3) und ist ein 1467 mm großer und 111 kg schwerer Humanoid-Roboter, der mit einem omnidirektionalen Antrieb

ausgestattet ist. Dadurch kann der Roboter jederzeit in jede beliebige Richtung fahren. Insgesamt besitzt der Roboter 47 Freiheitsgrade und kann mit den Händen den Boden berühren um z. B. Objekte aufzuheben. Zudem kann der Roboter sich an Objekte anpassen, deren Kräfte aufnehmen und dadurch Menschen stützen und tragen. Der Roboter kann daher den Menschen sowohl im Haushalt bei alltäglichen Arbeiten als auch im Service helfen [Was12].

Abbildung 3: TWENDY-ONE [Was12]

Alle vorgestellten Roboter haben gemeinsam, dass sie sich in vorgegebenen Grenzen autonom bewegen können. Sie folgen keinen programmierten Handlungsabläufen, sondern können mit der Umwelt interagieren. Sie treffen Entscheidungen, die von den Umgebungsbedingungen abhängen. Diese Roboter sind das Ergebnis jahrzehntelanger Entwicklung und stellen den aktuellsten Stand von Humanoid-Robotern dar. Jedoch fehlt es den Robotern überwiegend noch an Präzision sowie an der Fähigkeit, menschliche Bewegungen natürlich zu imitieren.

2.3 Denavit-Hartenberg-Transformation

Für die Berechnung der direkten Kinematik sind mehrere Verfahren verbreitet. Für Roboter wird insbesondere die Denavit-Hartenberg-Transformation verwendet, da sie für serielle Kinematiken einfach und methodisch bestimmt werden kann [Pfe87].

Die Denavit-Hartenberg-Transformation ist eine Methode, um eine Matrix zu konstruieren, die eine Koordinatentransformation von einem Gelenk zum nächsten Gelenk darstellt. Zur Berechnung der kartesischen Koordinaten jedes Gelenks wird auf die Koordinaten des Ur-

sprungspunkts der kinematischen Kette nacheinander die Transformationsmatrix für jedes Gelenk angewendet.

Die Denavit-Hartenberg-Transformationsmatrix besteht aus vier verknüpften Teiltransformationen. Abbildung 4 zeigt die Parameter der Transformationen.

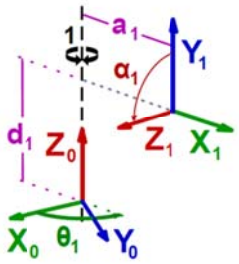


Abbildung 4: Denavit-Hartenberg-Transformation [Web09]

Die erste Transformation beschreibt die Drehung des neuen Koordinatensystems um die Gelenkachse. Der Winkel θ stellt die Rotation dar, die notwendig ist, um die x_0-Achse auf die x_1-Achse zu drehen.

$$Rot_\theta(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Die zweite Transformation beschreibt die Verschiebung entlang der Gelenkachse z_0. Der Ursprung des nächsten Gelenks ist um d verschoben.

$$Trans_d(d) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Die dritte Transformation beschreibt den seitlichen Abstand zum nächsten Gelenk. Der Abstand a wirkt entlang der x_1-Achse.

$$Trans_a(a) = \begin{bmatrix} 0 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Die letzte Transformation beschreibt die Drehung von der alten Gelenkachse z_0 zur neuen Gelenkachse z_1. Die Rotation α findet um die Achse x_1 statt.

$$Rot_{\alpha}(\alpha) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & cos\ \alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Die komplette Transformationsmatrix ergibt sich durch Multiplikation aller Teilmatrizen:

$$T(\theta, \alpha, a, d) = Rot_{\theta}(\theta) \cdot Trans_{d}(d) \cdot Trans_{a}(a) \cdot Rot_{\alpha}(\alpha) \tag{2.5}$$

Ein klassischer kartesischer Vektor besteht aus drei Komponenten. Um auf diesen die beschriebene 4×4 -Transformationsmatrix anwenden zu können, ist eine Erweiterung um eine vierte Komponente mit dem festen Wert 1 notwendig (sog. homogene Transformation) [Web09].

Die Werte der Parameter θ, α, a und d sind durch die Roboterkinematik festgelegt und können z.B. durch Messungen am realen Roboter oder aus seinem CAD-Model ermittelt werden.

2.4 Bewegungserfassung

Die Erfassung und Digitalisierung der menschlichen Bewegung wird Motion Capturing genannt. Hierbei werden verschiedene Verfahren anhand ihrer verwendeten Hardware unterschieden. Es lassen sich optische, magnetische, mechanische und bildbasierte Systeme voneinander differenzieren [Hor02] [Kre13].

2.4.1 Optische Bewegungserfassung

Die Bewegungserfassung mit optischen Verfahren ist das am weitesten verbreitete System. Die zu erfassende Person trägt hierbei einen Anzug, auf dem Marker angebracht sind. Es handelt sich entweder um passive Marker, die zum Beispiel Infrarotstrahlung reflektieren, oder aktive Marker, wie zum Beispiel Leuchtdioden. Die von den Markern emittierte Strahlung wird dann von einer oder mehreren Kameras erfasst.

Um aus den Daten auf die Bewegung und die Proportionen einer Person schließen zu können, muss die Position der Kameras im Raum exakt bekannt sein. Vorteile der optischen Verfahren sind die hohe Genauigkeit und Abtastrate sowie bei passiven Markern die Bewegungsfreiheit. Zu den Nachteilen zählt die notwendige Nachbearbeitung der Messdaten bei der Verdeckung eines Markers. Aus diesem zusätzlichen Bearbeitungsschritt resultiert auch die Tatsache, dass optische Systeme in der Regel keine Echtzeiterfassung zulassen, da die Marker aufwändig aus den Bilddaten berechnet werden müssen [Hor02] [Kre13].

2.4.2 Magnetische Bewegungserfassung

Neben der optischen Erfassung werden Bewegungen über die Änderung eines Magnetfeldes erfasst. Die zu erfassende Person trägt dabei einen Anzug, auf dem Hall-Sensoren angebracht sind. Diese Sensoren messen ein Magnetfeld aus, das durch eine Transmittereinheit aufgebaut wird. Hierdurch wird die exakte Position im Raum bestimmt, vorausgesetzt die exakte Ausrichtung des Magnetfeldes ist bekannt.

Vorteile dieses Verfahrens sind die Echtzeitverfolgung und die Möglichkeit, mehrere Personen gleichzeitig zu verfolgen. Insbesondere stellt sich bei magnetischen Systemen nicht die Problematik eines verdeckten Markers. Der Preis ist außerdem deutlich günstiger als bei optischen Aufbauten. Ein Nachteil dieser Systeme ist jedoch, dass jeder Sensor ein eigenes und abgeschirmtes Kabel benötigt, um Störungen zu reduzieren. Trotzdem können durch metallische Gegenstände in der Nähe Störungen hervorgerufen werden, der Verwendungsbereich ist durch das benötigte Magnetfeld stark eingeschränkt [Hor02] [Kre13].

2.4.3 Mechanische Bewegungserfassung

Die dritte Möglichkeit ist die mechanische Erfassung von Bewegung. Bei dieser Methode trägt die zu erfassende Person einen Exoskelett, mit dem die Bewegung der einzelnen Gelenke direkt am Körper ermittelt wird.

Zusätzlich werden elektromechanische Sensoren, wie z.B. Gyroskope eingesetzt, um die Lage im Raum oder Drehungen zu ermitteln. Vorteile dieses System ist die universelle Einsetzbarkeit. Mechanische Systeme sind zusätzlich als Einzige in der Lage, eine Bewegungserfassung im Freien durchzuführen.

Abbildung 5: Motion Capturing mit einem Exoskelett [Jac06]

Prinzipiell können zudem beliebig viele Personen gleichzeitig erfasst werden, da sich die Sensoren nicht gegenseitig stören können. Ein Nachteil dieses Systems stellt die notwendige Anpassung des Exoskeletts auf den Benutzer dar. Die Gelenkabstände müssen sehr exakt gemessen und eingestellt werden. Liegen diese Werte allerdings einmal für einen Benutzer vor, lässt sich dieses System sehr schnell einsetzen. Nachteil ist ferner die im Vergleich zu den anderen Systemen geringe zeitliche Auflösung. Dazu kommt eine durch das Exoskelett eingeschränkte Bewegungsfreiheit [Hor02] [Kre13].

2.4.4 Bildbasierte Bewegungserfassung

Die günstigsten Systeme zur Bewegungserfassung sind bildbasierte Systeme. Hierbei ist keinerlei Hardware außer einer Kamera notwendig. Mit Hilfe eines Softwarealgorithmus wird die Bewegung aus einem oder mehreren Bildern rekonstruiert. Diese Technologie befindet sich noch in der Entwicklung. Sie könnte in Zukunft aber das Motion Capturing deutlich vereinfachen [Hor02] [Kre13].

In Tabelle 1 ist ein Vergleich zwischen den vier Bewegungserfassungssystemen hinsichtlich Genauigkeit, Freiheitsgraden, Flexibilität, Bewegungsfreiheit der Benutzer und Kosten dargestellt.

Tabelle 1: Vergleich der Bewegungserfassungssysteme

Erfassung	Genauigkeit	Freiheitsgrade	Flexibilität	Bewegungsfreiheit	Kosten
Optisch	+	+++	+++	+++	+
Magnetisch	++	++	+	++	-
Mechanisch	++	--	---	--	-
Bildbasiert	-	+++	+++	+++	++

2.5 Microsoft Kinect-System

Das Kinect-System von Microsoft (Abbildung 6) ist eine Kombination verschiedener Sensoren. Es wurde für die Steuerung von Spielen über Gesten, Bewegung und Sprache entwickelt. Dazu besteht das System aus einer Kombination von Farbkamera mit einem Tiefensensor und einem 3D-Mikrofon. Aus diesen Daten wird mittels einer speziellen Software die Position und Körperstellung der im Sichtfeld der Kamera erkennbaren Menschen ermittelt.

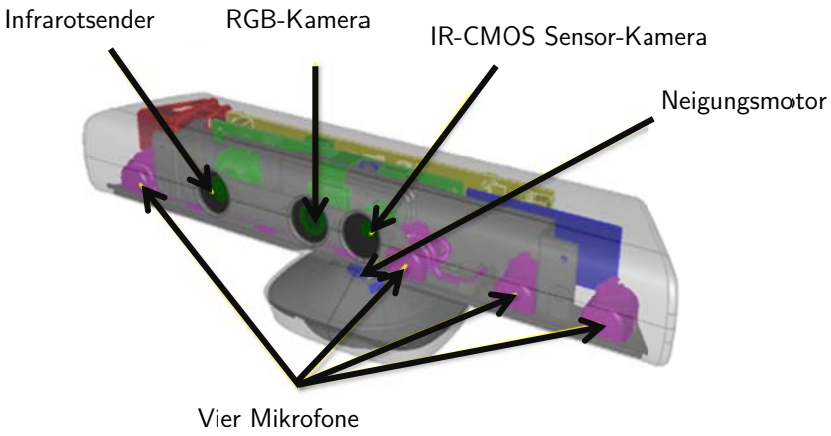


Abbildung 6: Microsoft Kinect Kamera [Mic11]

Das System wurde schon nach kurzer Zeit für Anwendungen außerhalb von Spielen benutzt [Gil10]. Das Spektrum der Anwendungen reicht von der Steuerung von Computeranwendungen [Gal11] über die autonome Erkennung von Objekten [Gre11] bis zur Steuerung des Flugverhaltens von Quadrokoptern [Sto11].

Die Innovation des Kinect-Systems ist jedoch der Tiefensensor. Er ermöglicht es, für jeden Bildpunkt des Kamerabildes eine Tiefe zu ermitteln, wodurch unterschieden werden kann, ob zwei benachbarte Bildpunkte zum selben Objekt gehören. Der Tiefensensor arbeitet mit einem als „Light Coding“ bezeichneten Verfahren [Alb07]. Bei diesem wird die Umgebung von einem Infrarot-Laser mit einem speziellen Punktmuster beleuchtet, eine Infrarot-Kamera nimmt die Reflexionen des Punktmusters auf (Abbildung 7). Objekte im Raum verzerren das reflektierte Muster, was eine Berechnung des Abstands von jedem Punkt zur Kamera ermöglicht.

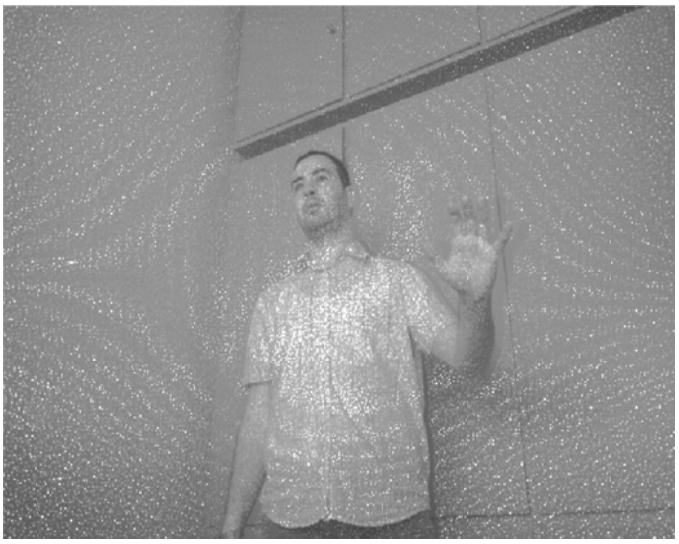


Abbildung 7: Punktmuster der Kinect Kamera

Abbildung 8 zeigt einen Vergleich zwischen RGB Bild und dem Tiefenbild derselben Szene. Anhand des RGB und Tiefenbild kann eindeutig zwischen den gezeigten Objekten unterschieden werden.

Abbildung 8: Vergleich von Kamerabild und Tiefenbild

In die zum Kinect-System gehörende Software ist ein Algorithmus implementiert, mit dem menschliche Formen im Tiefenbild erkannt und daraus für verschiedene Körperpartien die Position im Raum ermittelt wird [Sho11]. Die Ergebnisse werden in Form eines Skeletts mit

Gelenken ausgegeben. Die erkennbaren Körperpartien sind in Abbildung 9 dargestellt.

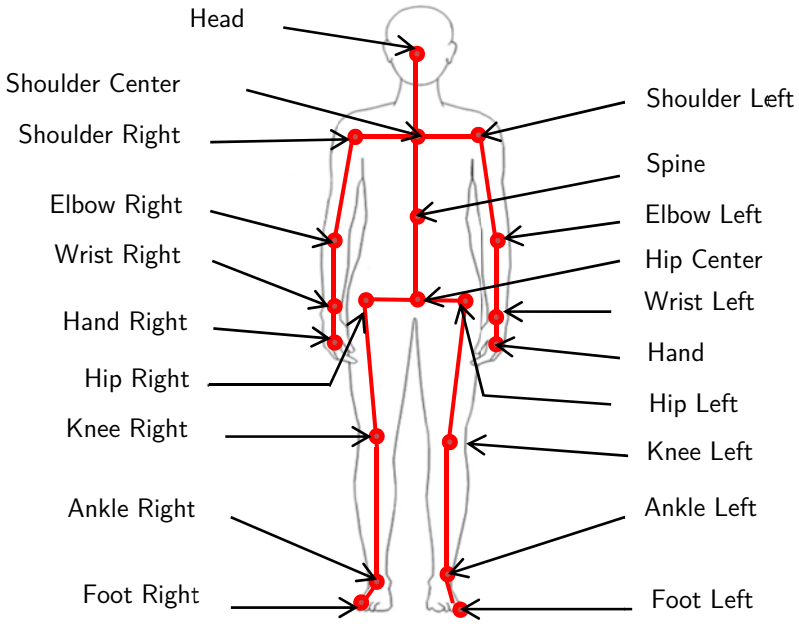


Abbildung 9: Kinect-Skelett [Mic11]

Das System kann die Körperstellungen von bis zu zwei Personen und die Position von vier weiteren Personen erkennen. Neben der Skeletterkennung verfügt das System auch über Routinen zur Erkennung von Gesichtsausdrücken.

Tabelle 2: Hardwareeigenschaften der Kinect Kamera [Mic11]

RGB Kameraauflösung	640x480 Pixel bei 30 fps
Farbtiefe	32 Bit
Sichtbereich	horizontal 57, vertikal 43
Auflösung der Tiefensensor	320x240 Pixel bei 30 fps
Erfassungsbereich	[1.20m , 3,50m]
Neigungswinkel	-27 bis +27°
Audio	4 Mikrophone, 16 Bit bei 16 kHz

Das zentrale Element der Kinect Entwicklung und des Software Developement Kits ist die NUI Library (Natural User Interface). Die NUI Library bildet die Schnittstelle zwischen Anwendung und Hardware. Mit ihrer Hilfe werden die drei Streams, also die Bilddaten, die Tiefeninformation und die Audiodaten, die die Kamera erzeugt, in der Software verarbeitet

und ausgegeben [Mic11]. In der NUI Library sind ebenfalls die für die Skeletterkennung notwendigen Algorithmen implementiert. In Abbildung 10 ist der Softwareaufbau der Kinect Kamera dargestellt.

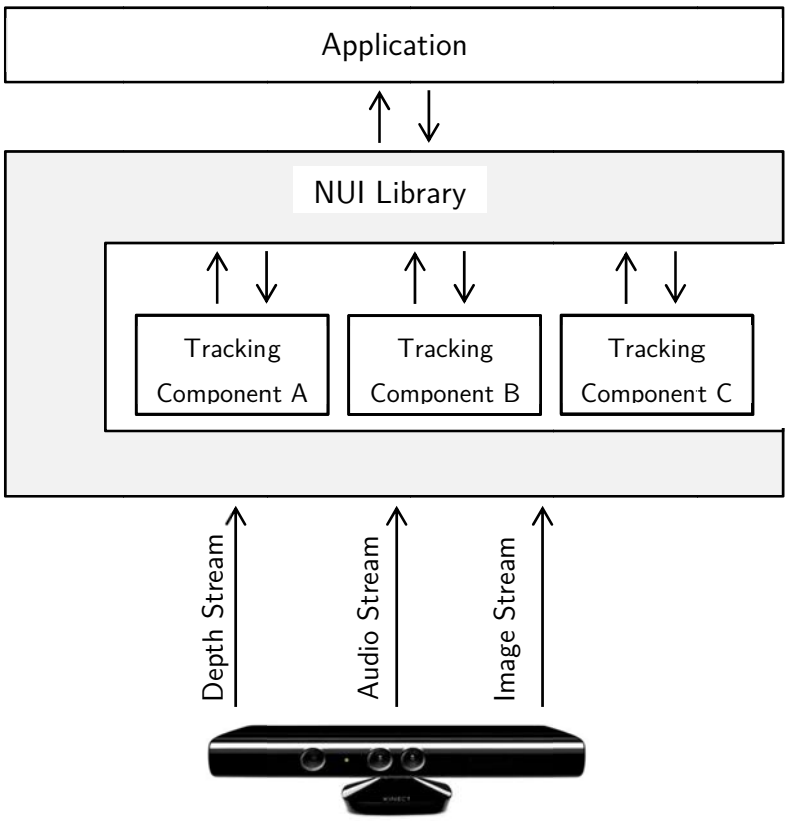


Abbildung 10: Übersicht über den Aufbau der Kinect Software [Mic11]

2.6 Compute Unified Device Architecture (CUDA)

Im Zusammenhang mit der für bildbasierte Systeme benötigten Bildbearbeitung, stellt die von Nvidia entwickelte Technologie CUDA (Compute Unified Device Architecture) eine leistungsstarke Infrastruktur bereit.

CUDA ist eine Programmiertechnik für parallele Berechnungen auf Nvidia Grafikprozessoren. Moderne Grafikkarten besitzen einer Vielzahl von Prozessorkernen (z.B. die Nvidia GTX 980 besitzt 2048 CUDA Kerne) die parallel genutzt werden können [Nvi14]. Dadurch werden hohe Rechenleistungen von mehreren TFLOPS/s erreicht [Pri09].

Im Gegensatz zu normalen Prozessoren haben Grafikprozessoren einen sehr beschränkten Befehlssatz und einen sehr kleinen Cache, wodurch der Einbau einer sehr hohen Anzahl von Recheneinheiten in einem kleinen Chip möglich ist. Diese enorme Anzahl von Recheneinheiten ermöglicht eine Partitionierung der Berechnungen auf einem Gitter dieser Recheneinheiten (Abbildung 11). Die Gitter bestehen aus mehreren Blöcken mit mehreren Threads. Innerhalb eines Gitters kann zwischen den einzelnen Threads sowohl ein Datentransfer als auch eine Koordination stattfinden [Far11].

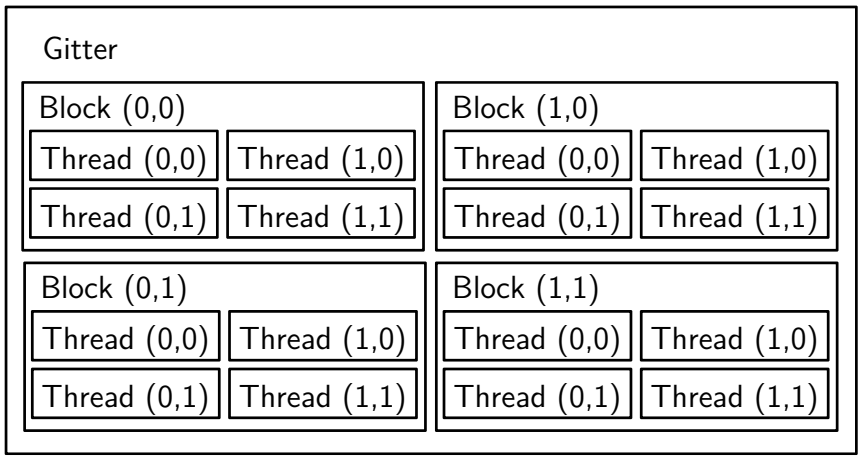


Abbildung 11: CUDA-Thread-Architektur [Pri09]

Die Kommunikation zwischen Blöcken bzw. Therads findet über die Speicherregister, Shared-Memory, Cachespeicher, Texturen-Speicher und den globalen Speicher statt.

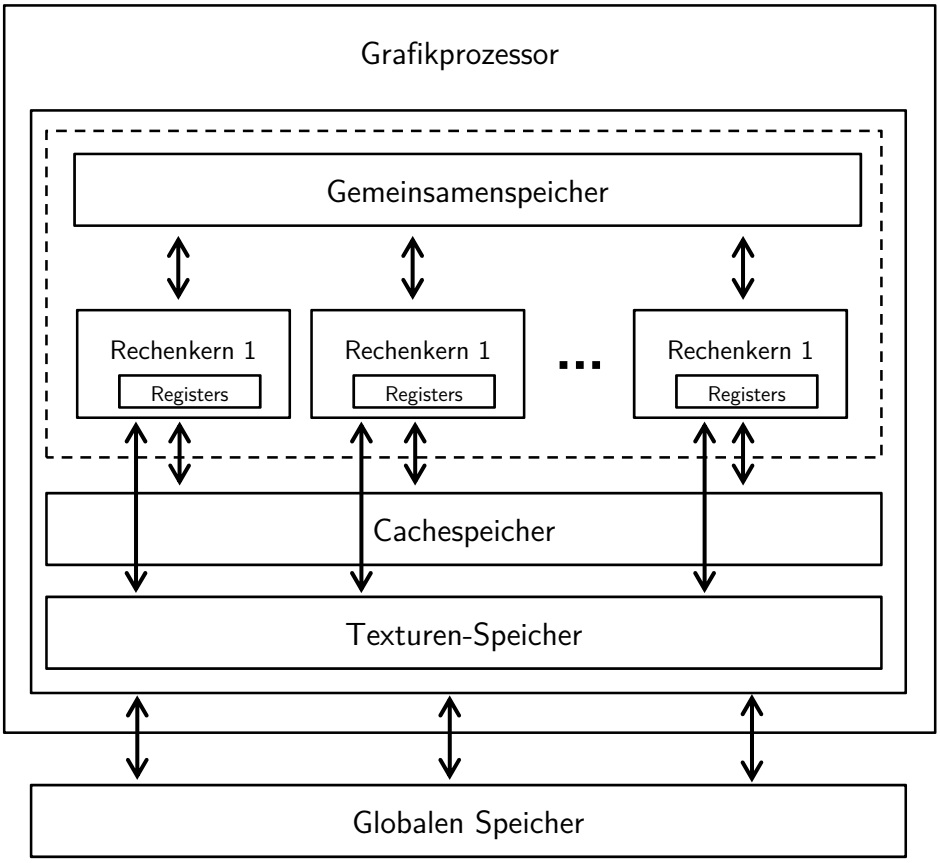


Abbildung 12: CUDA-Speicherarchitektur [Pri09]

2.7 Simultane Lokalisierung und Kartographierung

Die simultane Lokalisierung und Kartographierung (engl. Simultaneous localization and mapping - SLAM) ist ein Verfahren mit dem ein Roboter dessen Umgebung wahrnehmen kann, indem diese gescannt und kartographiert wird. Anhand der erstellten Karte und der im Roboter eingebauten Sensoren kann der Roboter in einer unbekannten Umgebung navigieren und Kollisionen vermeiden [Wan03].

Die Erzeugung einer dreidimensionalen Karte wird anhand der verwendeten Sensoren und Lokalisierungsverfahren in drei Kategorien klassifiziert [Nüc06]:

- planare 3D-Kartographierung (engl. Planar 3D mapping)
- scheibenweise 6D SLAM Methode (engl. Slice-wise 6D SLAM)
- vollständiges 6D SLAM (engl. Full 6D SLAM)

Mit Hilfe verschiedener Sensoren (Ultraschall, Laserscanner, Videokamera...) wird die Umgebung eines Roboter gescannt und anhand der gewonnenen Daten eine digitale Karte erstellt.

Die dabei auftretenden Fehler (Rauscheffekte, nicht zuordnungsfähige Messwerte und Messfehler) werden mit Hilfe verschiedener Filter eliminiert [Huh10].

2.7.1 Datenfilter

Um die erfassten Umgebungsdaten zu filtern werden üblicherweise die drei folgenden Filter verwendet:

Bilateraler Filter

Der Bilateral-Filter wird eingesetzt, um Bilddaten zu glätten und Messlücken durch einen Abgleich der Amplituden zu überbrücken. Die Besonderheit des Verfahrens liegt in der natürlichen Glättung sowie in der Erhaltung von Kanten [New11] [Tom98].

Auf Basis der Kombination von Reichweiten- und Feldfiltern werden Lücken bis zum Nachbarpunkt geschlossen. Das resultierende Element $D(u)$ berechnet sich wie folgt:

$$D(u) = \frac{1}{W_u} \sum_{q \in \mathcal{N}_q} G_{\sigma_s}(\|u - q\|_2) G_{\sigma_s}(\|R(u) - R(q)\|_2) R(q) \tag{2.6}$$

Non-Local Means (NL-Means) Filter

Der Vorteil dieses Filters liegt in der Erhaltung der Selbstähnlichkeit. Es können selbst feine Details vom Rauschen befreit werden [Huh10]. Beim NL-Means-Filter wird bei der Gewichtung hingegen das Zentralelement mit einem Cluster derer verglichen. Als resultierendes Element $D(u)$ ergibt sich dann:

$$D(u) = \frac{1}{Z_u} \sum_{q \in \mathcal{N}_q} w(u, q) R(u) \tag{2.7}$$

Das Gewicht wird wie folgt berechnet:

$$w(u, q) = e^{-\frac{1}{g} \sum G_\sigma(\|k\|_2)(R(u+k) - R(q+k))^2} \tag{2.8}$$

K-Nearest Neighbor (KNN) Filter

Der KNN-Filter ist ein sehr effizientes Verfahren zur Reduktion von Rauscheffekten. Es wird innerhalb einer umliegenden Nachbarschaft $\mathcal{N}_q$ jedes Element in Abhängigkeit von der euklidischen Distanz gegenüber der Gauß-Standardabweichung gewichtet [Kha07].

$$D(u) = \frac{1}{C_u} \sum_{q \in \mathcal{N}_q} R(y) e^{\frac{|R(q)-R(u)|^2}{{G_1}^2}} e^{\frac{|q-u|^2}{{G_2}^2}} \tag{2.9}$$

2.7.2 Klassifikation von Außenseitern

Um Außenseiter (nicht zuordnungsfähige Messwerte) zu identifizieren und zu eliminieren werden folgende drei Verfahren angewendet:

Plane Fit Kriterium (PF-Kriterium)

Beim PF-Kriterium wird der Abstand eines Messpunktes zu einer Fläche, welche aus der durchschnittlichen quadratischen Distanz von mehreren nacheinander liegenden Punkten generiert. Dieses Verfahren wird ebenfalls angewendet, um Rauscheffekte zu reduzieren [Wey04].

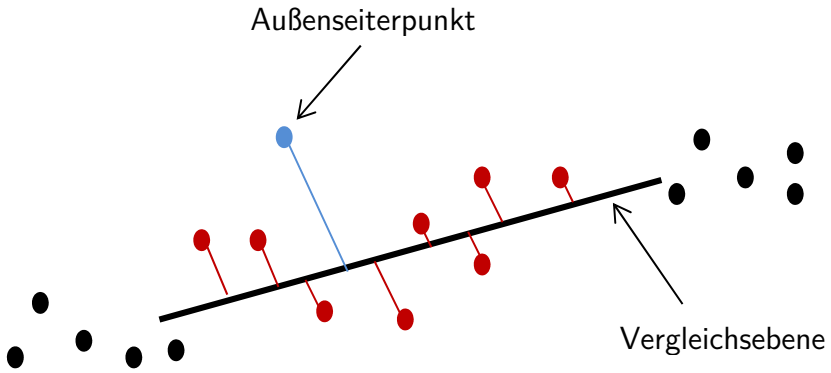


Abbildung 13: Plane Fit Kriterium [Wey04]

Miniball-Kriterium

Beim Miniball-Kriterium wird ein Punkt als Außenseiter definiert, wenn dieser außerhalb einer Sphäre mit einem vordefinierten Radius von nacheinander liegenden Punkten liegt [Wey04].

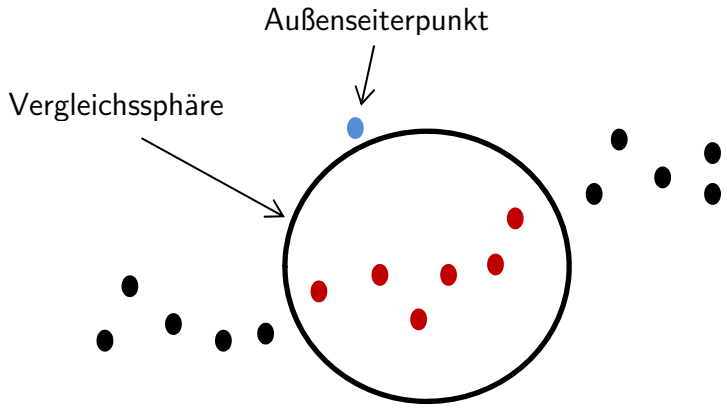


Abbildung 14: Miniball-Kriterium [Wey04]

Nearest-Neighbor-Reciprocity-Kriterium (NNR)

Das NNR-Kriterium basiert auf der Annahme, dass gültige Punkte innerhalb einer Nachbarschaft eines Außenseiters liegen, jedoch ein Außenseiter nicht innerhalb einer Nachbarschaft liegen kann.

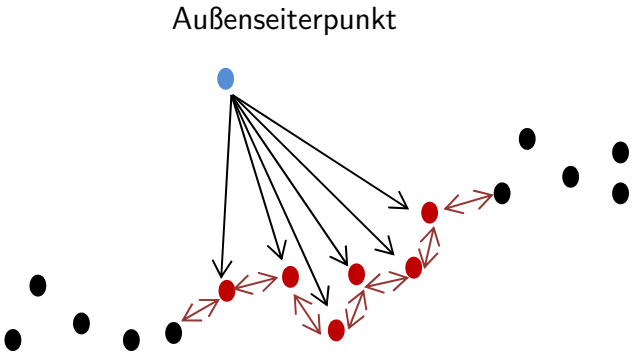


Abbildung 15: NNR-Kriterium [Wey04]

2.8 Bildsynthese mit dem Lochkameramodell

Die Erzeugung eines Bildes (Szene) aus Rohdaten wird Bildsynthese oder Rendering genannt. Eine Szene ist ein virtuelles räumliches Modell, das mit Hilfe unterschiedlicher Rendering-Tools (Lochkameramodell, Strahlverfolgung, Sichtbarkeitsberechnung, Schattierung...) generiert wird [Tön10].

Mit der Annahme, dass alle Punkte eines Raumes sich über ein einen Brennpunkt/Lochblende auf einer Ebene abbilden lassen, werden 3D-Elemente auf einem 2D-Bildschirm erzeugt. Dieses Verfahren wird Lochkameramodell (engl. Pinhole camera) genannt [Jäh12].

Die ausgehenden Strahlen aller Punkte P_i mit den Koordinaten $[x_i, y_i, z_i]$ (Abbildung 16) werden im Brennpunkt gebündelt und auf der Bildebene gespiegelt als Punkte P'_i mit den Koordinaten $[x'_i, y'_i, z'_i]$ abgebildet [Tön10] [Jäh12].

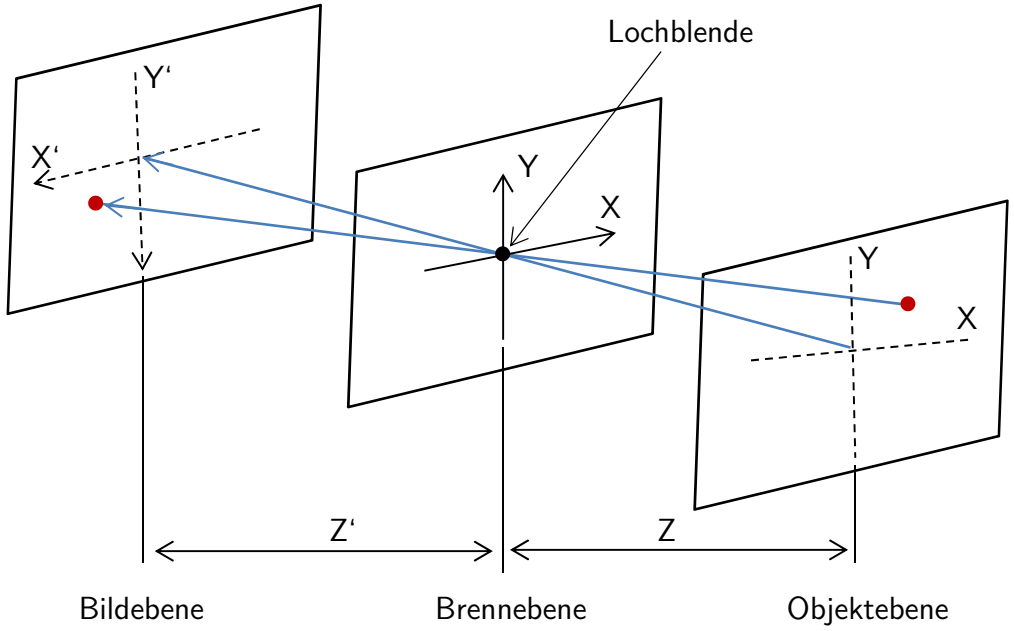


Abbildung 16: Bilderzeugung mit der Lochkamera [Jäh12]

2.9 Echtzeitsysteme und Echtzeitsteuerung

Die Steuerung von Maschinen setzt voraus, dass auf externe Ereignisse in einer angemessenen Zeit reagiert werden kann. So ist zum Beispiel ein Not-Aus-Schalter an einer Maschine, der die Maschine erst nach einer zufälligen, langen Zeitspanne abstellt, eine akute Gefahr für Leib und Leben.

Systeme und Prozesse, bei denen die Reaktionszeit festgelegte Anforderungen erfüllen muss, werden Echtzeitsysteme genannt. Typischerweise wird zwischen harter und weicher Echtzeit unterschieden [Wör05]. Bei harten Echtzeitsystemen ist ein Einhalten der Reaktionszeit absolut zwingend. Nach Ablauf der festgelegten Zeit muss die Reaktion auf ein Ereignis

vollständig abgeschlossen sein, da ansonsten Schäden am System oder für die Umgebung entstehen können. Das oben aufgeführte Beispiel eines Not-Aus-Schalters ist ein Beispiel für harte Echtzeit.

Bei einem weichen Echtzeitsystem wird zwar die Einhaltung der Reaktionszeit angestrebt, aber ein Überschreiten führt nicht unmittelbar zu einer Fehlfunktion. Nach Ablauf der Reaktionszeit nimmt die Bedeutung des Ergebnisses der Reaktion ab, bis es keine Rolle mehr spielt. Ein Beispiel hierfür ist ein digitales Telefongespräch. Damit das Gespräch störungsfrei abläuft, müssen alle Datenpakete früh genug beim Empfänger ankommen. Einzelne verspätete Pakete führen zwar zu kurzen Aussetzern, aber nicht zum vollständigen Abbrechen des Gesprächs [Ben09].

Einige Autoren verwenden weiterhin den Begriff feste Echtzeitsysteme [Ben09]. Der Unterschied zwischen weicher und fester Echtzeit ist, dass die Bedeutung der Reaktion bei weicher Echtzeit nach Überschreiten der Reaktionszeit langsam abnimmt während sie bei fester Echtzeit unmittelbar auf null fällt.

Die meisten real existierenden Systeme stellen eine Mischung der drei Typen dar. Bei einem komplexen System wie einem Roboter treten unzählige Ereignisse auf, von denen jedes andere Echtzeitanforderungen haben kann.

Umgangssprachlich wird Echtzeit oft mit einer besonders schnellen Reaktion gleichgesetzt, was aber keineswegs der Fall ist. Echtzeitsysteme befassen sich nur damit, ob eine Reaktion innerhalb einer festgesetzten Zeit erfolgt. Auch ein System mit einer Reaktionszeit von mehreren Tagen kann noch ein hartes Echtzeitsystem sein.

Es gibt verschiedene Methoden, mit denen ein Echtzeitsystem praktisch umgesetzt werden kann [Wör05]. In der einfachsten Variante wird für jedes Ereignis dezidierte Verarbeitung, zum Beispiel durch eine elektronische Schaltung, realisiert. Es muss dann nur noch darauf geachtet werden, dass die Verarbeitungszeit innerhalb der erlaubten Zeit bleibt. Verschiedene Ereignisse beeinflussen sich nicht gegenseitig, da sie gleichzeitig bearbeitet werden. Ein großer Nachteil ist die dadurch entstehende Komplexität, da alle Teilergebnisse zueinander asynchron sind und für die weitere Verarbeitung erst aufwändig synchronisiert werden müssen.

Diese Komplexität kann vermieden werden, indem die einzelnen Ereignisse in einer Schleife nacheinander abgearbeitet werden. Dadurch steigt aber für alle Ereignisse die kürzeste

erzielbare Reaktionszeit auf die Durchlaufzeit der Schleife. Die Erweiterung des Systems um zusätzliche Ereignisse verlängert also die Reaktionszeit aller Ereignisse.

Bei umfangreicheren Problemstellungen werden häufig sogenannte Prozesse verwendet [Liu73]. Bei dieser Technik werden die Verarbeitungsroutinen in verschiedenen Prozessen gruppiert. Jedem Prozess wird eine Priorität zugewiesen. Es wird immer der Prozess mit der höchsten Priorität aus allen aktiven Prozessen ausgeführt.

Ein Prozess wird entweder in regelmäßigen Abständen, oder als Reaktion auf ein externes Ereignis auf aktiv gestellt. Wenn der aktivierte Prozess eine höhere Priorität als der gerade laufende Prozess hat, wird dieser unterbrochen und der neue Prozess ausgeführt.

Die Reaktionszeit eines Prozesssystems ergibt sich aus der Laufzeit der Verarbeitung zusammen mit der nötigen Zeit für einen Prozesswechsel. Zusätzlich müssen noch die maximalen Laufzeiten aller höher priorisierten Prozesse berücksichtigt werden. Bei falscher Auslegung kann ein Prozess „verhungern“: Wenn die höher priorisierten Prozesse andauernd aktiv sind, werden die Prozesse mit geringerer Priorität nicht mehr abgearbeitet. Ein prozessbasiertes Echtzeitsystem kann daher nur verwendet werden, wenn für jedes Ereignis eine maximale Auftrittsfrequenz definiert werden kann, anhand der die maximal nötige Rechenleistung berechnet werden kann [Hei94] [Ben09].

Ein System egal welcher Ausprägung kann nur dann harte Echtzeitanforderungen erfüllen, wenn alle Reaktionen deterministisch ablaufen. Um dies sicherzustellen, muss die Implementierung des Systems auf Korrektheit geprüft werden. Dies ist insbesondere für eine Umsetzung als Software problematisch, da die Verifizierung von Computerprogrammen schwierig ist [Hei94].

2.10 Modularität

Der Begriff „Modularität“ stammt von dem lateinischen Wort „modulus“ ab, welches „kleines Maß“, „Maßstab“ oder „Modell“ bedeutet. Die Bezeichnung „Module“ für standardisierte Baueinheiten wird jedoch erst ab etwa 1955 erstmal in der englischen Fachliteratur verwendet [Mer07].

Modularität im technischen Bereich ist die Aufteilung eines Systems in Teile, die als Module, Komponente oder Bausteine genannt werden, diese Module sind funktional und Physikalisch unabhängig und besitzen eine klar definierte Einflusszuordnung [Göp98].

Modularisierung beschreibt auch die Reduktion der Anzahl von Elementen innerhalb eines Systems durch die Gruppierung der Elemente in eine kleinere Anzahl von Subsystemen [Mar06].

Die Kommunikation zwischen den Modulen wird durch das Black-Box-Prinzip umgesetzt, d.h. der Datenaustausch ist nur über explizite Schnittstellen zugänglich [Ste01].

2.11 Mikrokontroller-Ringpuffer

Für die Interprozesskommunikation zwischen den Systemteilen eines mechatronischen Systems ist eine geeignete Prozesssynchronisation notwendig. Bei einem Mikrokontroller wird ein Pufferspeicher nach dem „Warteschlangen-Prinzip“ verwendet. Die Umsetzung dieses Prinzips in einer Software wird Ringpuffer genannt [Sim03] [Ste11].

Ein Ringpuffer ist logisch gesehen ein Puffer mit einer bestimmten Anzahl von Speicherplätzen, die ohne Anfang und Ende aneinandergereiht sind. Die Dateneingabe und -entnahme wird über zwei Zeiger gesteuert. Der Lesezeiger zeigt auf das erste Pufferelement mit gültigen Daten. Wenn dem Puffer ein Element entnommen werden soll, wird der Lesezeiger um einen Speicherplatz auf das nächste Element verschoben. Der Schreibzeiger zeigt auf das erste freie Element hinter dem letzten belegten Element. Ein neues Element wird an der Stelle des Schreibzeigers geschrieben, der danach auf das nächste Element verschoben wird. Der so gebildete Puffer stellt einen FIFO-Puffer (First In, First Out) dar, da die Elemente in der gleichen Reihenfolge entnommen werden, wie sie gespeichert wurden. [Sim03] [Ste11]

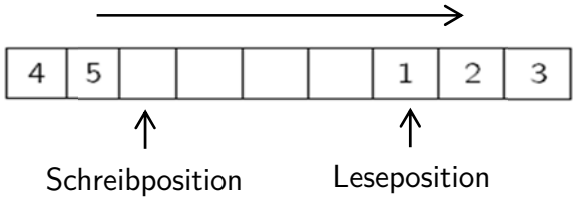


Abbildung 17: Ringpuffer mit 12 Speicherplätzen

In der praktischen Umsetzung muss beachtet werden, dass der Speicher eines Mikrocontrollers nur linear und nicht ringförmig angesprochen werden kann. Daher wird der logische Ring an einer Stelle aufgeschnitten. Bei jedem Verschieben der beiden Zeiger muss geprüft werden, ob der Zeiger nun außerhalb des Pufferbereichs ist und gegebenenfalls wieder auf das erste Element gesetzt werden. Abbildung 17 verdeutlicht den Zusammenhang zwischen den Zeigern und den Speicherplätzen. [Sim03] [Ste11].

3 Architektur des modularen Gestensteuerungssystems

Um den Anforderungen aus der Aufgabenstellung gerecht werden zu können, werden folgende Grundüberlegungen und Prämissen getroffen:

- Der in dieser Arbeit entwickelte Roboter ist eine Kombination aus den drei Roboterarten „*Variable Sequence Robot*“, „*Playback Robot* „ und „*Intelligent Robot*“ gemäß JARA.
- Bei dem in dieser Arbeit entwickelten System/Steuerung handelt es sich um eine direkte Steuerung anhand optischer Bewegungserfassung.
- Für die Transformation von detektierten Bewegungsdaten in die Gelenkwinkel des Roboters wird eine Kombination aus Vorwärts- und Rückwärtstransformationen verwendet.
- Als Typ der simultanen Lokalisierung und Kartographierung wird Full 6D Slam verwendet.
- Die Steuerung ist als hartes Echtzeitsystem mit einem Bearbeitungsintervall von 33ms auszuführen.
- Um für Teleoperation eine maximale Immersion des Bedieners zu erreichen, muss der Roboter seine detektierten Umgebungsdaten in Form einer virtuellen Realität visualisieren.

3.1 Aufbau und Funktionsprinzip des Systems

In Abbildung 18 wird das Prinzip des gesamt Systems dargestellt. Mit Hilfe einer Kinect Kamera werden die Bewegungen des Benutzers erfasst, bearbeitet und als Steuerungsbefehle an den Roboter in der realen Umgebung geleitet. Anhand der im Roboterkopf eingebauten Stereo-Kamera sowie Laserscanner wird die Roboterumgebung digitalisiert und in eine virtuelle Umgebung vor dem Benutzer rekonstruiert/projiziert. Bei der Übertragung von Daten zwischen realer und virtueller Umgebung sind alle Regelkreise aufeinander abgestimmt und Verzögerungen eliminiert.

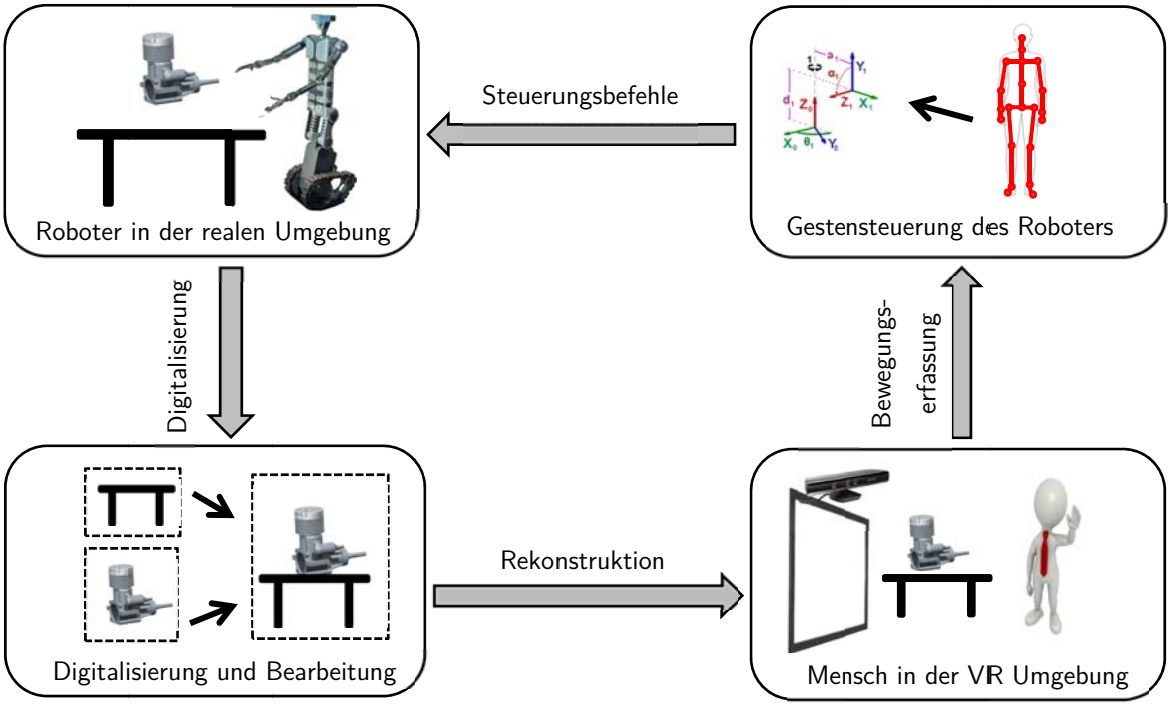


Abbildung 18: Aufbau und Funktionsprinzip des Systems

3.2 Spezifikation des Systems

Wegen der Komplexität und dem geplanten Realisierungszeitrahmen von drei Jahren wurde das Projekt in drei Unterprojekte (PC Software, Echtzeit Embedded System (EES) und Humanoid-Roboter) aufgeteilt, die gleichzeitig gestartet und parallel realisiert wurden. Für jedes dieser Unterprojekte wurde eine Anforderungsliste erstellt.

3.2.1 Spezifikation der Software

Die Software soll anhand der bildbasierten Bewegungserfassung die Position sowie alle Gelenke einer Person im Raum erkennen. Aus deren Haltung sollen die Zielpositionen für einen Humanoid-Roboter berechnet werden. Die Software soll auch in der Lage sein, die Roboterumgebung in Echtzeit aufzunehmen und in einem virtuellen 3D Raum zu rekonstruieren.

Die Software muss in C/C++ programmiert werden und auf einem 64-Bit Windows basierten Rechner betrieben werden.

3.2.2 Spezifikation der Echtzeit Embedded System (EES)

Das Embedded System soll einen echtzeitfähigen Prozessor mit mindestens 100MIPS (Million Instructions Per Second) besitzen. Er soll 16 Gelenke steuern und 16 Winkelsensoren gleichzeitig auswerten. Er soll auch das Signal von 10 Berührungssensoren auslesen und auswerten. Die Kommunikation mit dem Steuerungsrechner soll mit Hilfe von USB, Netzwerk und RS232 Schnittstellen stattfinden.

Das System muss zu einem NOT-AUS-STOP der Kategorie 0 (Energiezufuhr der Antriebselemente wird sofort getrennt) kompatibel sein.

Das System soll einen wiederbeschreibbaren internen Speicher zum dauerhaften Speichern von Konfigurationen besitzen und gegen Stromausfälle geschützt werden.

3.2.3 Spezifikation des Humanoid-Roboters

Der Roboter ist ein Demonstrator für das Gestensteuerungssystem und nicht für den produktiven Einsatz gedacht und soll aus kostengünstigen Komponenten gebaut werden. Als Humaniod-Roboter soll das System über zwei Arme mit jeweils mindestens sechs Freiheitsgraden verfügen. Der Kopf soll zwei Freiheitsgrade besitzen und mit einem 3D Scanner sowie einer Full HD Stereokamera ausgestattet sein. Die Hände des Roboters sollen wie eine menschliche Hand gestaltet sein und über fünf Finger mit Berührungssensoren verfügen.

Der Roboter darf maximal 90 kg wiegen, die maximale Hubkraft des Roboters mit gestrecktem Arm soll größer 4,9 Newton (500g) sein.

Der Antrieb soll mit Schrittmotoren realisiert und die Gelenke jeweils mit Winkelsensoren ausgestattet werden.

3.3 Module des Gestensteuerungssystems

Das entwickelte System wurde in mehrere Module mit definierten Schnittstellen aufgeteilt. Zwischen dem Benutzer und der Maschine sind mehrere Teilmodule parallel und seriell geschaltet (Abbildung 19). Die einzelnen Module und Teilmodule lassen sich zu einem Gesamtsystem kombinieren. Die Module/Teilmodule können leicht ausgetauscht oder teilweise abgeschaltet werden.

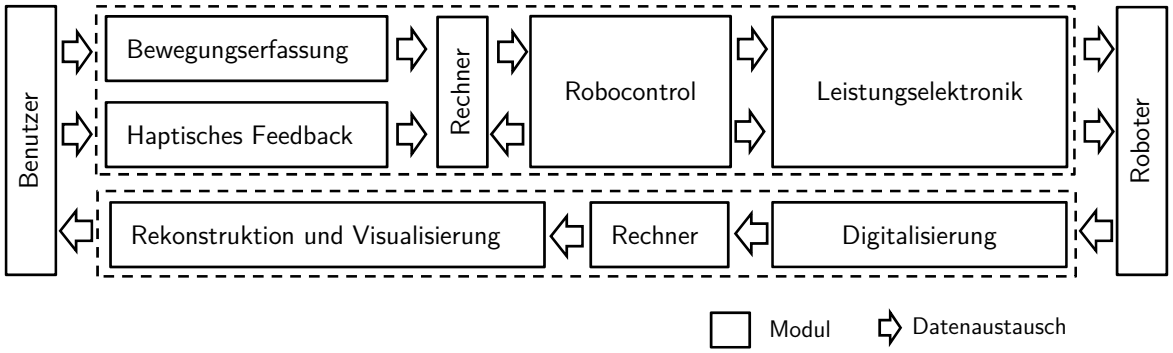


Abbildung 19: Module des Gestensteuerungssystems

Um die Modularität des Systems zu gewährleisten, wurden für die einzelnen Module geeignete Schnittstellen mit möglichst höchster Kompatibilität zu anderen Komponenten definiert.

Für den Datenaustausch zwischen den Modulen wurde ein spezielles Kommunikationsprotokoll entwickelt (Siehe Abschnitt 4.3).

Bewegungserfassung:

Die Bewegungserfassung des Bedieners wurde in drei Bereiche geteilt: Kopf, Hände und Körper (Abbildung 20).

Die Erfassung der Körperbewegung wird mit Hilfe von einer RGB Kamera mit einer Auflösung von 1280x720 Pixel und einer 3D Infrarotkamera mit einer Auflösung von 640x480 Pixel realisiert.

Für die Gesichtserkennung und Kopfbewegungserfassung wurden zwei baugleiche Kameras wie bei der Körpererfassung verwendet. Mit diesen zwei Kameras wird das Gesicht herangezoomt, um Details zu erkennen (Augen, Mund und Nase). Diese sind notwendig um den Blickwinkel des Bedieners zu ermitteln.

Die Fingererfassung kann nicht mit einer stationären Kamera erfolgen, weil die Hände in bestimmten Stellungen durch den Körper verdeckt werden. Die Bewegungserfassung der Finger erfolgt deshalb mit Hilfe von 10 Flex-Sensoren, die in einen Handschuh integriert sind.

Die Datenübertragung zwischen dem Bewegungserfassungsmodul und der Rechnereinheit wird mit Hilfe von zwei USB und einer RS232 Schnittstelle realisiert.

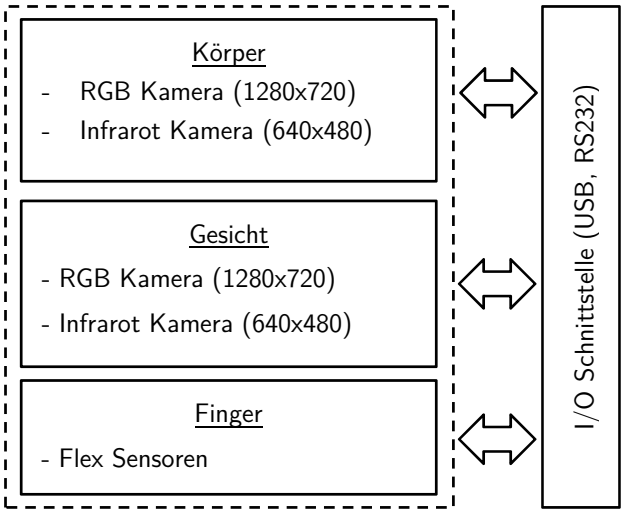


Abbildung 20: Bewegungserfassung

Rechnereinheit:

Die Rechnereinheit (Abbildung 21) ist für die Bearbeitung der Rohdaten der vier Kameras und 10 Flex Sensoren zuständig. In diesem Modul werden das Körperskelett, der Blickwinkel und die Fingerposition des Bedieners ermittelt, visualisiert und an das RoboControl-Modul weitergegeben.

Die Datenbearbeitung wird in mehreren Schritten parallel mit Hilfe der SMT Technik (Simultaneous Multithreading) realisiert [Ber05].

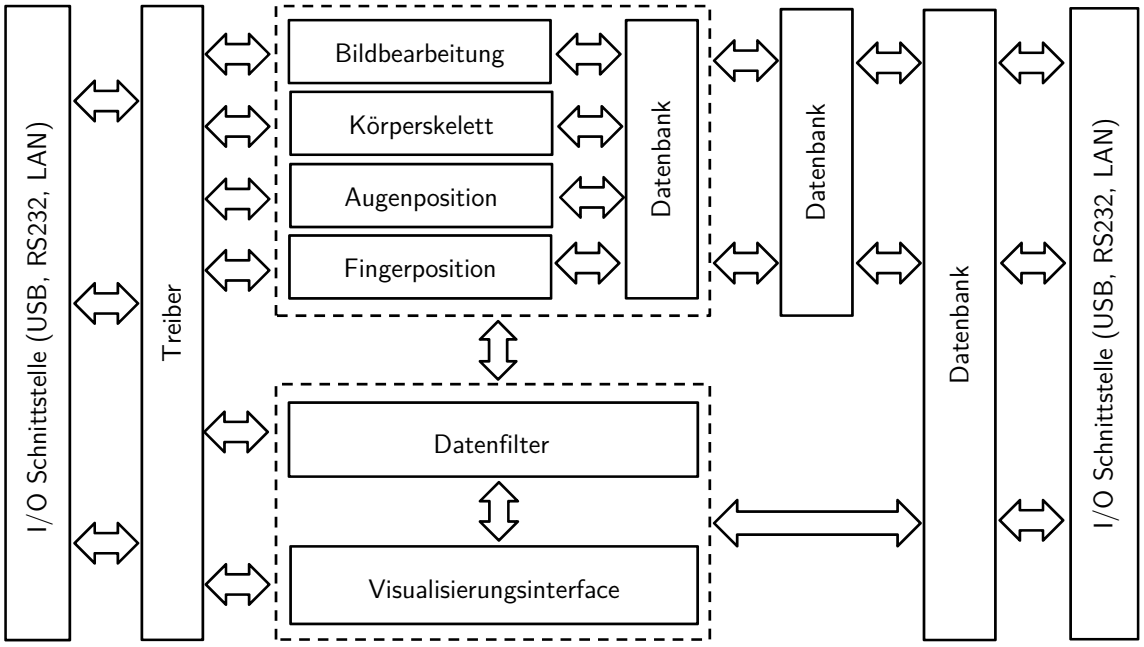


Abbildung 21: Rechnereinheit

Steuerungsmodul „RoboControl“:

Die ermittelten Daten (Körperskelett, Blickwinkel und Fingerposition) werden mit Hilfe von einer USB, einer RS232 und einer LAN Schnittstelle an das Steuerungsmodul „RoboControl“ (siehe Abschnitt 4.3.2) weitergegeben.

Das Steuerungsmodul „RoboControl“ (Abbildung 22) wertet alle Signale der Sensoren aus und sorgt für die notwendige Weiterverarbeitung. Anhand der gewonnenen Daten (Körperskelett, Blickwinkel, Fingerposition und die Robotersensorwerte) wird die Stellung aller Gelenke des Roboters mittels der vorhandenen Schrittmotoren gesteuert.

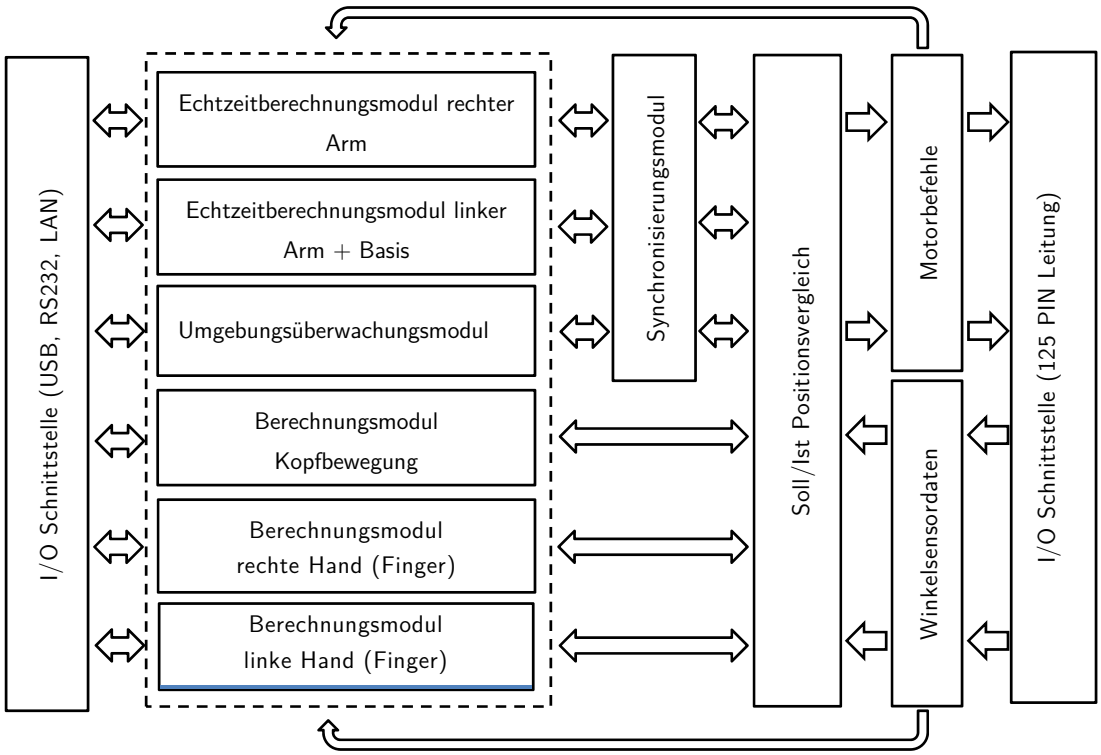


Abbildung 22: Steuerungsmodul (RoboControl)

Leistungselektronik:

Das Leistungselektronikmodul (Abbildung 23) ist die Einheit, die alle Steuersignale umwandelt und damit die Verbindung zwischen dem Steuerungsmodul „RoboControl“ und den Antrieben des Roboters herstellt.

Das Modul besteht aus drei Teilmodulen. Das erste Teilmodul ist für die Schrittmotorenansteuerung zuständig und besteht aus dem Impulsinterface und den Motorentreibern. Das

zweite Teilmodul ist das Sicherheitsinterface, welches den Zustand des Not-Aus, die Feedbacksensoren sowie die Systembewegungsgrenzen überwacht.

Das dritte Teilmodul ist das Winkelsensoren-Interface, welches alle Sensorwerte auswertet, die Signale verstärkt und sie zum RoboControl Modul weiterleitet.

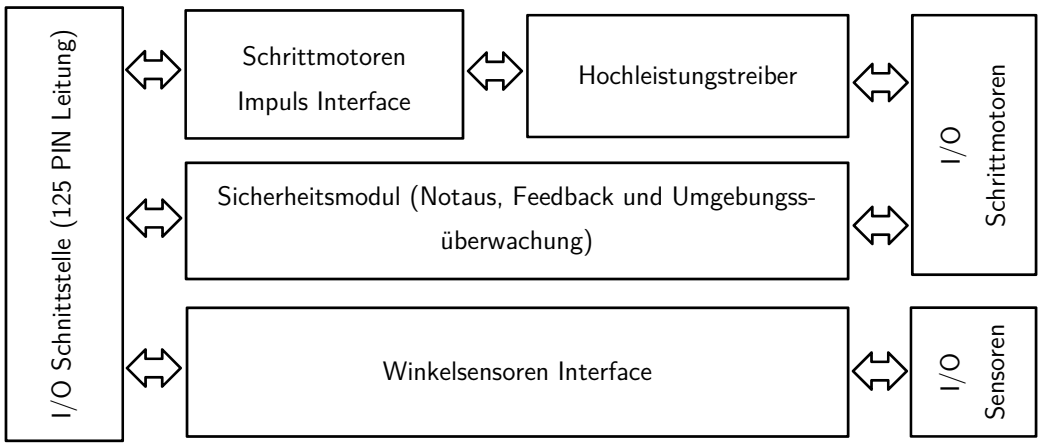


Abbildung 23: Leistungselektronik

Für die Datenübertragung zwischen dem RoboControl-Modul und dem Leistungselektronikmodul wurde auf konventionelle Schnittstellen verzichtet, weil die nötigen Anforderungen (Reaktionszeit, Impulslänge, Baud-Rate und Overhead) nicht erfüllt werden können. Um alle Motoren und Sensoren gleichzeitig anzusteuern, wurde eine parallele Kommunikation entwickelt und mit Hilfe von einer 125 Pin Leitung realisiert.

Umgebungssensoren Modul:

Zur dreidimensionalen digitalen Rekonstruktion der realen Umgebung des Roboters in einem VR Raum werden drei Kameras genutzt: eine Full HD Stereo Kamera, eine 3D Infrarot Kamera und eine RGB Kamera (Abbildung 24).

Darüber hinaus wurden die Finger des Roboters mit Berührungssensoren ausgestattet.

Die Datenübertragung wird mit Hilfe von drei USB Datenleitungen realisiert.

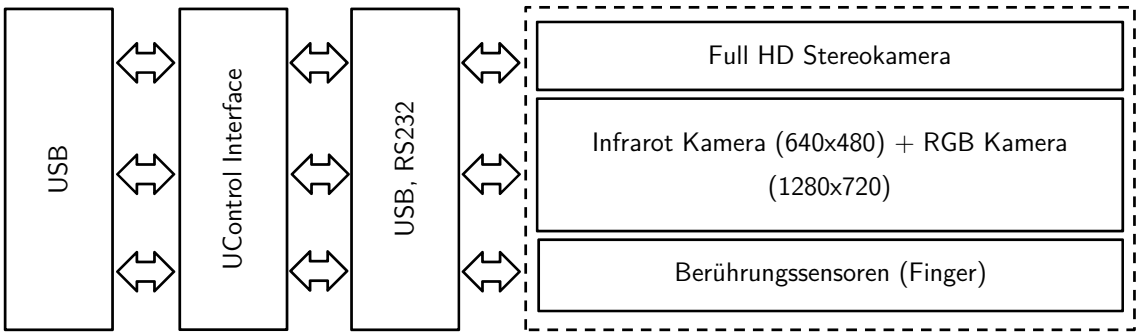


Abbildung 24: Umgebungssensoren

4 Software

Zur Steuerung des Humanoid-Roboters „Roki“ (siehe Abschnitt 5.3) wurden vier Software-Module entwickelt, eines für die Bewegungserfassung und Bildbearbeitung, ein Kernel, der für die ganze Berechnung der Positionssteuerung und Kommunikation zuständig ist, eine Software zur Visualisierung und 3D-Rekonstruktion der Roboterumgebung in einen virtuellen 3D Raum, sowie hardwarenahe Software, wie z.B. die Treiber und die Firmware der Mikrocontroller.

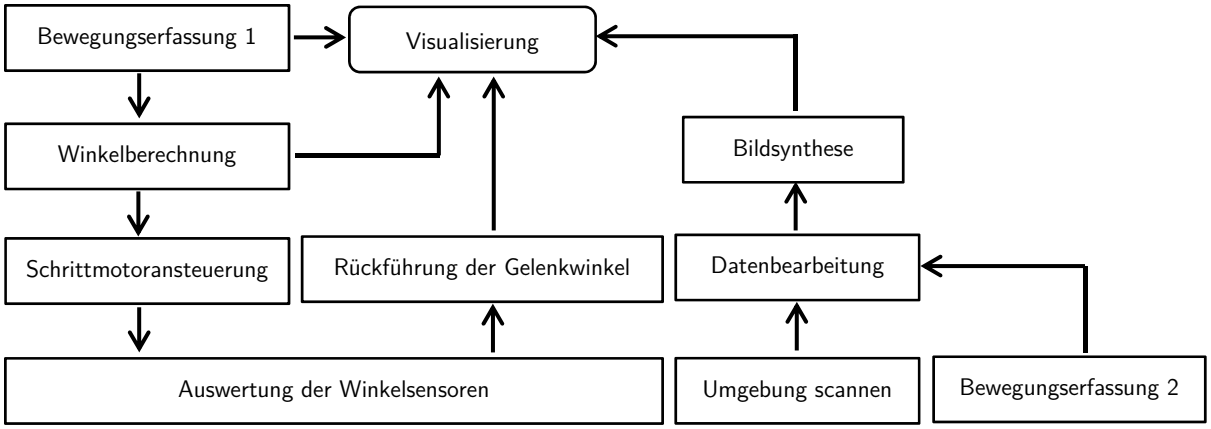


Abbildung 25: Vereinfachte Architektur der Gesamtsoftware

4.1 Bewegungserfassung

Dieser Abschnitt beschreibt die mathematischen Grundlagen, auf den die Software basiert. Es wird dargelegt, wie die mittels Kamera erfasste Körperstellung eines menschlichen Bedieners in Bewegungen des Roboters umgesetzt wird.

Zuerst wird die Umrechnung von Körperstellung in Gelenkpositionen des Roboters und zurück dargestellt. Anschließend wird gezeigt, wie daraus Bewegungsbefehle für die Motoren des Roboters generiert werden.

Die Festlegung der gewünschten Gelenkstellungen für den Roboter geschieht dabei in zwei Schritten: Zuerst wird aus dem Bildsignal der Kamera die Position und Stellung aller Gliedmaßen des Bedieners ermittelt. Anschließend werden aus diesen Werten die erforderlichen Gelenkstellungen des Roboters berechnet.

4.1.1 Erfassen des Skeletts und Winkelberechnung

Im ersten Schritt werden die Positionen aller Gliedmaßen des Bedieners ermittelt. Hierzu dient eine Kinect-Kamera von Microsoft. Diese erfasst, wie in Abschnitt 2.5 beschrieben, ein Farbbild und ein Tiefenbild. Aus diesen beiden Bildern bestimmt die Treibersoftware der Kinect-Kamera ein Skelett-Model, welches die Position aller wichtigen menschlichen Gelenke angibt.

Zuerst wird das Tiefenbild des menschlichen Körpers auf zusammenhängende Regionen mit gleicher Tiefe untersucht. Die gefundenen Regionen werden anschließend mittels mehrerer gemischter Entscheidungsbäume Körperteilen zugeordnet. Diese Entscheidungsbäume wurden zuvor anhand einer Vielzahl von Testbildern erstellt. Abschließend werden die gefundenen Bildkoordinaten in Raumkoordinaten umgewandelt.

Die so gewonnenen Positionsdaten sind noch mit starken Schwankungen behaftet und müssen vor der Verwendung gefiltert werden [Azi09].

Um die Fehler zu eliminieren werden als erstes kurze Ausreißer aus den Eingabepositionen X_n entfernt. Hierzu wird die neue Position mit dem Median X_{med} der letzten Positionen verglichen und nur dann übernommen, wenn sich die Position nicht stärker als ein Schwellenwert t ändert.

$$X_n = \begin{cases} x_n & wenn\ |x_n - X_{med}| < t \\ X_{med} & andersfalls \end{cases} \tag{4.1}$$

Die so vorgefilterten Werte werden mit einem doppelt exponentiellen Filter weiter geglättet. Bei diesem Algorithmus werden zwei exponentielle Glättungsfilter kombiniert. Jeder Filter wird durch einen gleitenden Durchschnitt mit exponentieller Gewichtung realisiert [Lav03].

Zuerst wird aus den vorhergehenden Ergebnissen ein Trend ermittelt. Mit dem Parameter γ kann die Stärke der Glättung bestimmt werden.

$$b_n = \gamma\left(\hat{X}_n - \hat{X}_{n-1}\right) + (1-\gamma)b_{n-1} \tag{4.2}$$

Anschließend wird der Trend mit dem aktuellen Eingabewert X_n verrechnet. Mit dem Parameter α wird eingestellt, wie schnell das Ergebnis zum Eingabewert konvergiert.

$$\hat{X}_n = \alpha X_n + (1-\alpha)(\hat{X}_{n-1} - n_{n-1)} \tag{4.3}$$

Aus den geglätteten Positionen der Gliedmaßen werden die Winkelstellungen der Roboterarme abgeleitet.

Hierzu werden aus den Stellungen der Gelenke und den Richtungsvektoren der verbindenden Gliedmaßen bestimmt. Die Zusammenhänge von Gelenken und Gliedmaßen können Abbildung 26 entnommen werden.

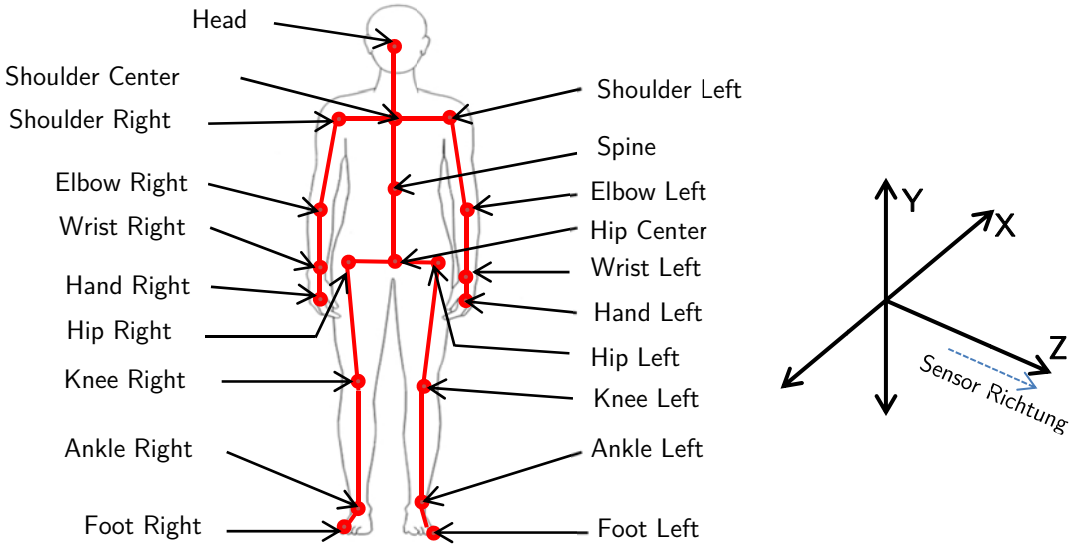


Abbildung 26: Skelett und Koordinatensystem des Kinect-Sensors [Mic11]

$$B_{spine} = P_{shoulder,center} - P_{spine} \tag{4.4}$$

$$B_{shoulder} = P_{shoulder,left} - P_{shoulder,right} \tag{4.5}$$

$$B_{shoulder,left} = P_{shoulder,left} - P_{shoulder,center} \tag{4.6}$$

$$B_{shoulder,right} = P_{shoulder,right} - P_{shoulder,center} \tag{4.7}$$

$$B_{arm,left} = P_{elbow,left} - P_{shoulder,left} \tag{4.8}$$

$$B_{arm,right} = P_{elbow,right} - P_{shoulder,right} \tag{4.9}$$

$$B_{forearm,left} = P_{wrist,left} - P_{elbow,left} \tag{4.10}$$

$$B_{forearm,right} = P_{wrist,right} - P_{elbow,right} \tag{4.11}$$

Dann wird ein Koordinatensystem festgelegt, dessen Ursprung in der Mitte zwischen beiden Schulterblättern liegt und das am Oberkörper entlang ausgerichtet ist:

$$x_{front} = B_{shoulder} \times B_{spine} \tag{4.12}$$

$$x_{right} = x_{front} \times B_{spine} \tag{4.13}$$

$$x_{down} = x_{front} \times x_{right} \tag{4.14}$$

Die Stellung der Roboterbasis ergibt sich aus dem Winkel zwischen dem Koordinatenvektor nach vorne und der Kameraachse:

$$\varphi_{basis} = \angle(\begin{bmatrix}0\\0\\1\end{bmatrix}, x_{front}) \tag{4.15}$$

Dabei gilt nach Merziger [Mer04]:

$$\angle(a, b) = arccos\frac{a \cdot b}{|a||b|} \tag{4.16}$$

Die Stellung der beiden Ellenbogengelenke ergibt sich direkt aus dem Winkel zwischen Arm und Unterarm:

$$\varphi_{elbow,left} = \angle(B_{arm,left}, B_{forearm,left}) \tag{4.17}$$

$$\varphi_{elbow,right} = \angle(B_{arm,right}, B_{forearm,right}) \tag{4.18}$$

Das menschliche Schultergelenk ist am Roboter durch zwei Gelenke nachgebildet worden. Der Winkel des Oberarms zum Oberkörper muss daher in zwei zueinander senkrecht stehende Winkel aufgeteilt werden.

Ein beliebiger Vektor v kann durch ein doppeltes Kreuzprodukt in eine Ebene mit dem Normalenvektor n projiziert werden:

$$v' = n \times (v \times n) \tag{4.19}$$

Das Schwenkgelenk am Roboterarm kontrolliert die Bewegungen in der Ebene mit dem Normalenvektor x_{front} während das Rotationsgelenk die Bewegung in der Ebene mit dem Normalenvektor x_{right} ermöglicht. Diese beiden Ebenen sind bedingt durch die Konstruktion zueinander senkrecht.

Daraus ergibt sich die Stellung der Schultergelenke wie folgt:

$$\varphi_{shoulder,pivot,n} = \angle(x_{front} \times \left(B_{arm,n} \times x_{front}\right), x_{down} \qquad n = left, right \tag{4.20}$$

$$\varphi_{shoulder,rotate,n} = \angle(x_{right} \times (B_{arm,n} \times x_{right}), x_{down} \qquad n = left, right \qquad (4.21)$$

Die Stellungen von Rotations- und Schwenkgelenk sind nicht voneinander unabhängig. Wenn der Winkel des Rotationsgelenkes größer als 90° ist, muss stattdessen der Supplementwinkel, wie in Abbildung 27 gezeigt, für das Schwenkgelenk benutzt werden.

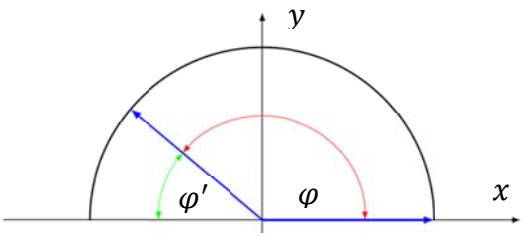


Abbildung 27: Winkel zwischen zwei Vektoren

$$\varphi'_{shoulder,pivot,n} = 180° - \varphi_{shoulder,pivot,n} \quad falls \quad \varphi_{shoulder,rotate,n} > 90° \qquad (4.22)$$

Die prinzipiellen Werte des Arcus Cosinus umfassen nur den positiven Bereich von [0, 180]. Daher muss zusätzlich geprüft werden, ob der tatsächliche Gelenkwinkel im dritten, oder vierten Quadranten liegt. Beim Schwenkgelenk der Schulter tritt dieser Fall ein, wenn der rechte Oberarm-Vektor nach links bzw. der linke nach rechts zeigt. Dies kann durch ein Skalarprodukt mit dem Koordinatenvektor x_{right} geprüft werden:

$$B'_{shoulder,pivot,left} \cdot x_{right} > 0 \qquad \text{bzw.} \qquad (4.23)$$

$$B'_{shoulder,pivot,left} \cdot x_{right} < 0 \qquad (4.24)$$

Für das Rotationsgelenk gilt das gleiche analog, nur dass hier das Punktprodukt mit dem Koordinatenvektor x_{front} genommen werden muss. Sollte eine der Gleichungen zutreffen, muss der ermittelte Winkel mit -1 multipliziert werden.

Der Rotationswinkel des Oberarms ergibt sich aus dem Winkel zwischen Unterarm und dem Koordinatenvektor x_{right}.

Wegen des seriellen Aufbaus des Roboterarms muss die Stellung des Schultergelenks berücksichtigt werden. Hierzu werden die beiden Vektoren in die gleiche Ebene, definiert durch die Stellung des Oberarms, projiziert.

$$B'_{front,n} = B_{arm,n} \times (B_{forearm,n} \times B_{arm,n)}) \qquad n = left, right \qquad (4.25)$$

$$x'_{right,n} = B_{arm,n} \times \left(x_{right} \times B_{arm,n)}\right) \qquad n = left, right \tag{4.26}$$

Aus diesen transformierten Vektoren wird der Rotationswinkel für den Unterarm berechnet.

$$\varphi_{arm,legft} = \angle(B'_{forearm,left}, x'_{right,left}) \tag{4.27}$$

$$\varphi_{arm,right} = \angle(B'_{forearm,right}, x'_{right,right}) \tag{4.28}$$

Bei einigen Gelenken des Roboters sind Nullpunkt und/oder Drehrichtung konstruktiv bedingt gegenüber der bisher erfolgten Berechnung abweichend. Bei diesen Gelenken muss daher noch eine konstante Verschiebung addiert bzw. die Drehrichtung invertiert werden:

$$\varphi'_{shoulder,pivot,left} = \varphi_{schoulder,pivot,left} - 90° \tag{4.29}$$

$$\varphi'_{shoulder,pivot,right} = \varphi_{schoulder,pivot,right} - 90° \tag{4.30}$$

$$\varphi'_{shoulder,rotate,left} = \varphi_{schoulder,rotate,left} + 90° \tag{4.31}$$

$$\varphi'_{shoulder,rotate,right} = \varphi_{schoulder,rotate,right} - 90° \tag{4.32}$$

$$\varphi'_{arm,left} = \varphi_{arm,left} - 90° \tag{4.33}$$

$$\varphi'_{arm,right} = \varphi_{arm,right} + 90° \tag{4.34}$$

Um das dynamische Verhalten des Roboters zu verbessern und das Ruckeln der Gelenke zu reduzieren, werden die berechneten Gelenkwinkel noch einmal mit einem einfachen gewichteten Mittelwert geglättet:

$$\varphi_i = \frac{3\varphi_{i,0} + 2\varphi_{i,-1} + \varphi_{i,-2}}{6} \tag{4.35}$$

4.1.2 Rückführung der Gelenkwinkel

Für eine Visualisierung der Stellung des Roboters in der GUI (graphical user interface) müssen die Gelenkwinkel mittels direkter Kinematik zurück in kartesische Koordinaten überführt werden.

Für den in dieser Arbeit entwickelten Roboter wurden mit Hilfe des Denavit-Hartenberg-Verfahrens (siehe Abschnitt 2.3) folgende Transformationsmatrizen bestimmt:

$$T_{base,left} = T(\gamma_{basis} + 90°, 90°, 0, 0) \cdot Trans(\frac{1}{2}) \tag{4.36}$$

$$T_{base,right} = T(\gamma_{basis} - 90°, -90°, 0, 0) \cdot Trans(\frac{1}{2}) \tag{4.42}$$

$$T_{shoulder,left} = T(\gamma_{shoulder,rotate,left}, -90°, 0, l) \cdot T(\gamma_{shoulder,pivot,left}, -90°, 0, 0) \tag{4.37}$$

$$T_{shoulder,right} = T(\gamma_{shoulder,rotate,right}, -90°, 0, -l) \cdot T(\gamma_{shoulder,pivot,right}, 90°, 0, 0) \tag{4.38}$$

$$T_{elbow,left} = T(\gamma_{arm,left} + 90°, -90°, 0, l) \tag{4.39}$$

$$T_{elbow,right} = T(\gamma_{arm,right} + 90°, 90°, 0, l) \tag{4.40}$$

$$T_{wrist,left} = T(\gamma_{elbow,left} - 90°, 90°, l, 0) \tag{4.41}$$

$$T_{wrist,right} = T(\gamma_{elbow,right} + 90°, 90°, l, 0) \tag{4.42}$$

Der Parameter l stellt dabei die Länge der Gliedmaßen des Roboters dar.

Der Parameter γ_i repräsentiert die jeweilige Gelenkstellung.

4.2 Schrittmotoransteuerung

Aus den berechneten Zielwerten für jedes Gelenk und den momentanen Gelenkstellungen werden die Steuersignale der Schrittmotoren an jedem Gelenk abgeleitet.

Zuerst werden dazu die Zielwerte mit einer Reihe von Grenzwerten verglichen. Dies sind zum einen Raumbegrenzungen, bei denen alle Zielwerte zusammen mit einem Satz von Grenzwerten verglichen werden und zum anderen Gelenkbegrenzungen, bei denen jeder Zielwert einzeln betrachtet wird.

Sei G ein Satz von minimalen und maximalen Grenzwerten für alle Gelenke und $\hat{G}$ die Untermenge der aktiven Grenzen, die sich durch ungleiche minimale und maximale Werte auszeichnen:

$$G = \{(\emptyset_{1,min}, \emptyset_{1,mx}), \dots, (\emptyset_{5,min}, \emptyset_{1,max})\} \tag{4.43}$$

$$G = \{G: \emptyset_{i,min} \neq \emptyset_{i,max}\} \tag{4.44}$$

Wenn der minimale Grenzwert kleiner als der maximale Grenzwert ist, stellt das Intervall $[\emptyset_{i,min}, \emptyset_{i,max}]$ den zulässigen Bereich für das Gelenk i dar. Wenn der minimale Grenzwert

der größere Wert ist, stellt das Intervall $]\emptyset_{i,max}, \emptyset_{i,min}[$ hingegen den verbotenen Bereich dar.

Die folgende Funktion M ist wahr, wenn der Winkel α außerhalb des zulässigen Bereichs $\emptyset$ ist:

$$M(\alpha, \emptyset) = \begin{cases} \alpha < \emptyset_{min} \vee \alpha > \emptyset_{max} & wenn \quad \emptyset_{min} < \emptyset_{max} \\ \alpha > \emptyset_{min} \wedge \alpha < \emptyset_{max} & andernfalls \end{cases} \tag{4.45}$$

Ein Satz von Grenzwerten ist verletzt, wenn alle aktiven Grenzen verletzt werden:

$$\{\hat{G}: M(\varphi_i, \emptyset_i) wahr\} = \hat{G} \tag{4.46}$$

Bei Verletzung eines Raumlimits wird eine konfigurierbare Aktion ausgelöst, dies hat aber keinen direkten Einfluss auf die Zielwerte sondern gibt eine Rückmeldung an den Benutzer zurück.

Die Gelenkbegrenzungen sind hingegen harte Limits, bei deren Überschreitung der Zielwert auf das näherliegende Limit beschränkt wird. Auch hier gilt bei einem kleineren minimalen Limit das Intervall $[\Pi_{i,min}, \Pi_{i,max}]$ als erlaubt und bei einem größeren minimalen Limit das Intervall $]\Pi_{i,min}, \Pi_{i,max}[$ als verboten.

Wenn der Zielwinkel außerhalb des erlaubten Intervalls liegt, wird er auf das näher liegende Limit begrenzt:

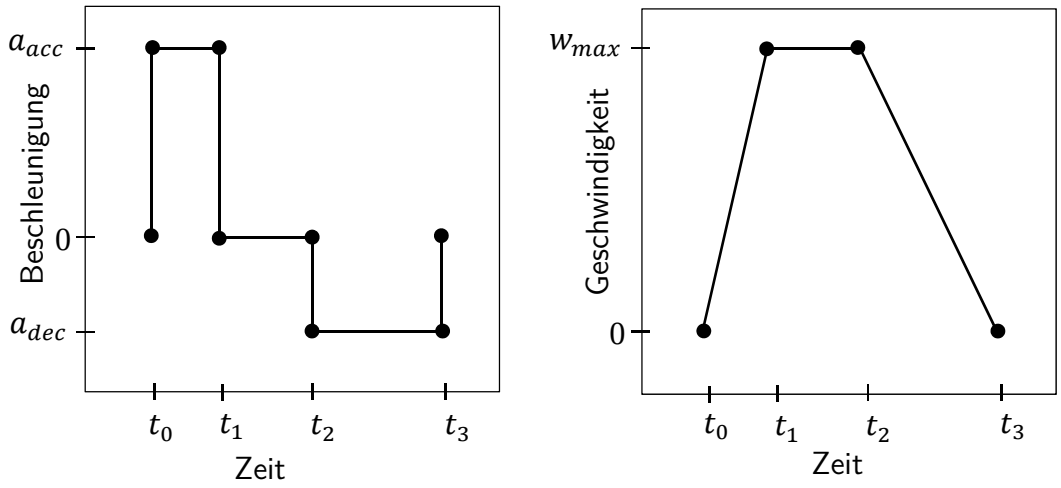
$$L(\alpha, \Pi) = \begin{cases} \Pi_{min} & wenn \quad |\alpha - \Pi_{min}| < |\alpha - \Pi_{max}| \\ \Pi_{max} & andernfalls \end{cases} \tag{4.47}$$

$$\widehat{\varphi_i} = \begin{cases} L(\varphi_i, \Pi_i) & wenn \quad \Pi_{i,min} < \Pi_{i,max}, \varphi_i \notin [\Pi_{i,min}, \Pi_{i,max}] \\ L(\varphi_i, \Pi_i) & wenn \quad \Pi_{i,min} > \Pi_{i,max}, \varphi_i \in]\Pi_{i,min}, \Pi_{i,max}[\\ \varphi_i & andernfalls \end{cases} \tag{4.48}$$

Anhand des angepassten Zielwinkels φ_i und der aktuellen Position wird für jede Achse die Zielgeschwindigkeit ermittelt. Die Geschwindigkeit wird anhand eines trapezförmigen Geschwindigkeitsprofils nach [Hei07] berechnet.

Bei dieser Art von Profil werden konstante Beschleunigung und konstante Verzögerung vorgegeben. Dadurch können die Beschleunigungs- und Verzögerungsphasen unabhängig voneinander angepasst werden, ohne dass besonderer Rechenaufwand betrieben werden muss. Nachteilig ist, dass sich die Beschleunigung sprunghaft ändert, was einen Ruck verur-

sacht. Die Beschleunigung muss daher soweit begrenzt werden, dass der Ruck innerhalb definierter zulässiger Grenzen bleibt.



(a) Beschleunigungsprofil (b) Geschwindigkeitsprofil

Abbildung 28: Trapezförmiges Geschwindigkeitsprofil

Abbildung 28 zeigt folgenden Ablauf: Ein still stehendes Gelenk wird zum Zeitpunkt t_0 mit der Beschleunigung a_{acc} beaufschlagt. Die Beschleunigung bleibt solange konstant, bis zum Zeitpunkt t_1 die maximale Geschwindigkeit w_{max} erreicht ist. Ab dem Zeitpunkt t_2 wird das Gelenk mit der Verzögerung a_{dec} abgebremst, bis es bei t_3 zum Stillstand kommt. Der Zeitpunkt t_2 muss dabei so gewählt werden, dass bei t_3 genau die Zielposition erreicht wird.

Zum Start wird aus dem Zielwinkel und die aktuelle Position γ_i des Gelenks der kürzeste Weg ε_i ermittelt:

$$\hat{\varepsilon}_i = \varphi_i - \gamma_i \tag{4.49}$$

$$\widetilde{\varepsilon_\iota} = \begin{cases} \hat{\varepsilon}_i - 360° \cdot sgn(\hat{\varepsilon}_i) & wenn \quad |\hat{\varepsilon}_i| > 180° \\ \hat{\varepsilon}_i & andernfalls \end{cases} \tag{4.50}$$

Die Bewegung darf nicht über die Gelenkbegrenzungen hinweg laufen. Falls dies passiert, muss die Bewegung stattdessen in der gegenteiligen Drehrichtung über den längeren Weg erfolgen:

$$\varepsilon_i = \begin{cases} \widetilde{\varepsilon_\iota} + 360° & wenn \quad \Pi_{i,min} - \gamma_i \; mod \; 360° > \widetilde{\varepsilon_\iota}, \widetilde{\varepsilon_\iota} < 0 \\ \widetilde{\varepsilon_\iota} - 360° & wenn \quad \Pi_{i,max} - \gamma_i \; mod \; 360° > \widetilde{\varepsilon_\iota}, \widetilde{\varepsilon_\iota} > 0 \\ \widetilde{\varepsilon_\iota} & andernfalls \end{cases} \tag{4.51}$$

Die zugehörige Drehrichtung d wird aus dem Weg ermittelt:

$$d_i = sgn(\varepsilon_i) \tag{4.52}$$

Aus dem zurückzulegenden Weg wird die zulässige Geschwindigkeit anhand des trapezförmigen Geschwindigkeitsprofils bestimmt. Die gerechnete Geschwindigkeit wird durch die maximal zulässige Verzögerung begrenzt:

$$a_{dec} = const. > 0 \tag{4.53}$$

Ausgehend von der angenommenen Geschwindigkeit w_i und der aktuellen Position γ_i werden die Werte nach der Zeit t bestimmt:

$$\Omega_i = \int_{\tau} -a_{dec}\, dt = w_i - a_{dec}\,\tau \tag{4.54}$$

$$\Gamma_i = \int_{\tau} \Omega_i\, dt = \gamma_i + w_i - \frac{1}{2} a_{dec}\,\tau^2 \tag{4.55}$$

Die Position Γ_i muss der Zielposition entsprechen:

$$\Gamma_i = \gamma_i + \varepsilon_i \Leftrightarrow \Gamma_i - \gamma_i = \varepsilon_i \Leftrightarrow\ \varepsilon_i = w_i\tau - \frac{1}{2} a_{dec}\,\tau^2 \tag{4.56}$$

Nach Erreichen der Zielposition wird die Geschwindigkeit gleich null gesetzt:

$$\Omega_i = 0 \Leftrightarrow\ 0 =\ w_i - a_{dec}\,\tau \Leftrightarrow\ \ \tau = \frac{w_i}{a_{dec}} \tag{4.57}$$

Hieraus ergibt sich die tatsächlich erlaubte Geschwindigkeit w_{dec}:

$$\varepsilon_i = \frac{{w_i}^2}{a_{dec}} - \frac{1}{2}\frac{{w_i}^2}{a_{dec}} = \frac{1}{2}\frac{{w_i}^2}{a_{dec}} \Leftrightarrow w_{i,dec} = \sqrt{2|\varepsilon_i| a_{dec}} \tag{4.58}$$

Die neue ermittelte Geschwindigkeit darf nicht höher als die maximale erlaubte Geschwindigkeit die mit Beschleunigung a_{acc} ansteigt. Aus der bisherigen Geschwindigkeit $w_{i,t-1}$ ergibt sich:

$$w_{i,acc} = w_{i,t-1} + \int a_{acc} dt \tag{4.59}$$

Die zulässige Geschwindigkeit ergibt sich somit aus den beiden eben berechneten Größen, sowie der absoluten Maximalgeschwindigkeit w_{max}:

$$w_i = \min(w_{i,dec}, w_{i,acc}, w_{max}) \tag{4.60}$$

Aus der Geschwindigkeit w_i wird abschließend die Schrittfrequenz f_i für jeden Schrittmotor ermittelt:

$$f_i = \frac{w_i}{360°} \tag{4.61}$$

4.3 Kommunikationsprotokoll mit dem Steuerungsmodul

Das Steuerungsmodul nimmt verschiedene Befehle über die Kommunikationsschnittstellen entgegen. Weiterhin werden regelmäßig die Positionen aller Achsen sowie der Zustand aller weiteren Sensoren zum Hauptprogramm gesendet. Hierzu ist ein leistungsfähiges und verzögerungsarmes Kommunikationsprotokoll notwendig, das sich für alle vorhandenen Schnittstellen gleichermaßen eignet.

4.3.1 Datenformate

Um die geforderten Eigenschaften zu erfüllen, werden Befehle und Informationen in Form von Datenpaketen mit fester Länge übertragen (mit Ausnahme der Befehlsmodul-Information). Durch die feste Länge kann die Übertragungsdauer auf allen Kanälen leicht bestimmt werden. Weiterhin ist es nicht notwendig, teilweise übertragene Daten zwischenzuspeichern, was Verzögerungen reduziert und die Implementierung vereinfacht.

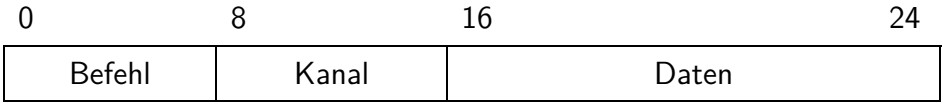


Abbildung 29: Allgemeines Format des Datenpakets

Abbildung 29 zeigt den Aufbau des generischen Datenpakets. Es besteht aus einem Byte mit der Befehlskennung, gefolgt von einem Byte mit dem Kanal und zwei befehlsspezifischen Datenbytes. Bei allen Feldern wird das niederstwertige Bit zuerst übertragen ('little-endian' nach [Coh81]).

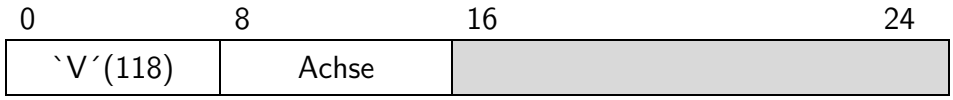
Auf einer Schnittstelle eingehende Befehle werden in der Eingangsreihenfolge abgearbeitet. Die Bearbeitungsreihenfolge von Paketen, die über verschiedene Schnittstellen eintreffen, ist nicht festgelegt. Es muss bei Befehlen, die ein Antwortpaket erzeugen, nicht auf die vollständige Abarbeitung gewartet werden, bevor ein weiterer Befehl gesendet werden darf. Es kann vorkommen, dass die Antwort nicht unmittelbar auf einen Befehl folgt, sondern dazwischen z. B. noch Pakete über geänderte Achspositionen folgen.

Bei einem unbekannten Befehl, einer fehlerhaften Übertragung oder einem anderen Verarbeitungsfehler wird ein Fehler-Paket gesendet, bei dem alle Bits auf 1 gesetzt sind.

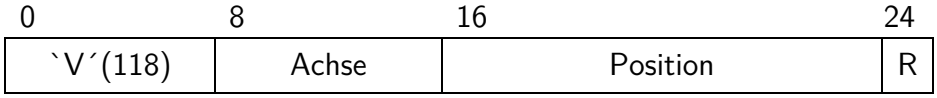
Es folgt eine beispielhafte Auflistung von einem Teil der Befehle und Antworten:

Aktuelle Achsposition:

Format der Abfrage:



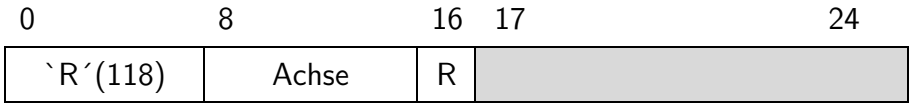
Format der Antwort :



R ist dabei die Variable für die Achsen Referenzierung.

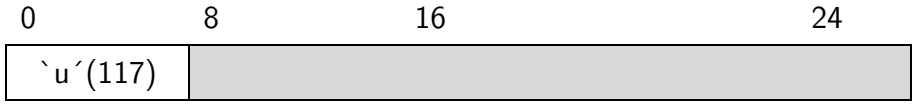
Achse auf referenziert oder unreferenziert setzen:

Format des Befehls:

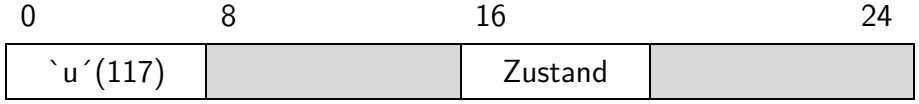


Logische Drehrichtung:

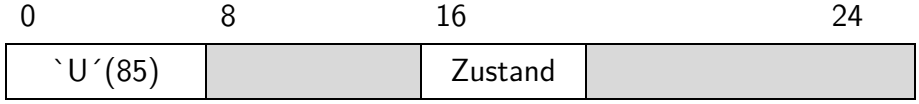
Format der Abfrage:



Format der Antwort:

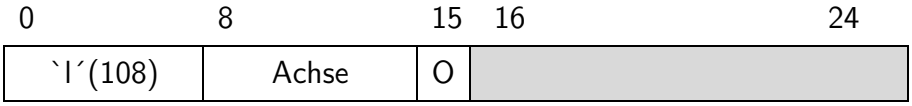


Zum Ändern der Drehrichtung wird folgender Befehl gesendet:

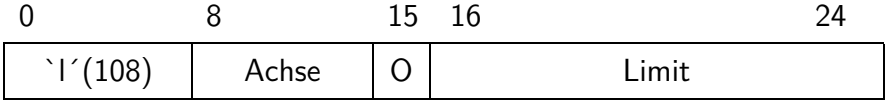


Positionslimit:

Format der Abfrage:

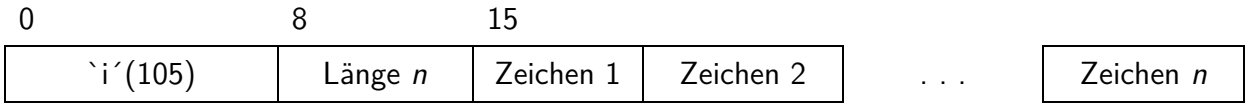


Format der Antwort:



Modul-Information:

Die Antwort auf die Anfrage der Modul-Information folgt nicht dem allgemeinen Paketformat. Es besteht aus dem Befehlsbyte 'i' (105) und einem Längenbyte. Danach folgt eine Zeichenkette der angegebenen Länge. Ein Null-Byte wie bei C-Strings wird nicht übertragen. Das Format aus Länge plus Daten wird auch als PASCAL-String bezeichnet [Cla11].



4.3.2 Umsetzung

Alle auf den verschiedenen Schnittstellen eingehenden Datenbytes werden pro Schnittstelle in einem Eingangspuffer zwischengespeichert. Für die seriellen Schnittstellen wird hierzu der jeweilige UART-Interrupt (Universal Asynchronous Receiver Transmitter) verwendet, was eine konstante Abfrage auf eingegangene Daten in der Hauptschleife überflüssig macht und Rechenzeit für wichtige Aufgaben freigibt.

Hierzu wird der UART-Baustein so konfiguriert, dass ein Interrupt ausgelöst wird, wenn mindestens vier Byte entsprechend der Paketgröße im Hardware-Empfangsregister vorhanden sind oder seit einiger Zeit keine weiteren Bytes empfangen wurden. In der Behandlungsroutine werden die Bytes nacheinander aus der Hardware ausgelesen und im Eingangspuffer angelegt. Der Puffer ist als sogenannter Ringpuffer nach [Sed05] ausgelegt (siehe Abschnitt 2.11).

Wenn in einem Eingangspuffer mindestens ein vollständiges Paket vorhanden ist, werden die Pakete im nächsten Durchlauf der Hauptschleife ausgewertet. Dazu werden die einzelnen

Bestandteile entsprechend dem definierten Paketformat aus dem Puffer extrahiert. Je nach Befehlsbyte wird dann die entsprechende Verarbeitung angestoßen. Ein eventuelles Antwortpaket wird über die Schnittstelle gesendet, über die der Befehl empfangen wurde.

Die gesendeten Daten werden ohne einen Ausgangspuffer direkt an die Hardware übergeben. Diese hat selbst einen kleinen Sendepuffer, der mit 16 Bytes (siehe [Nxp11]) ausreichend Platz für ein vollständiges Paket bereitstellt. Daher kann ein Antwortpaket fast ohne Verzögerung direkt an die Hardware weitergeleitet werden.

Die RS-232 kompatiblen, seriell angeschlossenen Steuermodule werden mit 115 200 Baud, acht Datenbits, einem Stoppbit, keinem Paritätsbit und Hardware-Flusskontrolle betrieben. Die virtuellen seriellen Anschlüsse über USB und das XPort-Modul werden mit 230 000 Baud ohne Flusskontrolle betrieben.

4.4 Controller Firmware

Aufgabe der Firmware des Steuerungsmoduls ist es, die Eingaben der angeschlossenen Sensoren auszuwerten und in eine für die Weiterverarbeitung sinnvolle Form zu bringen. Anhand der Eingabedaten wird für jeden Schrittmotor eine Zielposition bestimmt und der Motor entsprechend angesteuert. Weiterhin werden Befehle der PC-Steuersoftware umgesetzt und geänderte Sensorwerte zurückgesendet.

Die Firmware wurde mit Entwicklungsumgebung LPCXpresso™ von Code Red Technologies in der Version 4.2.3 erstellt. Als Übersetzer (Compiler) wurde die GNU Compiler Collection (GCC) V4.5.1 genutzt. Zum Übertragen der Firmware in den Speicher des Mikrocontrollers kam Flash Magic von Embedded Systems Academy, zum Einsatz.

Im folgenden Abschnitt werden die Programmstruktur, die Auswertung der Sensoreingaben, die Ansteuerung der Schrittmotoren sowie das Kommunikationsprotokoll zur PC-Steuersoftware genauer beschrieben.

4.4.1 Programmstruktur

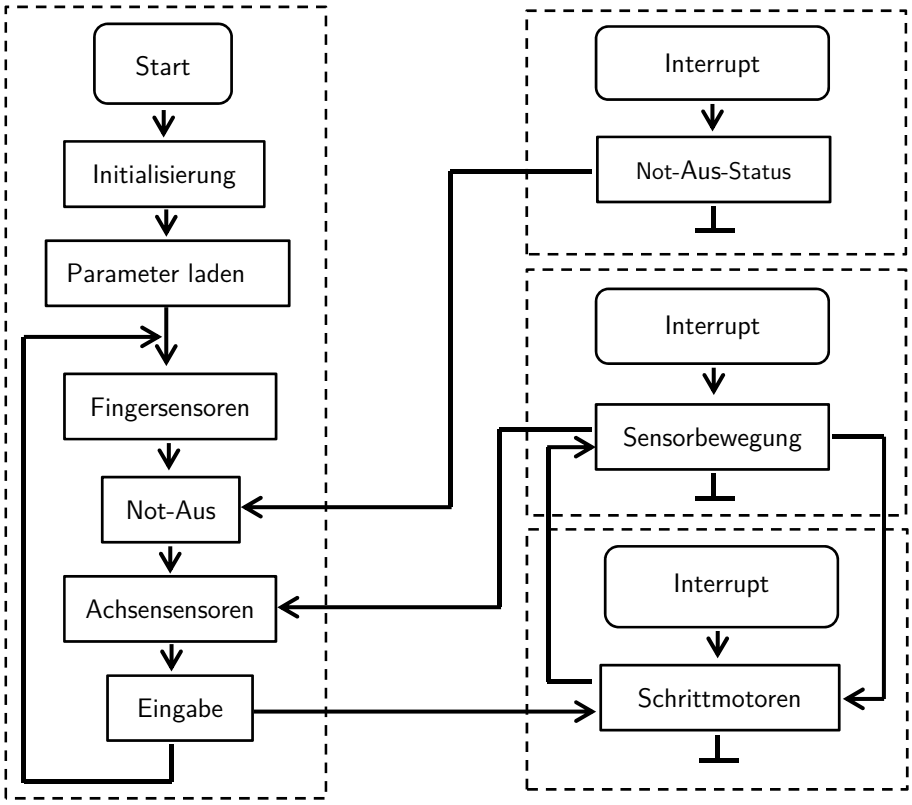


Abbildung 30: Programmstruktur der RoboControl Firmware

Die Struktur der Firmware ist in Abbildung 30 dargestellt. Sie besteht aus einer endlos ablaufenden Hauptschleife sowie einer Reihe weiterer durch Interrupts ausgelöster Prozesse.

Vor dem Eintritt in die Hauptschleife müssen alle Hardwarekomponenten initialisiert werden. Im ersten Schritt werden die einzelnen Funktionseinheiten den externen Pins des Mikroprozessors LPC 1769 zugeordnet. Dann können die Funktionseinheiten eingestellt und aktiviert werden. Das Steuerungsmodul benötigt die SPI-Schnittstelle (Serial Peripheral Interface) zum Ansprechen weiterer Bauelemente, wie Timer und PWM-Blöcke für die Schrittmotorsteuerung, UARTs für die seriellen Schnittstellen nach außen sowie einige allgemeine Digital-I/Os.

Anschließend werden die externen Hardware-Bausteine konfiguriert. Hier werden die Pins der drei MCP 23S18 (Multi Chip Package) jeweils entsprechend als Eingang oder Ausgang gesetzt, Pull-up-Widerstände aktiviert und die externen Interrupts freigeschaltet.

Wenn die Hardware-Initialisierung abgeschlossen ist, werden alle Variablen auf definierte Startwerte gesetzt. Aus dem Flash-Speicher des Mikrocontrollers werden die für die Sensorauswertung und Schrittmotorsteuerung benötigten Parameter wiederhergestellt.

Wenn die Initialisierung erfolgreich abgeschlossen ist, wird die Hauptschleife gestartet. Die Schleife wird kontinuierlich abgearbeitet und kann nur noch durch einen Reset verlassen werden (Abbildung 30).

Als erstes werden in jedem Durchgang die Fingersensoren abgefragt und bei einer Änderung des Zustands eine Nachricht an die PC-Software gesendet. Die Not-Aus-Signale der Schrittmotor-Boxen und die Positionen der Achsen werden gleichermaßen auf Änderungen geprüft und gemeldet. Abschließend werden alle von der PC-Software eingegangenen Befehle verarbeitet und beantwortet.

Ein Teil der Signalverarbeitung wird mit Hilfe von Interrupts zeit- oder ereignisbasiert unabhängig zur Hauptschleife ausgeführt. Änderungen des Schaltzustandes der Not-Aus-Signale lösen eine Interrupt-Routine aus, in der der neuen Zustand ausgelesen und in eine Variable an die Hauptschleife übergeben wird. Auch ein Signal eines der Inkrementalgeber der Achsen löst einen Interrupt aus, in dem die Achsenposition aktualisiert wird.

Nicht ereignisbasiert, sondern in einem festen Takt wird die Steuerroutine für die Schrittmotoren ausgeführt.

4.4.2 Auswertung der Winkelsensoren

Die Firmware muss die Signale einer Reihe von Sensoren auswerten. Zum einen sind dies die Fingersensoren und die Not-Aus-Signale der Schrittmotor-Boxen, zum anderen die in den Roboterachsen integrierten Winkelsensoren.

Die Winkelsensoren in den Roboterachsen liefern zwei Signale: Ein Quadratursignal für die rotatorische Bewegung und ein Schaltsignal für die Nullstellung.

Abbildung 31 stellt ein quadraturcodiertes Signal dar. Dieses Signal zeigt ausschließlich an, dass die Achse einen Schritt vorwärts oder rückwärts gemacht hat, aber nicht die absolute Position [Hop03].

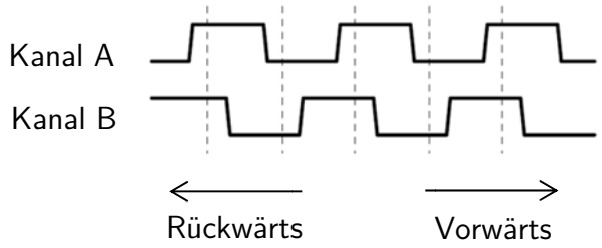


Abbildung 31: Quadratursignal eines Inkrementalgebers

Die Erkennung der Bewegungsrichtung geschieht anhand des Phasenversatzes zwischen den beiden Kanälen. Bei der Bewegung um einen Schritt vorwärts wechselt die Polarität des Kanals A *nach* der des Kanals B, bei einer Bewegung rückwärts *vor* Kanal B.

Bei jeder Änderung von Kanal A wird eine Interrupt-Routine aufgerufen. Diese liest die Signale der beiden Kanäle mittels SPI vom zugeordneten MCP-Chip ein.

Durch Vergleichen der beiden Kanäle wird die Richtung bestimmt, welche pro Achse konfigurierbar invertiert werden kann. Je nach Richtung wird die von der Firmware geführte Position um Eins erhöht oder vermindert. Wenn der neue Wert über dem für 360° zugeordneten Wert liegt, wird er auf null gesetzt; wenn er Null unterschreitet auf den Maximalwert. Zusätzlich wird auch noch die aktuelle Drehrichtung in einer Variablen gespeichert. Diese wird bei der Detektierung der Nullstellung benutzt. Hierzu wird das Signal des Nullschalters für alle unreferenzierten Achsen bei Änderungen mit einem Interrupt eingelesen. Die Achsen werden je nach aktueller Drehrichtung bei einer steigenden oder fallenden Flanke als referenziert markiert. Gleichzeitig wird die aktuelle Position der Achse auf null gesetzt.

4.4.3 Wiederholfrequenz der Schrittmotorpulse

Die Geschwindigkeit und Richtung der Schrittmotoren wird anhand des in Abschnitt 4.2 beschriebenen Algorithmus gesteuert. Es werden nur referenzierte und aktivierte Achsen angesteuert.

Die Berechnung wird mittels des Repetitive Interrupt Timer des Mikroprozessor LPC 1769 (siehe [Nxp10]) alle 0,5 ms angestoßen. Um diese Wiederholrate erreichen zu können, muss die Berechnung möglichst effizient ausgeführt werden. Daher ist eine Verwendung von Fließkomma-Arithmetik ausgeschlossen, da der Mikrocontroller dafür keine entsprechenden Hardware-Recheneinheiten besitzt. Die Berechnungen werden stattdessen ausschließlich in Festkomma-Arithmetik mit Ganzzahlen durchgeführt.

Eine Achse wird nicht gestoppt, wenn die exakte Zielposition erreicht ist, sondern schon wenn der Abstand unter einem definierten Schwellenwert liegt. Dies vermeidet kleine, schnelle Schwingungen um die Zielposition, die sonst durch die Trägheit der mechanischen Komponenten verursacht würden.

Zur Lösung der Gleichung (4.54) und zur Bestimmung der maximalen Geschwindigkeit in Abhängigkeit von der maximalen Verzögerung, muss eine Wurzel berechnet werden. Eine Wurzelberechnung ist eine aufwändige Operation, für welche der Mikroprozessor LPC 1700-Serie keine Hardwareunterstützung bietet. Daher wurde ein effizienter Algorithmus nach [Nie89] basierend auf einer partiellen Nachschlagetabelle implementiert.

Eine vollständige Tabelle für Zahlen mit 32-Bit Breite würde $2 \cdot 2^{32}$ Bytes Speicher belegen, was die Kapazität eines Mikrocontrollers deutlich überfordern würde. Durch die Zerlegung des Eingangswerts in zwei Komponenten kann der Speicherbedarf massiv gesenkt werden.

$$(a \cdot b)^{\frac{1}{2}} = a^{\frac{1}{2}} \cdot b^{\frac{1}{2}} \tag{4.62}$$

Eine beliebige Binärzahl X kann in wie folgt in zwei Komponenten zerlegt werden:

$$X = AB \tag{4.63}$$

$$A = 2^N \tag{4.64}$$

$$1 \leq B \leq 2 \tag{4.65}$$

Der Wert N kann einfach ermittelt werden, da er direkt der (1-basierten) Nummer des höchstwertigen 1-Bit entspricht. Das Ergebnis von $\sqrt{2^N}$ kann für alle 33 möglichen Werte von N aus einer vorberechneten Tabelle entnommen werden. Da nur Ganzzahlen verfügbar sind, müssen die Tabellenwerte gerundet werden. Um sicherzustellen, dass die berechnete Geschwindigkeit niemals größer als der exakte Wert bei unendlicher Genauigkeit wird, wird ausschließlich abgerundet.

$$T_1(0) = 0 \tag{4.66}$$

$$T_1(n) = floor\left(\sqrt{2^n}\right),\ n = 1 \ldots 32 \tag{4.67}$$

Die Berechnung von $\sqrt{B}$ wird ebenfalls über eine Tabelle mit 32 Einträgen vorgenommen, was einen guten Kompromiss aus Genauigkeit und Speicherbedarf darstellt. Die Werte werden als Festkommazahlen mit 15 Nachkommabits in der Tabelle abgelegt:

$$T_2(n) = floor\left(\sqrt{1 + \frac{n}{32} \cdot 2^{15}}\right),\ n = 0 \ldots 31 \tag{4.68}$$

Der Wert B entspricht den sechs (lg 32 + 1) signifikanten Bits von X bis einschließlich dem höchstwertigen 1-Bit. Das approximierte Ergebnis ist damit bei Berücksichtigung des Nachkommateils:

$$\sqrt{X} \approx \frac{T_1(N) \cdot T_2(B)}{2^{15}} \tag{4.69}$$

Da das Kommunikationsprotokoll nur maximal eine 16-Bit-Zahl für den Parameter der maximalen Verzögerung erlaubt, wurde noch ein konstanter Faktor eingefügt, um den Wertebereich nach oben hin zu erweitern.

Zur Berechnung der maximalen Geschwindigkeit anhand der erlaubten Beschleunigung muss eine Ableitung berechnet werden. Da die Zeitdauer zwischen zwei Berechnungen konstant festgelegt wurde, kann die Ableitung einfach durch einen Differenzquotienten ersetzt werden.

Die maximale Absolutgeschwindigkeit jeder Achse wird mit einem globalen Geschwindigkeitsfaktor multipliziert, was eine einfache Skalierung der Geschwindigkeit erlaubt, ohne die Parameter für jede Achse einzeln ändern zu müssen.

Aus der so berechneten Geschwindigkeit wird die Wiederholfrequenz f der Schrittmotorpulse bestimmt. Zur Einstellung der Timer-Module des Mikroprozessors LPC wird die Frequenz dann in eine Periodendauer p umgerechnet. Alle Timer zählen mit einem Takt von $1\,MHz$, die Pulsdauer in Timer-Inkrementen entspricht also:

$$p = \frac{10^6}{f} \tag{4.70}$$

Die Periodendauer kann abschließend in das entsprechende Register des Timers geschrieben werden. Wenn die neue Periodendauer kürzer als die alte Dauer ist, muss der Timer dafür kurz gestoppt werden. Würde der Timer weiterlaufen, könnte es passieren, dass der interne Zähler des Timers größer als die Periodendauer wird, was eine Fehlfunktion auslösen würde.

4.5 PC Software

Die PC Software besteht aus zwei Softwarepaketen, das erste Paket ist die Robotersteuerung und das zweite ist die Visualisierung und 3D Rekonstruktion der Roboterumgebung in einem virtuellen 3D Raum.

4.5.1 PC Software für die Robotersteuerung

Die Software besteht aus zwei Komponenten, dem Kern (Kernel), der für die Kommunikation sowie die Auswertung der Kinect Kamera zuständig ist und der grafischen Benutzeroberfläche (GUI).

Die GUI dient zur Visualisierung des Zustands des Roboters und zur Konfiguration der Steuerungsmodule.

Die Software berechnet die Gelenkstellungen, die vom Steuerungsmodul in Bewegungen umgesetzt werden. Dabei gibt sie dem Bediener visuelle Rückmeldung über die erkannten Kameradaten und den tatsächlichen Zustand des Roboters. Sie ermöglicht die Definition von verbotenen Raumbereichen, bei deren Verletzung der Bediener mit einer konfigurierbaren Aktion benachrichtigt wird.

Weiterhin leitet sie den Zustand der Fingersensoren an den Händen des Roboters an einen Feedbackhandschuh weiter und stellt Eingabedialoge für alle konfigurierbaren Einstellungen der Hardwaremodule bereit.

4.5.1.1 Programmstruktur

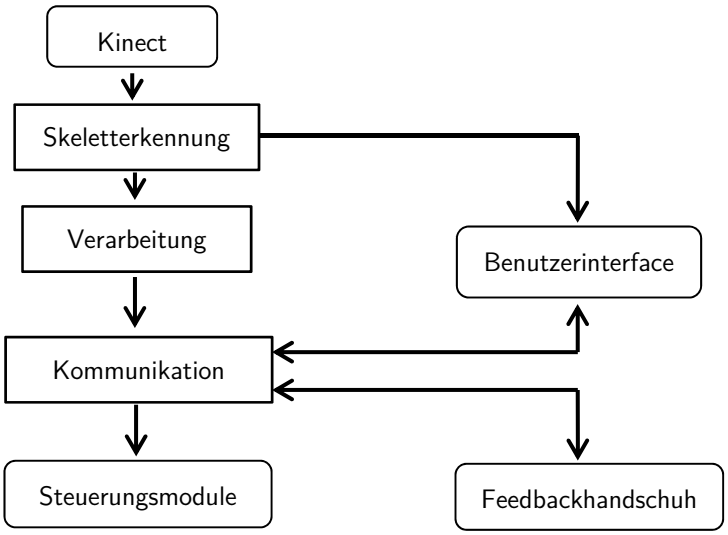


Abbildung 32: Vereinfachte Programmstruktur der PC-Software

Eine vereinfachte Programmstruktur der PC-Software ist in Abbildung 32 dargestellt. Das Programm besteht aus einem zentralen Verarbeitungsblock, um den sich die anderen Blöcke gruppieren. Dies sind die Erkennung des Skeletts sowie die Berechnung der Roboterposition, die Kommunikation mit den Steuerungsmodulen und die Ausgabe an das Modul mit dem Feedbackhandschuh. Ein Teil der Informationen wird auch noch an das Benutzerinterface weitergeleitet.

Alle Blöcke werden nach [Dab02] in eigenen Threads (Aktivitätsträger) ausgeführt und kommunizieren untereinander über Nachrichten. Hierdurch wird verhindert, dass eine länger andauernde Berechnung oder Datenübertragung in einem Block alle anderen Blöcke zeitweilig blockiert.

Im Skelett-Thread werden alle Berechnungen vom Bild der Kinect-Kamera bis zur Roboterposition ausgeführt. Die berechneten Werte werden an die Benutzerschnittstelle und den Verarbeitungs-Thread weitergeleitet.

Der Verarbeitungs-Thread setzt alle Informationen zusammen. Hier werden die Positionsdaten den Ausgangskanälen der Steuerungsmodule zugeordnet, die verbotenen Bereiche überwacht und die Vibrationsmotoren des Feedbackhandschuhs angesteuert. Je nachdem, welche Kanäle vom Bediener freigegeben wurden, werden die Daten an die Kommunikations-Threads weitergeleitet.

Dieser leitet die Daten über die serielle Schnittstelle weiter an das Steuerungsmodul. Gleichzeitig empfängt die von den Gelenksensoren gelieferte aktuelle Gelenkstellung, den Zustand der Handsensoren und der Not-Aus-Schalter. Die eingehenden Daten werden zurück an den Verarbeitungs-Thread geleitet und an die Benutzerschnittstelle zur Visualisierung weitergegeben. Im Gegenzug werden vom Benutzer geänderte Einstellungen des Steuerungsmoduls entgegengenommen und an das Modul gesendet.

4.5.1.2 Umsetzung

Die Software wurde in C++ nach ISO/IEC C++ 2003 geschrieben und mit Microsoft Visual Studio 2010 kompiliert. Die Benutzeroberfläche wurde unter direkter Benutzung der nativen Windows-Funktionen implementiert. Zusätzlich zum Software Development Kit (SDK) für Windows wird das Kinect SDK benötigt. Es wurde das SDK in der Version 1.5 benutzt.

Auswertung der Kinect-Kamera

Der Skelett-Thread liest die Daten der Kinect-Kameras aus und berechnet daraus für jedes Gelenk des Roboters den gewünschten Zielwinkel.

Die Software-Bibliothek der Kinect-Kamera enthält Funktionen, die aus den Farb- und Tiefenbildern einen menschlichen Körper erkennen und die Positionen aller seiner Gliedmaßen im Raum berechnen.

Die so ermittelten Positionen des Körpers sind noch zu stark mit Rauschen und Störsignalen behaftet, als dass sie direkt weiter benutzt werden können. Daher müssen diese zuerst nach dem in Abschnitt 4.1.1 beschriebenen Algorithmus geglättet werden.

Aus den geglätteten Positionen werden die Stellungen aller Gelenke ermittelt. Das grundsätzliche Vorgehen wurde in Abschnitt 4.1.1 dargestellt.

Zur Vereinfachung der Programmierung wurde eine C++-Klasse als Repräsentation eines Vektors erstellt. Diese wurde mit benutzerdefinierten Operatoren für das Kreuzprodukt und das Punktprodukt ausgestattet, was kompakten und übersichtlichen Programmiercode ermöglicht.

Bei der Umsetzung ergeben sich folgende Herausforderungen: Ein Computer führt Berechnungen mit Fließkommazahlen nur mit endlicher Genauigkeit durch. Daher wird z. B. ein

Test auf exakte Gleichheit mit einem Wert fast nie zutreffen, da es immer zu leichten Abweichungen kommt. Solche Tests müssen daher so geändert werden, dass geprüft wird, ob der Unterschied zwischen den beiden Werte kleiner als ein Schwellenwert ist.

Aufgrund dessen ist im Programmcode der Vergleich *arm_left * down* $<$ *-0.000001* eingefügt. Die Umkehrung des Winkels darf erst erfolgen, wenn das Punktprodukt tatsächlich kleiner als Null ist. Durch Rechenungenauigkeiten kann es aber passieren, dass ein Vergleich < 0.0 wahr wäre, obwohl das exakte Resultat des Punktprodukts noch positiv ist.

Bei der Berechnung des Winkels zwischen zwei Vektoren (siehe Gleichung. 4.16) wird durch die Länge der beiden Vektoren dividiert. Wenn dieser Divisor nahe oder gleich Null ist, ist das von der CPU berechnete Ergebnis nicht plausibel. Dieser Fall muss vorher getestet werden und direkt der erwartete Wert von Null Grad zurückgegeben werden. Da der Arcuscosinus eine mehrwertige Funktion ist, muss zusätzlich sichergestellt werden, dass die verwendete Implementierung Resultate im erwarteten Wertebereich zurückliefert. Ist dies nicht der Fall, wird eine passende Korrektur hinzugefügt werden.

Verarbeitung und Visualisierung

Der Verarbeitungs-Thread nimmt die Soll- und Ist-Werte aller Gelenke entgegen. Die Ist-Werte werden mit einer Liste von verbotenen Bereichen verglichen. Dies geschieht nach dem am Abschnitt 4.2 beschriebenen Algorithmus. Bei einem Treffer wird je nach Enstellung entweder die Bewegung des Roboters gestoppt oder über den Kommunikations-Thread eines der Relais auf dem Steuerungsmodul umgeschaltet der einen audiovisuellen Alarm auslöst.

Die eingehenden Soll-Werte werden an den entsprechenden Kommunikations-Thread weitergeleitet, wenn die Bewegung des Gelenks freigegeben worden ist und die automatische Kamerasteuerung aktiv ist.

Der Verarbeitungs-Thread bereitet alle Daten für die Visualisierung auf der Benutzeroberfläche vor. Hierzu müssen die Ist-Gelenkwinkel zurück in kartesische Positionen gewandelt werden. Dies geschieht anhand des in Abschnitt 4.1.2 beschriebenen Algorithmus.

Für jedes Gelenk wird mit dem Gelenkwinkel die zugehörige Denavit-Hartenberg-Transformation bestimmt. Die Koordinaten der Gelenke können nacheinander durch Anwenden der einzelnen Transformationen auf den Ausgangspunkt berechnet werden.

Sowohl die so berechnete Stellung des Roboters, als auch die Bilddaten der Kinect-Kamera werden im Benutzerinterface angezeigt. Hierzu wird die Direct2D-Schnittstelle benutzt, die das hardwarebeschleunigte Zeichnen von zweidimensionalen Bilder und Geometrien ermöglicht. Damit kann eine flüssige Darstellung der Informationen bei geringer Rechenlast realisiert werden.

Benutzeroberfläche

Die Benutzeroberfläche des Programms gliedert sich in das immer sichtbare Hauptfenster und ein zuschaltbares Fenster mit genauen Details über den Roboterzustand sowie verschiedene Konfigurationsdialoge.

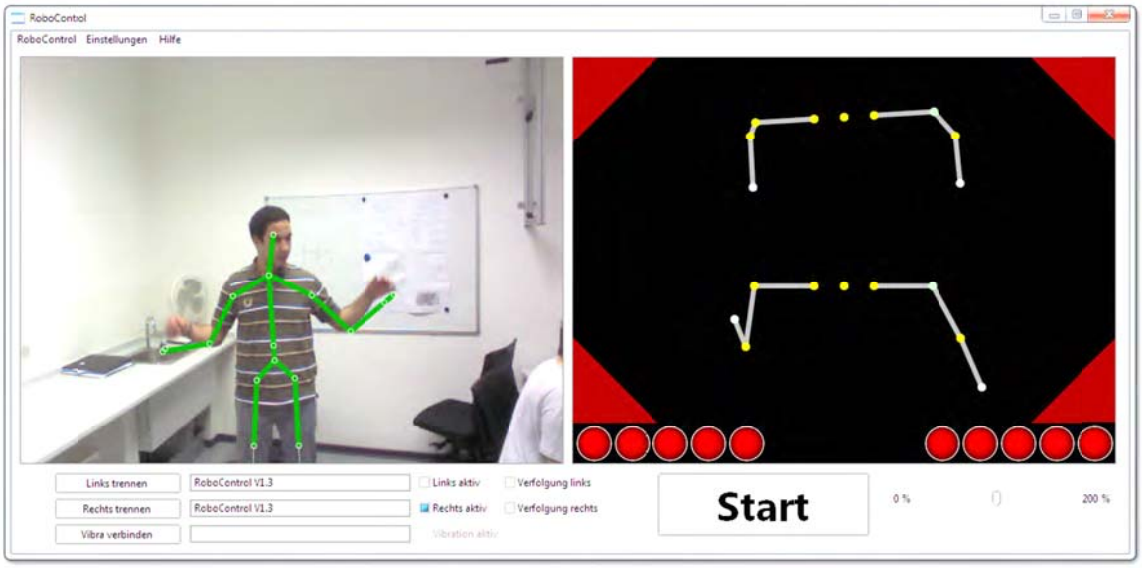


Abbildung 33: Screenshot des Hauptfensters

Das in Abbildung 33 gezeigte Hauptfenster besteht aus zwei Bereichen: Im oberen Bereich befinden sich zwei graphische Darstellungen, die den aktuellen Zustand des Systems visualisieren. Im unteren Bereich finden sich verschiedene Bedienelemente. Der obere Bereich zeigt auf der linken Seite das von der Kinect-Kamera eingefangene Bild an. Über das Bild wird eine Abbildung des erkannten Skeletts gelegt. Durch diese Kombination kann der Bediener sofort erkennen, ob die Skeletterkennung fehlerfrei abläuft.

Auf der rechten Seite wird der Zustand des Roboters dargestellt. Zum einen ist dies die Lage des Roboters im Raum, sowohl in einer Frontsicht als auch in einer Draufsicht, weiterhin werden gedrückte Not-Aus-Schalter durch ein rotes Dreieck in den Ecken des Bereichs angezeigt. Am unteren Rand wird der Zustand der Handsensoren visualisiert.

Über das Programmmenü kann das in Abbildung 34 gezeigte Fenster mit mehr Details über die einzelnen Gelenke des Roboters geöffnet werden.

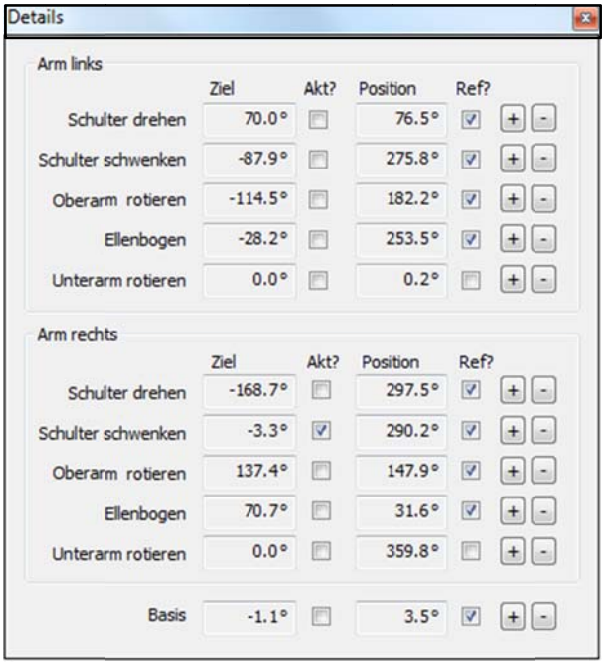


Abbildung 34: Screenshot des Detail-Fensters

Für jedes Gelenk gibt es eine Zeile mit der Bezeichnung der Achse. Des Weiteren kann die aktuelle Position des Gelenks sowie die aus dem Kamerabild ermittelte Zielposition abgelesen werden. Ein Kontrollkästchen zeigt an, ob das Gelenk referenziert ist. Hiermit kann die Referenzierung auch manuell gesetzt oder zurückgenommen werden.

Im Detail-Fenster kann jedes Gelenk einzeln mit einem Kontrollkästchen freigegeben und damit die im Hauptfenster einstellbare generelle Freigabe präzisiert werden. Über zwei Schaltflächen kann jedes Gelenk manuell vorwärts und rückwärts bewegt werden. Die manuelle Bewegung wird deaktiviert, wenn die automatische Steuerung aktiv ist.

Das Programm hat eine Reihe von weiteren Konfigurationsoptionen, die alle über das Programmmenü erreichbar sind.

Es steht ein Dialog zur Auswahl der benutzten Kinect-Kamera zur Verfügung. Dort kann auch der Kippwinkel der Kamera adaptiert werden. Die zuletzt benutzte Kamerakonfiguration wird abgespeichert und beim nächsten Programmstart wieder ausgewählt.

In einem weiteren Dialog können die seriellen COM-Ports für die beiden Steuerungsmodule und die Feedbackhandschuhe ausgewählt und konfiguriert werden. Auch diese Einstellungen werden bei Programmstart wiederhergestellt.

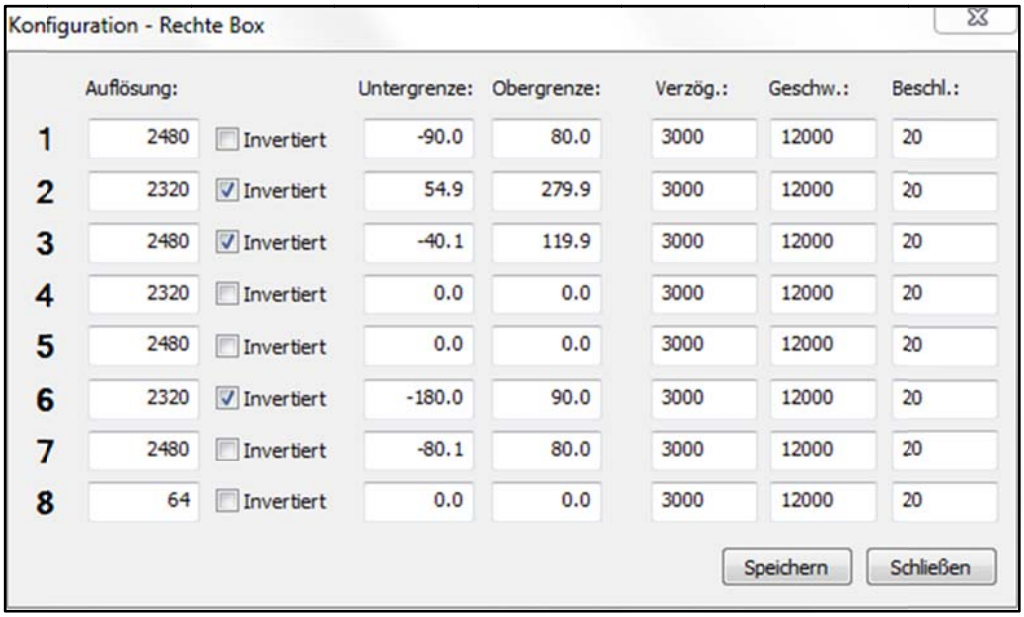


Abbildung 35: Screenshot des Konfigurationsdialogs für das Steuerungsmodul

Mit dem in Abbildung 35 gezeigten Dialog können die Steuerungsmodule konfiguriert werden. Für jeden Kanal des Moduls kann hier die Anzahl der Pulse des Inkrementalgebers für eine vollständige Umdrehung angegeben werden. Es besteht auch die Möglichkeit, die Drehrichtung des Inkrementalgebers gegenüber dem Drehsinn des zugehörigen Schrittmotors zu invertieren. Für die Steuerung der Schrittmotoren besteht die Möglichkeit, den erlaubten Drehbereich einzuschränken. Auch die Regelungsparameter für maximal erlaubte Beschleunigung, Verzögerung und Geschwindigkeit können verändert werden.

Geänderte Werte werden sofort an das Steuermodul übertragen und sind bis zu einem Reset des Moduls aktiv. Mit der Schaltfläche „Speichern“ können die Werte dauerhaft im Speicher des Moduls abgelegt werden.

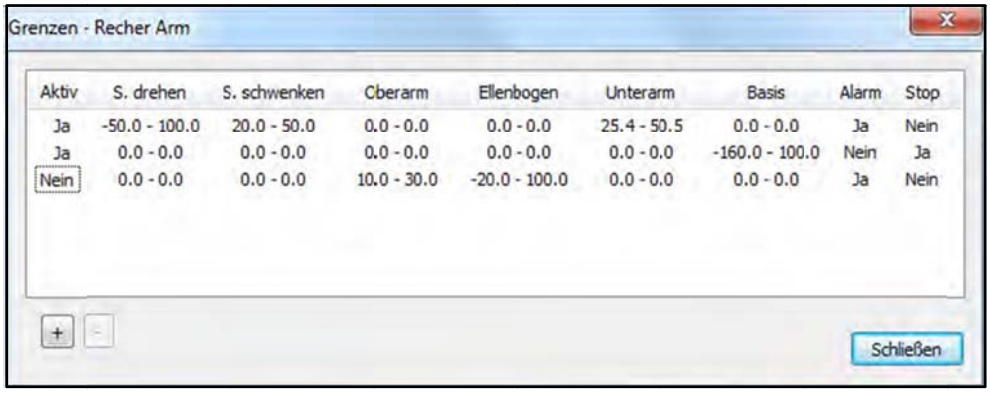


Abbildung 36: Screenshot der Liste der verbotenen Bereiche des Roboters

Für jeden Arm kann eine Liste mit verbotenen Bereichen vorgegeben werden. In Abbildung 36 ist der zugehörige Dialog abgebildet. Er besteht aus einem Listenfeld mit allen eingestellten Bereichen im oberen Bereich, zwei Schaltflächen zum Hinzufügen und Löschen eines Bereichs sowie einer Schaltfläche zum Schließen des Dialogs. Ein bestehender Bereich kann durch einen Doppelklick bearbeitet werden.

Mit den Kontrollfeldern kann festgelegt werden, ob der Bereich überhaupt überwacht werden soll und was bei einem Treffer passiert. Es können beide Relais des Steuermoduls umgeschaltet oder die Bewegung des Roboters vollständig gestoppt werden. Die Aktionen werden ausgelöst, wenn alle nicht ignorierten Gelenke gleichzeitig innerhalb des jeweiligen verbotenen Bereichs sind.

4.5.2 3D Rekonstruktion der Roboter Umgebung in den virtuellen 3D Raum

Die Fernsteuerung des Humanoid-Roboters und dessen Interaktion mit der Umgebung erfordern echtzeitfähiges Wissen über seiner Position sowie ein gesamtes Bild seines Umfeldes.

Anhand der im Roboterkopf (siehe Abbildung 37) eingebauten Sensoren (Full-HD-Stereokamera, RGB-Kamera und Infrarot-Kamera) wird die Roboterumgebung visuell erfasst und digitalisiert. Aus den Rohdaten der Sensoren wird mit dem Verfahren „simultanen Lokalisierung und Kartographierung SLAM“ (siehe Abschnitt 2.7) die Umgebung des Roboters in einem 3D virtuellen Raum rekonstruiert.

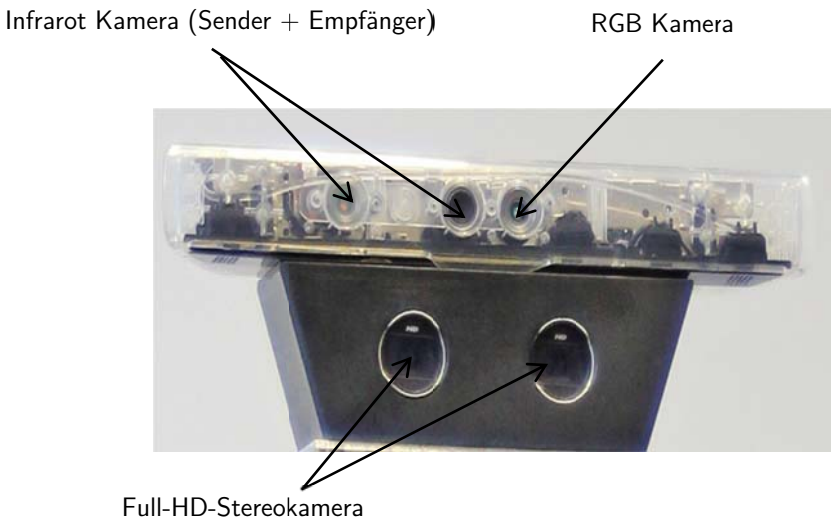


Abbildung 37: Sensoren System

4.5.2.1 Programmstruktur

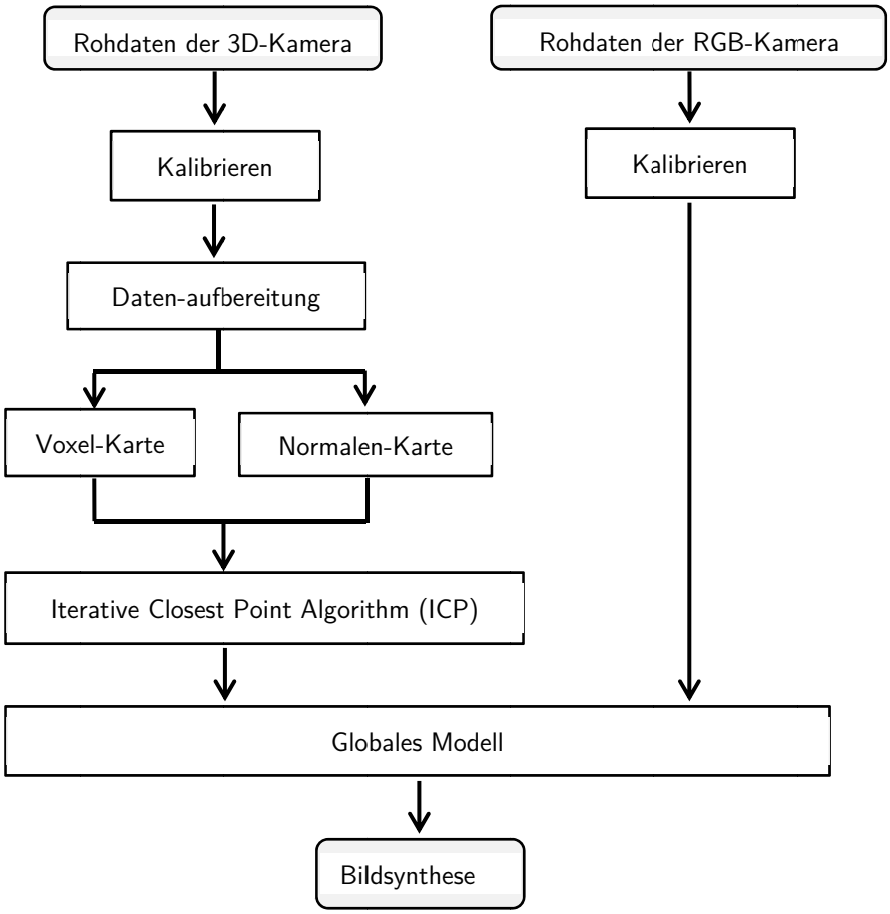


Abbildung 38: Programmstruktur SLAM

Die Struktur des SLAM-Programms ist in Abbildung 38 dargestellt. Die Rohdaten der 3D-Kamera werden zuerst kalibriert und anschließend mit der Anwendung der drei Filter (Bilateral Filter, Non-Local Means Filter und K-Nearest Neighbor Filter) sowie vordefinierter Ausschluss-Kriterien bearbeitet (siehe Abschnitt 2.7.1 und 2.7.2).

Aus den Daten werden die Positionen der Voxel und der zugehörigen Normalen berechnet. Die resultierenden Punktwolken werden mit Hilfe der ICP Algorithmus bearbeitet und die Koordinatentransformationen jedes Punkt so bestimmt, dass die Abstände innerhalb der Punktwolken minimiert werden. Diese Daten werden im Anschluss im globalen Modell gespeichert. Parallel dazu werden die Rohdaten der RGB-Kamera kalibriert und ebenfalls direkt in dem globalen Modell gespeichert. Die Geometrie-, Textur- und Positionsdaten aus dem globalen Modell werden im Anschluss zur Bildsynthese verwendet.

Zur Reduktion von Messfehler wird der Vorgang so oft wiederholt, bis eine Fehler Quote unter 10 % im generierten Bild erreicht (siehe Abbildung 39).

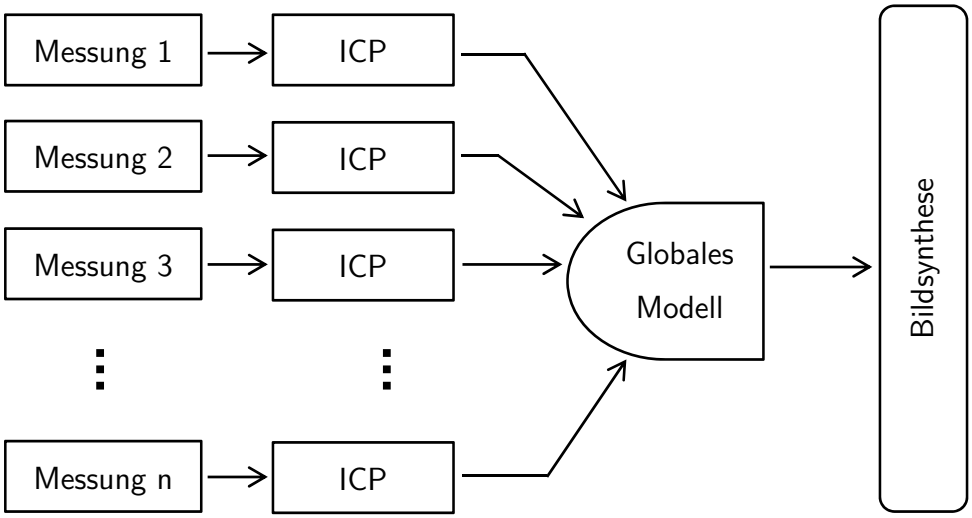


Abbildung 39: Fehlerreduktion

Um die Rechenzeit des Programms zu verkürzen, wird aufgrund der anspruchsvollen und hohen Anzahl an Berechnungen die Technologie der GPGPU´s (General Purpose Computation on Graphics Processing Unit) verwendet. Durch die Verwendung des Grafikkarten-Prozessors werden die Berechnungen stark parallelisiert, sodass tausende Operationen parallel ausgeführt werden können.

Mit Hilfe der von Nvidia entwickelte Technologie CUDA (Compute Unified Device Architecture) werden die Berechnungen der digitalen Bildverarbeitung für die 3D Rekonstruktion auf der Grafikkarte durchgeführt (siehe Abschnitt 2.6).

Um die Leistungen der GPU in Anspruch zu nehmen, wird eine geeignete Strategie bezüglich der Anwendung gegenüber der Anzahl von Threads und Blöcken entworfen sowie die Wahl der Speicherstrukturen festgelegt.

Die Programmstruktur ist in der Abbildung 40 dargestellt.

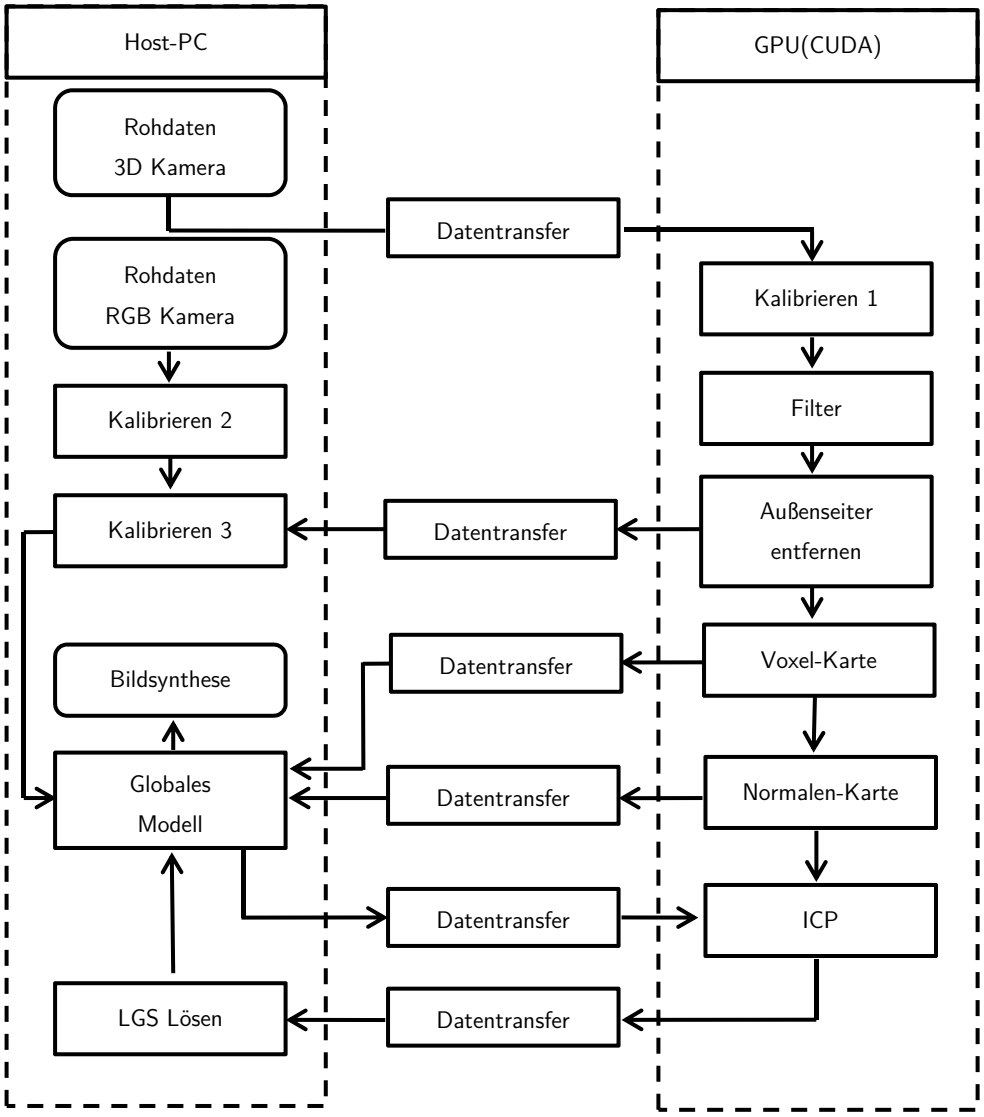


Abbildung 40: Programmstruktur SLAM-CUDA

Die Rohdaten der 3D-Kamera werden auf dem internen Arbeitsspeicher der Grafikkarte gespeichert. Auf der GPU werden die einzelnen Pixel bzw. Voxel parallel kalibriert, gefiltert

und Messfehler entfernt. Eine Kopie dieser Daten wird auf dem Host-PC gespeichert. Mit deren Hilfe werden die Rohdaten der RGB-Kamera kalibriert.

Die kalibrierten Pixel bzw. Voxel werden auf der GPU zur Erzeugung der Voxel- und Normalenkarte verwendet. Eine Kopie der erzeugten Karten wird zum Host-PC zurückgeführt und in dem globalen Modell gespeichert.

Die Starrkörpertransformation der Voxel- und Normalenkarte wird weiter auf der GPU mit Hilfe der ICP (Iterative Closest Point Algorithm) durchgeführt.

Am Ende werden die Daten zum Host-PC transferiert und zur Lösung der linearen Gleichungen sowie zur Bildsynthese verwendet.

4.5.2.2 Bildsynthese

Zur Erzeugung der 3D Bilder wird ein Rendering-System basierend auf dem Lochkameramodell implementiert (siehe Abschnitt 2.8).

Eine virtuelle Kamera wird in den Ursprung des Koordinatensystems gelegt. Mit den intrinsischen Parametern wird eine perspektivische Abbildung mit gebundenen Lichtstrahlen zum Ursprung erzeugt [Tön10]. Abbildung 41 zeigt das geometrische Grundprinzip auf.

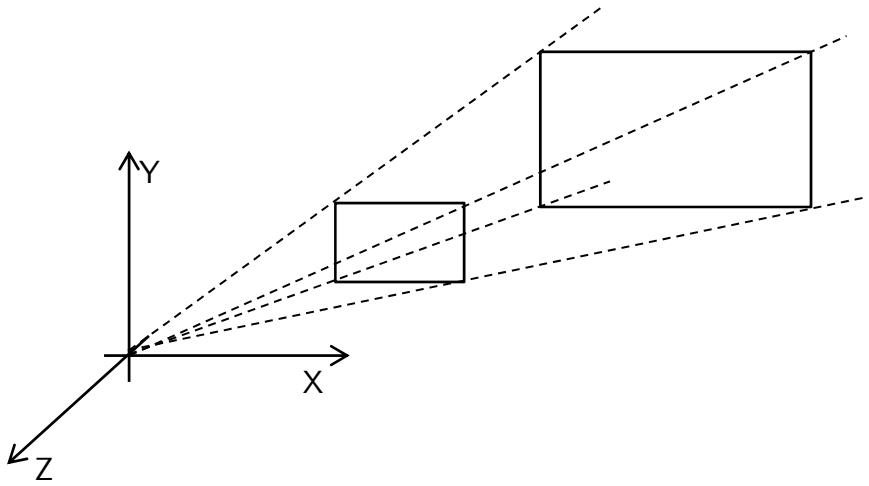


Abbildung 41: Virtuelles Lochkamera-Modell [Tön10]

Der Darstellungsbereich wird zur Leistungsverbesserung auf eine nahe und eine entfernte Ebene beschränkt.

Die nahe Ebene wird im Folgenden Nahebene genannt und ist vom Ursprung ausgesehen die erste Ebene des Objekts. Die weiter entfernte Ebene wird mit Fernebene bezeichnet.

Die Projektion des 3D-Raums, beschränkt durch die äußeren Strahlengänge, sowie der Nah- und Fernebene, wird durch die Projektionsmatrix P beschrieben.

$$P = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{r+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{4.81}$$

Hier ist n der direkte Abstand vom Koordinatenursprung zur Nahbene, f der Abstand zur Fernebene, l und r die x-Koordinaten der linken und rechten vertikalen Kante der Nahbene und b und t die zugehörigen y-Koordinaten [Tön10].

Die geometrische Abbildung der 3D Aufnahmen wird anhand der Bildsynthese von viereckigen Oberflächenabschnitten vollzogen.

Für die Erzeugung des 3D-Bildes wird aus den monochromatischen Tiefeninformationen und der Farbinformationen eine dreidimensionale Oberfläche erzeugt.

Die quadratischen Bilder, die durch Tiefensensor und Farbsensor aufgenommen werden, können jeweils zu einem rechteckigen Panorama zusammen gefügt werden. Die beiden Panoramen werden anschließend von der Mitte beginnend symmetrisch aufeinander gelegt.

Die Abbildung 42 zeigt eine virtuelle Rekonstruktion des Mechatronik-Labors am IPeG. Das 3D-Modell besteht aus 74 komplementären Messungen bzw. 22.732.800 Messpunkten. In der Abbildung sind starke Messabweichungen zu erkennen. Diese resultieren zum Großteil aus der geringen Auflösung des Infrarot Kamerasystems.

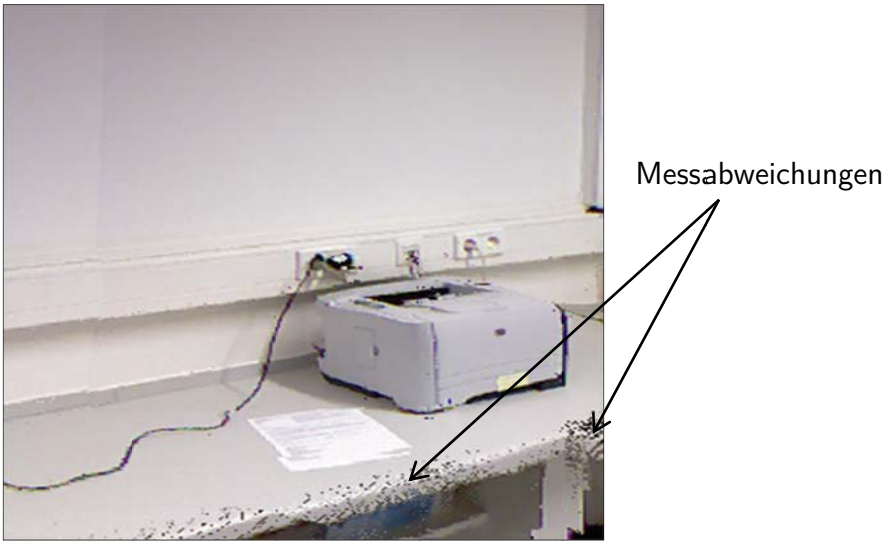


Abbildung 42: Beispiel eine Rekonstruktion des Mechatronik Labors des IPeG

4.5.2.3 Benutzeroberfläche

Die Steuerungsoberfläche der implementierten 3D-Rekonstruktionsfunktionen ist in Abbildung 43 dargestellt. Um den positiven Parallaxe-Effekt (3D) zu bewirken werden die Bildinformationen getrennt für das linke und rechte Auge dargestellt.

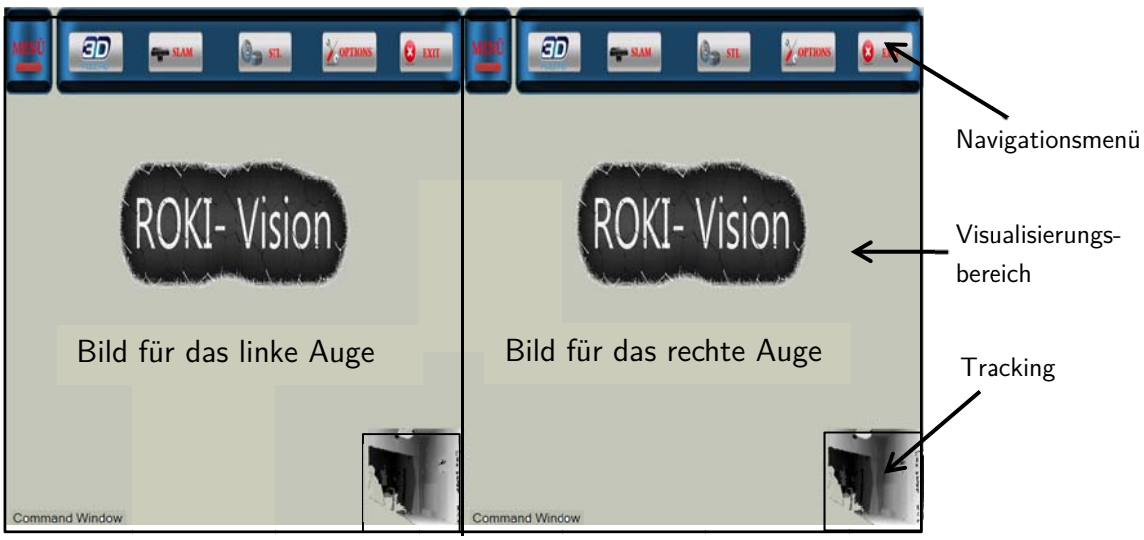


Abbildung 43: Benutzeroberfläche RoKi-Vision in 3D (Side By Side)

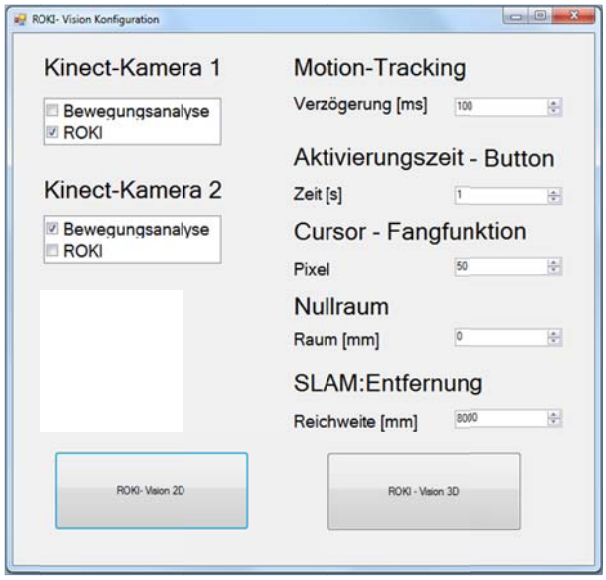


Abbildung 44: Konfigurationsmenü RoKi-Vision

Mit Hilfe des Konfigurationsmenüs (Abbildung 44) werden die Sensorsysteme sowie die Reaktionszeit und Empfindlichkeit des Trackingsystem eingestellt. Die eingestellten Parameter werden in einer Konfigurationsdatei gespeichert und vom Hauptprogramm beim Start angerufen.

4.5.2.4 Schnittstelle zum Laden und zur Visualisierung von 3D-Objekten

Bei der Fernsteuerung eines Humanoid-Roboters in einer fremden Umgebung sind Vorkenntnisse über diese Umgebung sehr hilfreich. Zum Beispiel, wenn Gebäudepläne oder CAD-Dateien von dieser Umgebung vorhanden sind, können diese zuerst im virtuellen Raum betrachtet und studiert werden, um Hindernisse und Kollisionen beim realen Einsatz zu vermeiden.

Zu diesem Zweck wurde eine Schnittstelle zum Laden und Visualisieren von 3D-Geometrien entwickelt. Diese Schnittstelle bietet die Möglichkeit, CAD-Dateien im STL-Format mit einer Anzahl von maximal $23 \cdot 10^6$ Kontenpunkten einzulesen und zu visualisieren.

Aufgrund der fehlenden Informationen zur Textur, bzw. der Farbe der Oberflächen in dem STL-Format wird eine statische Farbe vorgegeben (monochrom). Abbildung 46 zeigt die Visualisierung der in einem CAD-System konstruierten Mechatronik-Labors des IPeG.

Abbildung 45: Daten-Vorschau

Abbildung 46: Visualisierung

4.5.2.5 Eingabemechanismen

Die Bedienung der Oberfläche kann mit der Maus im 2D Modus oder über Gestensteuerung im 3D Modus ausgeführt werden. Um Konflikte bei der Gestensteuerung mit der Steuerung des Humanoid-Roboters zu vermeiden, wurde eine zweite Kamera zur Bewegungserfassung des Bedieners verwendet.

Die Eingaben im 3D Modus werden mit der Position der linken und der rechten Handfläche durchgeführt.

Das Zoomen, Verschieben und Rotieren von virtuellen Objekten wird durch vordefinierte Gesten realisiert (Abbildung 47), z.B. das Vergrößern virtueller Objekte wird durch ein einfaches Auseinanderbewegen der Hände vor dem Körper ausgeführt.

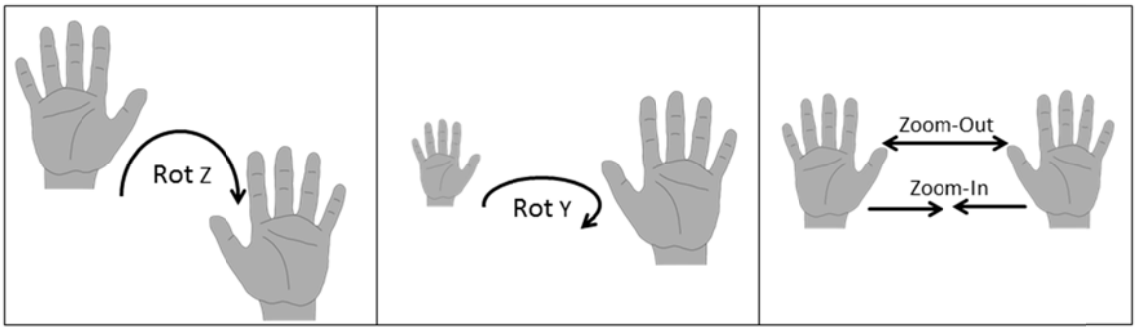


Abbildung 47: Beispiel von Steuerungsgesten

5 Hardware

Die Hardware für den Roboter besteht aus folgenden Modulen: einer Steuerelektronik (Steuerungsmodule, Motorentreiber, Stromversorgung...), einer Antriebseinheiten und einem Roboter Skelett (Gelenke, Führungen).

5.1 Steuerungsmodul: RoboControl

Durch die Verknüpfung der verschiedenen Hardware bzw. Software Komponenten entstehen einige Herausforderungen. Das unkontrollierte Zeitverhalten der Regelkreise ist aufgrund der vielen getrennten Verarbeitungsschritte und der jeweils nötigen Datenübertragung ungenügend. Zur Lösung des Problems wurde eine spezielle Steuerungshardware auf Basis von Echtzeitprozessoren entwickelt, welche die gesamten Bewegungsabläufe des Humanoid-Roboters steuert und kontrolliert.

Das Steuerungsmodul wertet alle Signale der am Roboterarm vorhanden Sensoren aus und sorgt für die notwendige Weiterverarbeitung. Anhand der gewonnenen Daten werden die Positionen aller Gelenke des Arms mittels der vorhandenen Schrittmotoren gesteuert.

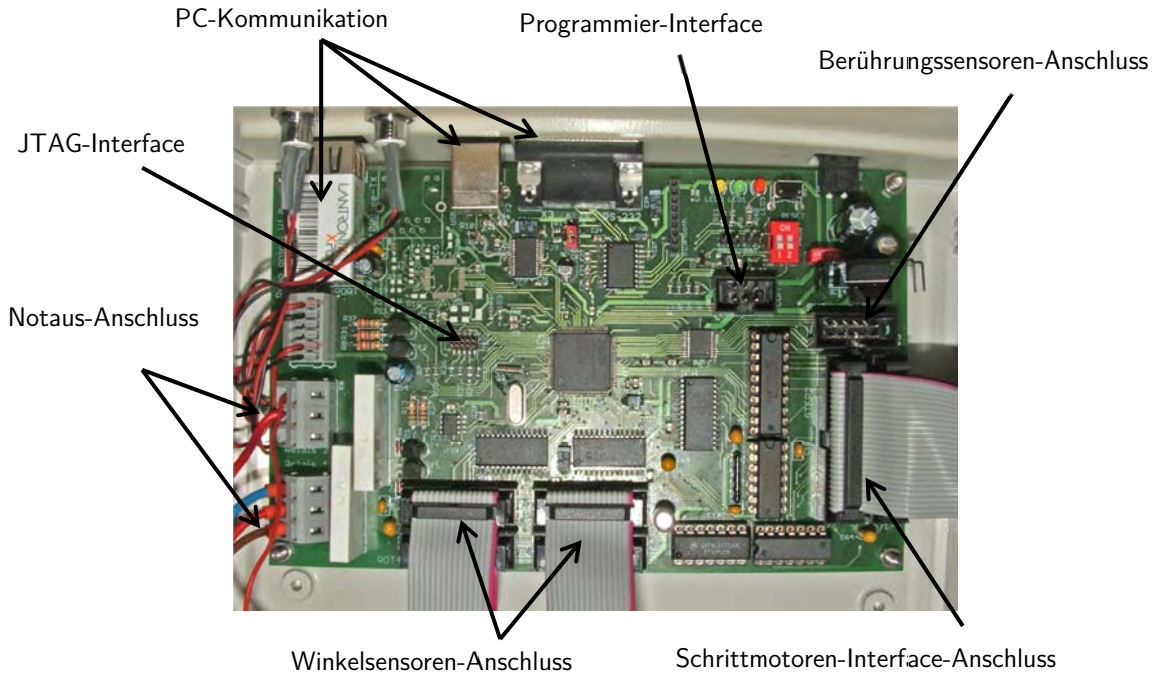


Abbildung 48: RoboControl Steuerungsmodul

Ein Modul dient zur Steuerung eines Roboterarms. Dies bedeutet, es müssen sechs Gelenke mit je einem Inkrementalgeber für die Position, einem Nullschalter und einem Schrittmotor unterstützt werden. Weiterhin müssen mindestens fünf Berührungssensoren für die Finger eingelesen werden und zwei Relais für externe Signalgeber zur Verfügung stehen. Um zukünftige Erweiterungen zu ermöglichen, sind Ein- und Ausgange für jeweils acht Gelenke und Berührungssensoren vorgesehen.

Für die Kommunikation mit der PC-basierten Steuerungssoftware sind eine serielle Schnittstelle nach RS-232, eine USB-Schnittstelle (Universal Serial Bus) sowie eine Ethernet-Schnittstelle eingebaut.

Die Funktionalität wird größtenteils durch Software bereitgestellt, welche auf einem reprogrammierbaren Mikrocontroller gespeichert ist. Die Erweiterungshardware dient zur notwendigen Konditionierung, Anpassung und Wandlung der elektrischen Signale. Durch diesen Aufbau ist es möglich, Fehler zu beheben oder sich ändernde Anforderungen zu realisieren, ohne die Hardware anzupassen.

Die Hardware des Steuerungsmoduls besteht aus einem zentralen Mikrocontroller, verschiedenen integrierten Schaltkreisen (ICs) für die Bereitstellung von Ein- und Ausgängen und weiteren Schaltkreisen für die Schnittstelle zum Computer. Zusätzlich sind diverse Hilfsschaltungen, z. B. für die Spannungsversorgung, die Relaisansteuerung oder die Programmierung des Mikrocontrollers, vorhanden.

Abbildung 49 zeigt eine vereinfachte Version des logischen Aufbaus sowie die Verknüpfungen zwischen den einzelnen Funktionsgruppen.

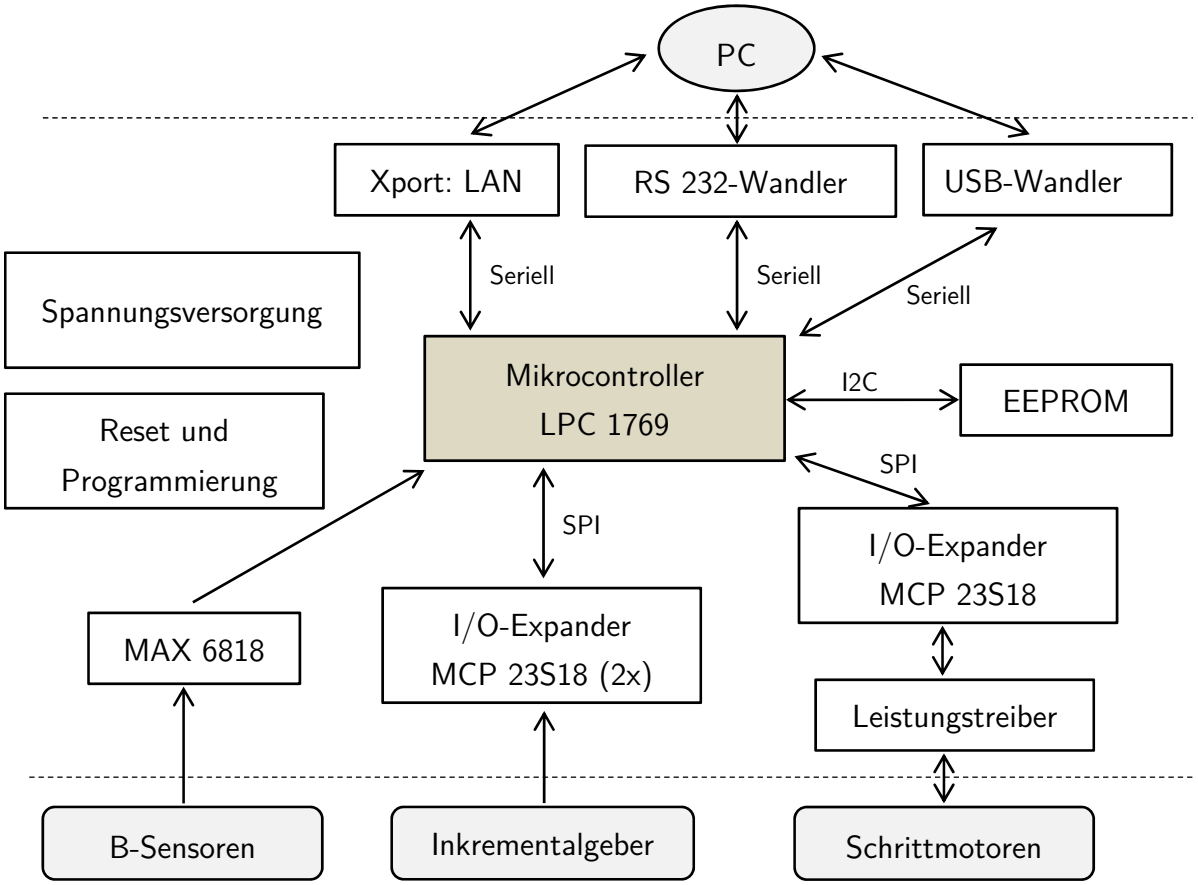


Abbildung 49: Logischer Aufbau des Steuerungsmoduls und die Verknüpfungen zwischen

Die zentrale Einheit des Steuerungsmoduls ist der Mikrocontroller und die dazu nötige zusätzliche Beschaltung. In den folgenden Abschnitten werden die einzelnen Blöcke genauer erläutert.

5.1.1 Mikroprozessor

Als Mikrocontroller wurde ein LPC 1769 von NXP Semiconductors ausgewählt. Der Controller kombiniert eine ARM Cortex-M3 CPU mit 512 KiB Programmspeicher, 64 KiB Arbeitsspeicher und einer Vielzahl von weiteren Funktionsblöcken. Diese sind u.a. vier serielle UARTs (Universal Asynchronous Receiver Transmitter), ein integrierte USB und ein Ethernet-Controller, mehrere SPI-Schnittstellen (Serial Peripheral Interface), Analog- Digital-Wandler, Timer und PWM-Bausteine (Pulsweitenmodulation) sowie generische digitale Ein-/Ausgabe-Pins. Der Controller kann mit bis zu 120 MHz getaktet werden und erreicht dabei bis zu 120 MIPS (Million Instructions Per Second) [Nxp11]. Dem Mikrocontroller wurde ein

EEPROM (Electrically Erasable Programmable Read-Only Memory) mit 8 KiB Speicherplatz zur dauerhaften Speicherung von Konfigurationsdaten beigestellt. Die Anbindung erfolgt über einen I²C-Bus (Inter-Integrated Circuit).

Der LPC 1769 besitzt zwei Quarz-Oszillatoren. Der erste Oszillator erzeugt den Haupttakt für den CPU-Kern und alle weiteren Funktionsgruppen. Dieser Takt kann intern mit einer PLL (Phase Locked Loop) vervielfacht werden, sodass auch ein Quarz mit niedriger Frequenz benutzt werden kann. Für diesen Oszillator wird ein Quarz mit 12 MHz eingesetzt, da diese durch den PLL 10x multipliziert werden kann und zum maximalen CPU-Takt von 120 MHz passt. Der zweite Oszillator versorgt die eingebaute Echtzeit-Uhr und wird mit einer festen Frequenz von 32.768 kHz getaktet.

Beide Oszillatoren (Q) sind Grundton-Pierce-Oszillatoren. Ein schematischer Aufbau ist in Abbildung 50 dargestellt. Diese enthält jeweils zwei Kondensatoren C_1 und C_2. Der dritte Kondensator C_s stellt kein real vorhandenes Bauteil dar, sondern fasst die parasitären Kapazitäten, z. B. von Gehäuse, Anschlussbeinen oder Leiterbahnen zusammen.

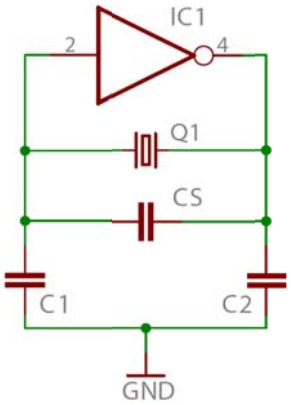


Abbildung 50: Pierce-Oszillator

Die externen Kapazitäten müssen der Lastkapazität C_L des Quarzes entsprechen [Pie23]:

$$\frac{C_1 C_2}{C_1 + C_2} + C_s = C_L \tag{5.1}$$

Aus praktischen Gründen wird die Kapazität von C_2 identisch zu C_1 gewählt:

$$\frac{C_1}{2} + C_s = C_L \tag{5.2}$$

Die geforderte Lastkapazität wird aus dem Datenblatt des verwendeten Quarzes entnommen und beträgt bei dem hier benutzen Quarz ungefähr 20 pF. Die parasitären Kapazitäten betragen 8 pF [Cer04].

$$C_1 = 2(C_L - C_s) \approx 2(20pF - 8pF) = 24pF \tag{5.3}$$

5.1.2 Schrittmotoreninterface

Jedes Gelenk des Roboters wird durch einen Schrittmotor bewegt. Diese werden mit einem Motortreiber HP 5042 gesteuert. Der Treiber benötigt einen definierten Signalverlauf als Eingangssignal. Zusätzlich liefert er einige Signale mit Informationen zurück. Besonders hierbei hervorzuheben ist hier das Not-Aus-Signal.

Für eine ruhige und gleichmäßige Bewegung der Gelenke ist es wichtig, dass die Steuersignale zu einem exakten definierten Zeitpunkt kommen. Variationen würden sich in unsteten Bewegungen und damit Ruckeln auswirken.

Zur Erzeugung der Steuersignale werden acht im LPC 1769 integrierte Timer- und PWM-Einheiten verwendet. Diese arbeiten unabhängig vom CPU-Kern und können daher exakte Signale erzeugen.

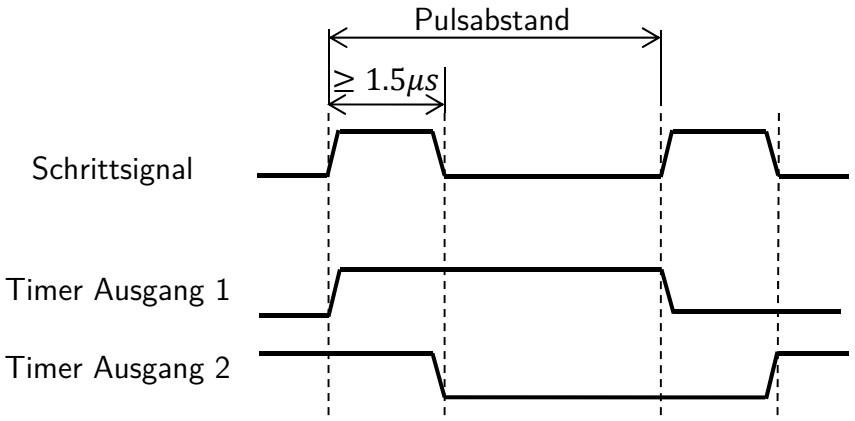


Abbildung 51: Signalform zur Schrittmotoransteuerung

Das zu erzeugende Schrittsignal ist in Abbildung 51 dargestellt. Es besteht einerseits aus einem aktiven Pegel, der mindestens für 1.5 µs gehalten wird. Anderseits besteht es aus einer inaktiven Pause, mit der die Geschwindigkeit der Motoren kontrolliert wird. Bei jeder fallenden Flanke von aktiv nach inaktiv bewegt der Motortreiber den Schrittmotor um ein Inkrement weiter. Die Drehrichtung wird mit einem separaten Signal vorgegeben.

Die geforderte Signalform kann nicht direkt mit den vier Timer-Einheiten erzeugt werden. Jeder Timer besitzt mindestens zwei Ausgänge, deren Polarität mit einer einstellbaren Frequenz invertiert werden kann. Dabei ist das Signal immer gleich lang aktiv bzw. inaktiv. Durch Verknüpfen zweier phasenverschobener Ausgänge mittels eines Exclusiv-Oder-Gatters kann die gewünschte Signalform wie in Abbildung 51 dargestellt realisiert werden.

Die notwendige Schaltung ist in Abbildung 52 dargestellt. Die vier PWM-Einheiten des LPC sind erweiterte Timer-Einheiten, die unter anderem die Exclusiv-Oder-Verknüpfung schon integriert haben. Somit benötigen diese keine weiteren Logik-Einheiten.

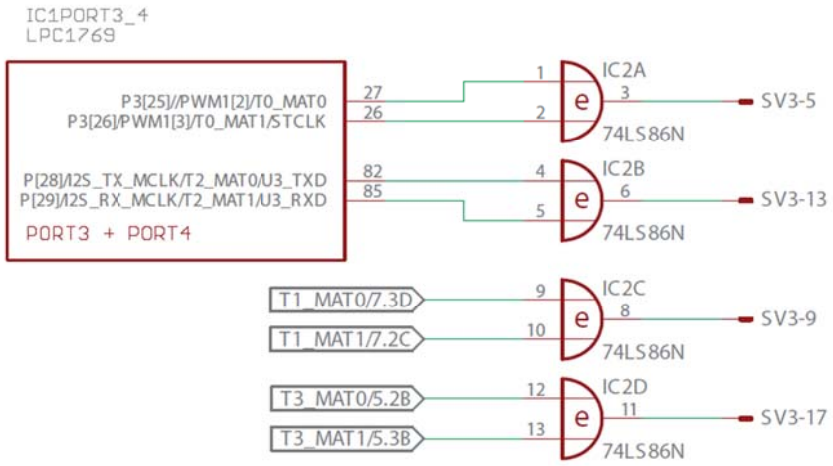


Abbildung 52: Schaltplanausschnitt der Schrittmotoransteuerung

5.1.3 Sensorinterface

In jedem Gelenk ist ein magnetischer Inkrementalgeber vom Typ AS 5304A von *austriamicrosystems* als Positionsgeber verbaut. Jeder Inkrementalgeber liefert digitale Signale. Dies sind die Quadraturkanäle A und B, welche die Rotation codieren und ein Index-Signal, welches einen magnetischen Nulldurchgang anzeigt. Das Index-Signal wird zur Auswertung der Rotation nicht benötigt, soll aber für eventuelle Erweiterungen trotzdem eingelesen werden. Alle drei Signale sind TTL-Signale mit einem Pegel von 5V.

Weiterhin ist in jedem Gelenk ein Hall-Sensor vom Typ SS 443A von Honeywell für die Nullstellung vorhanden. Da diese Sensoren nicht mechanisch arbeiten, tritt kein Prellen auf.

Der LPC 1769 hat nicht genügend Eingänge, um alle 32 Signale der acht Gelenke direkt anschließen zu können, weshalb mehrere I/O-Expander für zusätzliche Eingänge eingesetzt

werden. Für die Auswertung der Rotationssignale per Software ist es vorteilhaft, wenn bei der Änderung eines Signals ein Interrupt ausgelöst werden kann.

Es wurden zwei integrierte Schaltkreise vom Typ MCP 23S18 von *Microchip* als I/O Expander gewählt [Mic08]. Dieser IC bietet 16 konfigurierbare Ein- und Ausgänge in zwei Gruppen zu je acht Eingängen. Alle Eingänge können als Interrupt-Quelle genutzt werden. Die Verbindung zum Mikrocontroller wird mit einem SPI-Port realisiert. Der IC hat integrierte Pull-Up-Widerstände, weshalb an den Eingängen keine weitere Beschaltung notwendig ist.

5.1.4 Kommunikationsschnittstelle

Primäre Schnittstelle zur PC-Software ist ein serieller Anschluss nach dem RS-232-Standard [Ele69]. Zusätzlich sind eine USB-Schnittstelle und ein Ethernet Netzwerkanschluss vorhanden.

Abbildung 53 zeigt die Umsetzung des seriellen TTL-Signals des LPC 1769 in ein normgerechtes RS-232-Signal. Ein TTL-Signal hat einen Spannungshub von 5V gegenüber Masse und stellt eine Eins durch eine vorhandene positive Spannung dar. Ein RS-232-Signal hingegen schwankt zwischen -12V und +12V, mit einer Eins bei negativer Spannung.

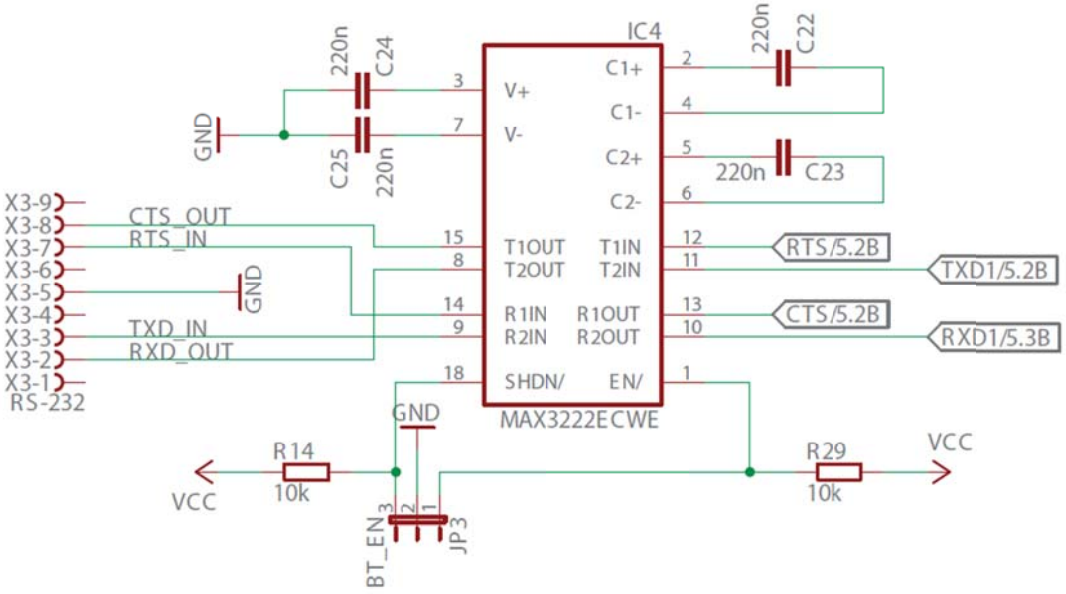


Abbildung 53: Schaltplan des RS-232-Interface

Der verwendete IC MAX 3222E von Maxim realisiert sowohl die Pegelwandlung als auch die nötige Invertierung der Signale. Die benötigten ± 12 Volt werden durch eine integrierte Ladungspumpe aus der normalen Versorgungsspannung von 3.3V erzeugt.

5.2 Boardaufbau

Das Layout der Platine wird hauptsächlich dadurch bestimmt, dass die externen Anschlüsse auf eine möglichst kompakte Platine platziert werden können. Abbildung 54 zeigt eine Übersicht über die Komponenten und die Signalbahnen ohne die Massefläche auf der Lötseite.

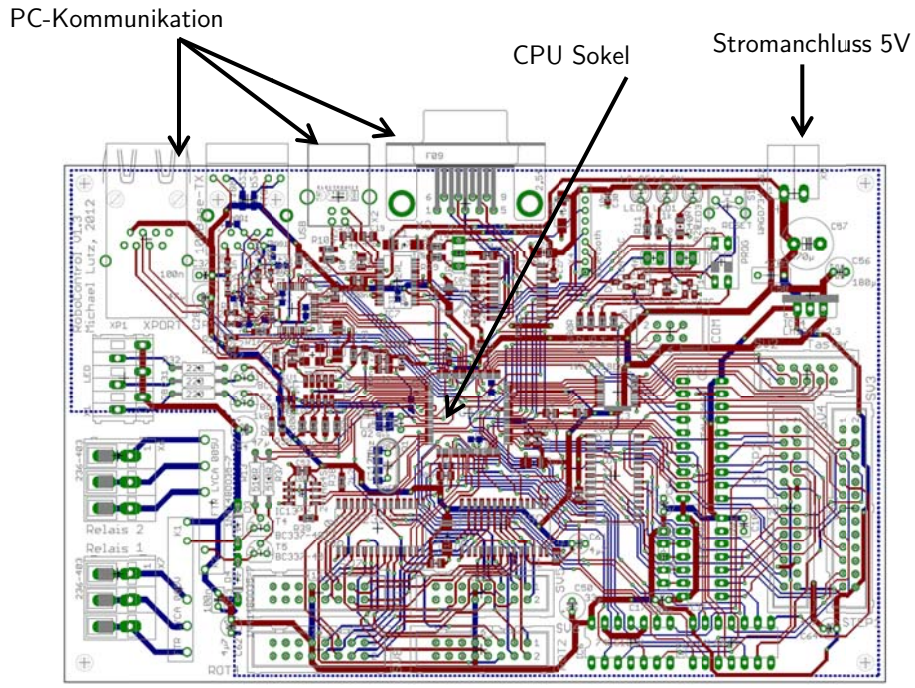


Abbildung 54: Boardlayout ohne Massefläche mit Komponenten

Beim Erstellen des Layouts werden zuerst die Schnittstellen zum Computer an einer Seite der Platine direkt angeordnet. Davon ausgehend werden die zu jeder Schnittstelle gehörigen Komponenten platziert.

Anschließend wird der Mikrocontroller zusammen mit den unmittelbar notwendigen Komponenten mittig auf der Platine platziert. Hierbei ist es besonders wichtig, dass die Siebkondensatoren der Spannungsversorgung möglichst nahe an den jeweiligen Versorgungspins des Controllers platziert werden. Auch die Leiterbahnen von den Taktquarzen zu den Oszillatoranschlüssen am Controller müssen möglichst kurz ausgelegt werden. Direkt unter den Quar-

zen dürfen keine anderen Leiterbahnen verlegt werden, da in diese sonst Störsignale eingekoppelt würden.

Die Anschlüsse für die Sensoren und Schrittmotoren werden zusammen mit den I/O Expandern und Leitungstreibern im unteren Bereich der Platine platziert. Die Anschlüsse werden als Pfostenstecker ausgeführt, da für eine Ausführung z. B. als Sub-D-Buchsen direkt auf der Platine ein deutlich größeres und damit teureres Board nötig gewesen wäre. Die Anschlüsse für die Relais und einige Status-LEDs sind an der linken Seite auf Federklemmen geführt, da diese Elemente vorzugsweise mit Einzeldrähten angeschlossen werden.

Nach dem Setzen aller Bauelemente werden die Leiterbahnen verlegt. Zuerst werden die kritischen, schnell schaltenden Signale verlegt (Taktleitungen, serielle Schnittstellen, SPI, Ethernet, etc.), bei denen es auf kurze und geradlinige Führung ankommt. Danach werden die Versorgungsspannungen platziert. Hierbei kommt es darauf an, dass alle Verbraucher ausreichend niederohmig an die Spannungsquelle angeschlossen sind. Um einen gleichmäßigen Energietransport zu gewährleisten, werden alle Verbraucher über einen angenäherten Ring angeschlossen. Abschließend werden die restlichen, unkritischen Leiterbahnen verlegt.

Die Schaltungsmasse wird nicht als diskrete Leiterbahn sondern als Massefläche ausgeführt.

Bei Digitalschaltungen ist eine saubere Masseführung besonders wichtig, da die Masse die Spannungsreferenz für alle Signale darstellt. Falls durch eine schlechte Verbindung unterschiedliche Massepotentiale in der Schaltung auftreten, kann es passieren, dass Signale nicht mehr richtig erkannt werden [Nüh98]. Eine Massefläche kann dies effektiv verhindern.

5.3 Robotermechanik

Der Roboter bildet einen humanoiden Oberkörper nach. Er besitzt zwei Arme mit jeweils fünf Gelenken, die entweder eine Schwenkbewegung oder eine Rotation ausführen. Bei den Rotationsgelenken ist die Rotationsachse koinzident mit der Gelenkachse, während die Rotationsachse der Schwenkgelenke senkrecht zur Gelenkachse steht. Alle Gelenke sind seriell angeordnet. Jeder Arm besitzt ein Schultergelenk, das aus einem Rotationsgelenk und einem Schwenkgelenk kombiniert ist. Es folgen ein Rotationsgelenk für den Oberarm an, ein Schwenkgelenk für den Ellenbogen und wieder ein Rotationsgelenk für den Unterarm. Der Roboter weist zusätzlich zwei Handnachbildungen auf.

5.3.1 Gestalt

Im Gegensatz zu Industrierobotern ist der entwickelte Humanoid-Roboter nicht für den Einsatz in realen Situationen gedacht, sondern dient als Demonstrator für das entwickelte Steuerungssystem. Beim Designentwurf wird dem Roboter ein menschenähnliches Aussehen und ähnliche Körpergröße gegeben, ohne dabei den Maschinencharakter zu unterdrücken.

Für die Mobilität des Roboters wurde ein Segway ähnlicher Antrieb entworfen. Dieser besteht aus zwei Ketten, die durch zwei Elektromotoren angetrieben sind. Im Prototyp wurde aus zeitlichen und finanziellen Gründen auf ein Antriebsystem verzichtet.

5.3.2 Torso

Arme:

Üblicherweise verfügen die Arme von Humanoid-Robotern bis zu sieben Freiheitsgrade (drei im Schultergelenk, zwei im Ellbogengelenk und zwei im Handgelenk), die größtenteils von Servomotoren über ein Harmonic-Drive Getriebe und Zahnriemen angetrieben und am Gelenk befestigt werden [Bru07]. Diese Konstruktion führt zu einem großen Volumen der Arme, was ein hohes Gewicht zur Folge hat und die Beweglichkeit der Arme einschränkt.

Für die Arme des Roboters wird das modulare Gelenksystem (Robolink) des Herstellers *igus®* verwendet. Die Robolinkgelenke ermöglichen unterschiedliche Konfigurationsmöglichkeiten. Jedes Gelenk besitzt ein bis zwei Freiheitsgrade und besteht aus Kunststoff und Aluminium. Der Antrieb erfolgt durch Zugseile, was die Auslagerung der Antriebsmotoren außerhalb der Gelenke ermöglicht und so das Gewicht der Arme reduziert.

Abbildung 55 zeigt das für den Einsatz am Roboter verwendete Gelenk mit Aluminiumrohren. Die Armlänge dabei frei wählbar ist. Die Gelenke können eine Drehbewegung um $\pm 270°$ und eine Schwenkbewegung um $\pm 90°$ erzeugen. Die Enden können wiederum um weitere Gelenke erweitert werden, was die Anzahl der Freiheitsgrade erhöht. Für *RoKi* sind sechs Freiheitsgrade in die Arme eingebaut, um die geforderten Bewegungen ausführen zu können.

Abbildung 55: Robolink Gelenke [Igus]

Antrieb:

Für jeden Freiheitsgrad wird eine separate Antriebseinheit benötigt, welche in. Abbildung 56 dargestellt sind. Diese besteht aus einem Bipolar Schrittmotor Nema 23, einem Planetengetriebe mit der Übersetzung $i = 16$ und einem Antriebsrad.

Abbildung 56: Antriebseinheit [Igus]

Die verwendeten Nema 23 Schrittmotoren haben ein Haltemoment von 2Nm und ein Schrittwinkel von 0.9° im Vollschritt-Betrieb, d.h. 400 Vollschritte pro Umdrehung. Um die Auflösung der Schrittmotoren zu erhöhen und bessere Positioniergenauigkeit zu erreichen, werden die Schrittmotoren mit $\frac{1}{16}$ Schrittwinkel angesteuert. Somit wird eine Auflösung von 6400 Schritten pro Umdrehung erreicht.

Das eingebaute Getriebe dient neben der Erhöhung der Schrittmotorauflösung zur Kraftsteigerung. Bei einer Übersetzung von $i = 16$ wird die Auflösung um ein 16-faches erhöht; gleiches gilt für das Haltemoment der Motoren.

Kopf:

Im Roboterkopf sind zwei HDV-Kameras und eine Kinect-Kamera eingebaut, um den Raum zu scannen. Der Kopf kann sich um 180° drehen und um 120° neigen. Dafür wird die Kopfeinheit mit zwei Servos (*Dynamixel AX-18A)* für die Neigung und einem für die Drehung ausgestattet.

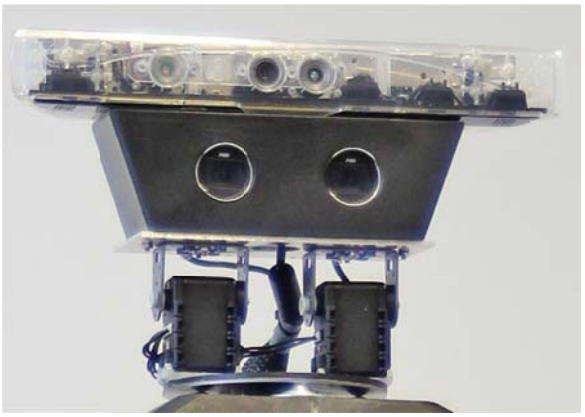


Abbildung 57: Kopf des Roboters *RoKi*

Hände:

Neben der Armbewegung verfügt *RoKi* über weitere Freiheitsgrade an den Händen, die ebenfalls über einen Seilzug mit Motoren verbunden sind. Für jede Hand sind jeweils fünf Motoren verbaut, um die Finger zu bewegen.

Abbildung 58: Hand des Roboters *RoKi*

An den Fingerspitzen sind Berührungssensoren eingebaut, um ein haptisches Feedback an den Benutzer weiterzuleiten. Dies erfolgt über einen Feedback-Handschuh.

In Abbildung 59 ist der prinzipielle Aufbau zur Erzielung der Fingerbewegung zu erkennen. Der Antrieb der Hand sitzt zentral am Körper des Roboters. Jeder Finger wird von einem Servomotor angetrieben. Die Kraftübertragung vom Motor zu den Fingern erfolgt ebenfalls über Seilzug.

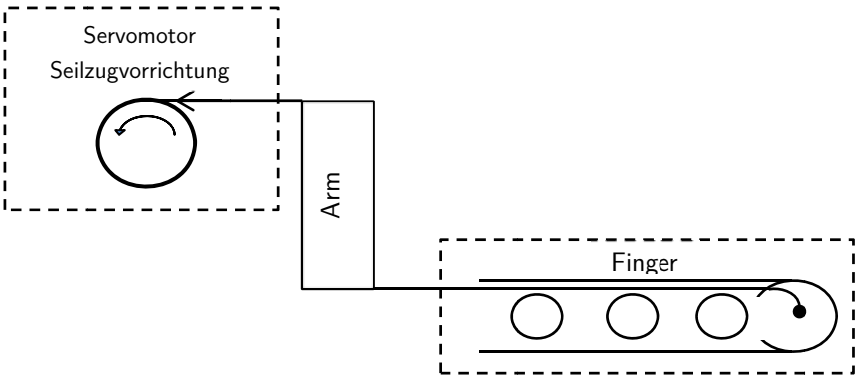


Abbildung 59: Aufbau des Seilzuges eines Fingers

Prototyp:

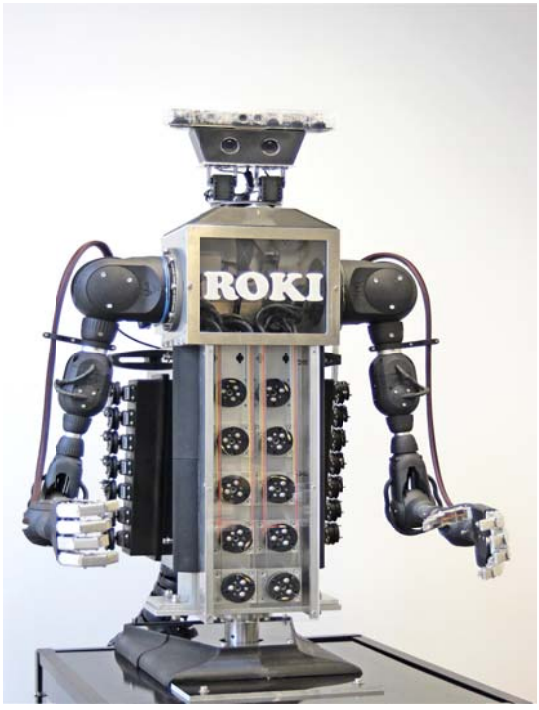


Abbildung 60: Prototyp des Humanoid-Roboters *RoKi*

6 Evaluierung und Validierung

Nach Fertigstellung des Roboter-Demonstratoraufbaus werden mit der finalen Anschteuersoftware eine Reihe von Tests zur Bestimmung der Positionier- und Wiederholgenauigkeit sowie der Hubkraft durchgeführt.

6.1 Positioniergenauigkeit

Für den Positioniergenauigkeitstest werden zehn Positionen im Raum zufällig so gewählt, dass die Grenzen des Roboterarbeitsraums nicht überschritten werden. Diese Positionen werden vom Bediener angefahren und die in der Software errechnete Zielposition wird zusammen mit der tatsächlich erreichten Position aufgenommen und anschließend verglichen.

Versuchsaufbau

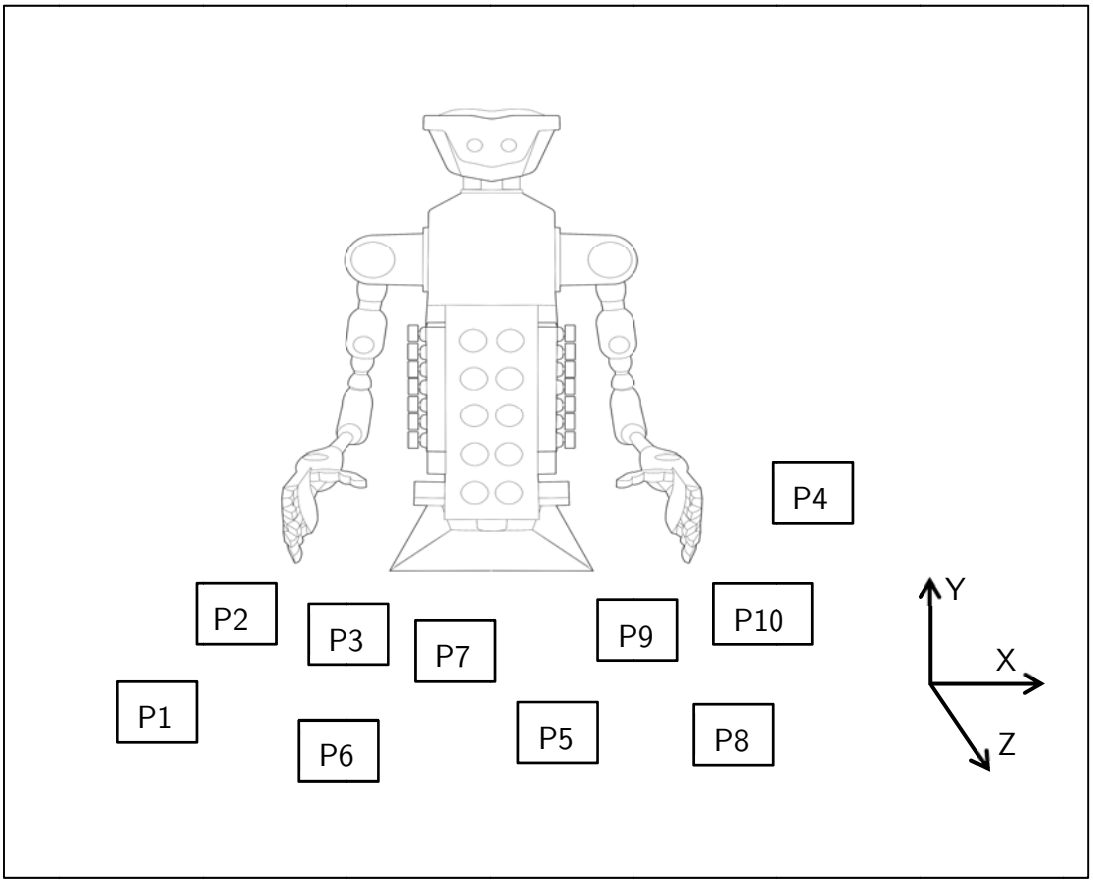


Abbildung 61: Positioniergenauigkeitsversuch

Ergebnisse

Tabelle 3: Positioniergenauigkeitsversuch

	Bedienervorghabe (Soll) [mm]			berechnete Zielposition [mm]			Ist-Position[mm]		
Versuch	X	Y	Z	X	Y	Z	X	Y	Z
1	100	100	100	103	101	95	110	93	102
2	150	150	150	149	149	151	153	152	144
3	200	200	200	199	201	205	200	205	198
4	250	250	250	245	250	254	240	244	253
5	100	200	300	102	199	304	101	203	298
6	50	300	100	51	300	104	49	299	105
7	100	20	300	96	15	301	101	19	306
8	500	500	500	505	500	503	500	504	496
9	150	500	300	153	495	296	150	502	296
10	200	300	300	202	296	303	195	301	298

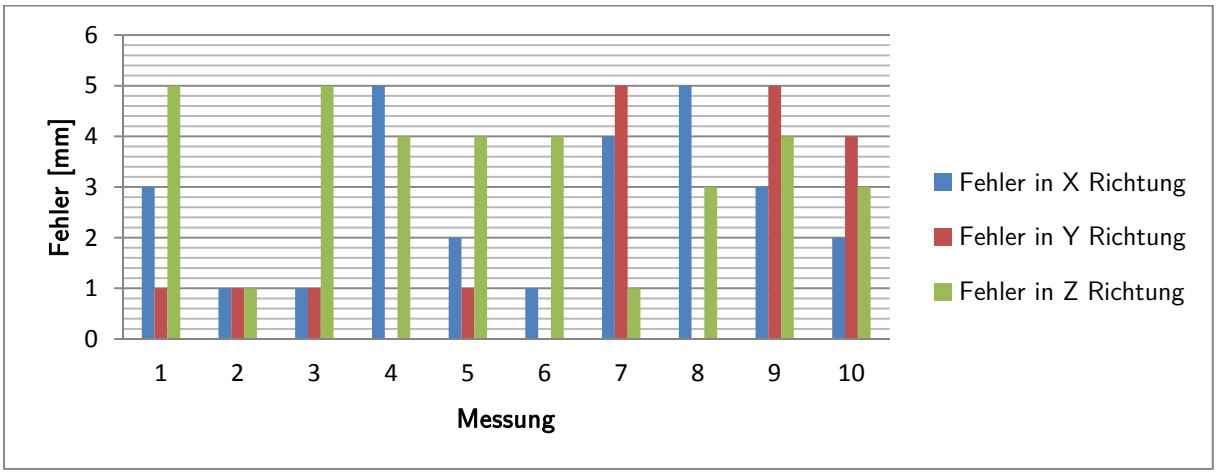


Abbildung 62: Positionierfehler der berechneten Zielposition

Die für die Erfassung der Bedienvorgaben genutzte 3D Infrarot Kamera hat eine Genauigkeit von $\pm$5mm.

Die Tests haben gezeigt, dass die von Hard- und Software berechneten Zielwerte im Toleranzintervall (Abbildung 62) der Kamera liegen [±5mm]. Die tatsächlich angefahrenen Positionen (Abbildung 63) hingegen unterscheiden sich zum Teil in größerem Maß von den Vorgaben.

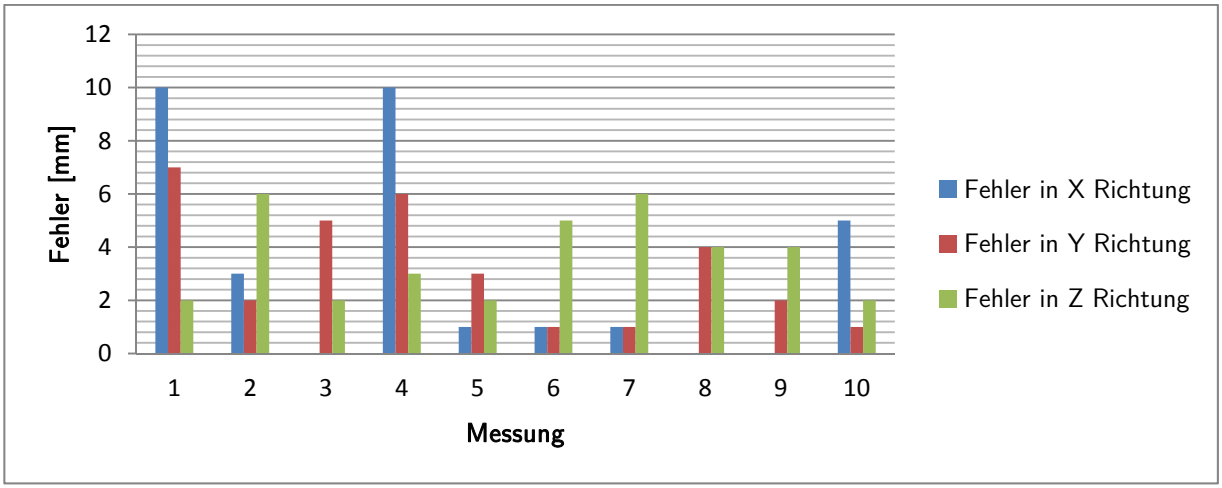


Abbildung 63: Positionierfehler der Roboterkinematik

Die Ursachen hierfür sind:

Geometrische Fehlereinflüsse:

- Nulllagenfehler der Gelenksensoren: Die verwendeten Gelenke und deren Sensoren sind konstruktionsbeding unterschiedlich gebaut. Die Position der Nullpunktsensoren ist jedem Gelenk anders justiert.
- Armlängenfehler: Die verwendeten Bauteile sind 3D gedruckt und weisen große Geometrieungenauigkeiten.
- Winkelfehler: Die verwendeten Winkelsensoren sind sehr störungsanfällig.

Mechanische Fehlereinflüsse:

- Gelenkelastizität
- Zugseildehnung
- Reibung in den Bowdenzüge
- Reibung in den Gelenkgleitlagern
- Umkehrspiel in den Planetengetriebe

Diese mechanischen und geometrischen Fehlereinflüsse können nur bedingt softwareseitig kompensiert werden, weil die Reibung und Dehnung des Seilzugsystems sich ständig in jeder neuen Position ändern. Da die Bowdenzüge aus mehreren Gliedern mit unterschiedlichen Geometrien und Materialien bestehen, stellt sich die Berechnung der Reibung und Dehnung der Bowdenzüge sehr aufwendig dar.

6.2 Wiederholgenauigkeit

Beim Wiederholgenauigkeitstest werden zwei Positionen im Raum (P1, P2, siehe Abbildung 64) festgelegt.

Die Roboterhand wird dann über die Bedienvorgabe zehnmal von P1 zu P2 bewegt. Die Ist-Positionen werden jeweils abgemessen und mit den in der Software berechneten Soll-Positionen verglichen.

Versuchsaufbau

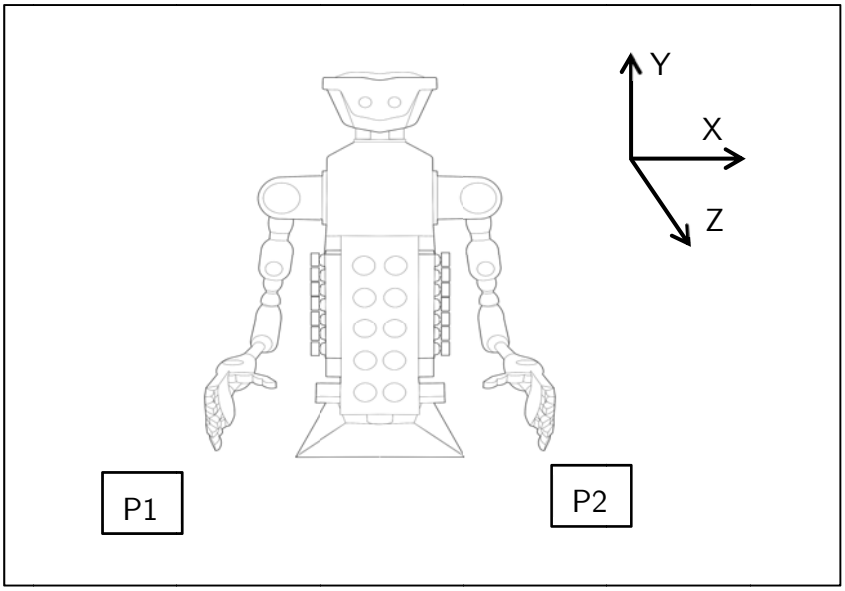


Abbildung 64: Wiederholgenauigkeitsversuch

Ergebnisse

Tabelle 4: Wiederholgenauigkeitsversuch

	Bedienvorgabe (Soll) [mm]		Berechnete Zielposition [mm]	Ist-Position [mm]
V	P1	P2	P2	P2
1	100, 100, 100	500, 500, 500	500,501,495	507,499,498
2	100, 100, 100	500, 500, 500	502,502,500	500,500,492
3	100, 100, 100	500, 500, 500	497,499,498	506,495,502
4	100, 100, 100	500, 500, 500	500,496,502	503,504,496
5	100, 100, 100	500, 500, 500	499,499,502	500,506,500
6	100, 100, 100	500, 500, 500	505,497,495	498,496,506
7	100, 100, 100	500, 500, 500	498,495,503	504,500,501
8	100, 100, 100	500, 500, 500	505,502,504	508,496,504
9	100, 100, 100	500, 500, 500	499,504,500	499,502,502
10	100, 100, 100	500, 500, 500	502,496,503	512,500,500

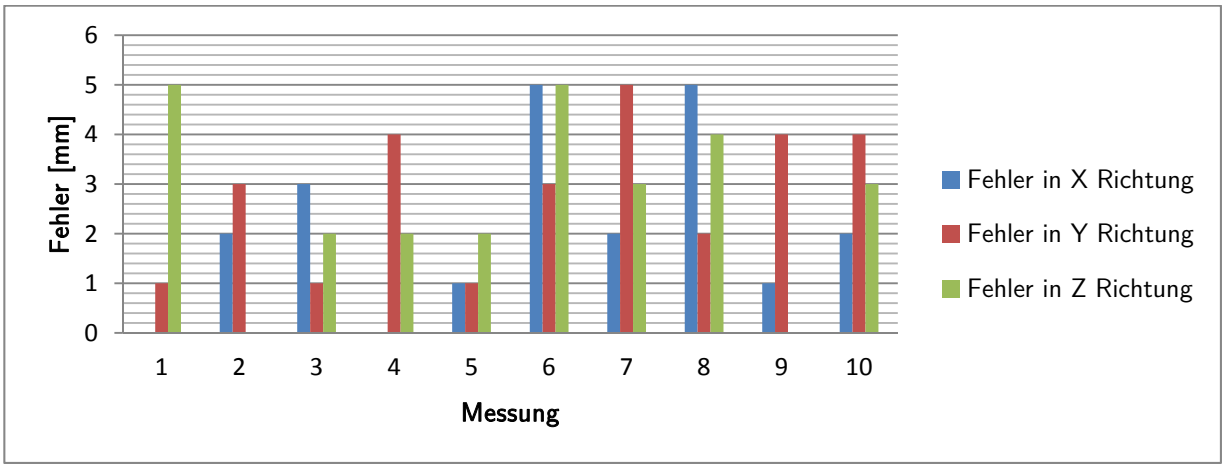


Abbildung 65: Wiederholgenauigkeitsfehler der berechneten Zielposition

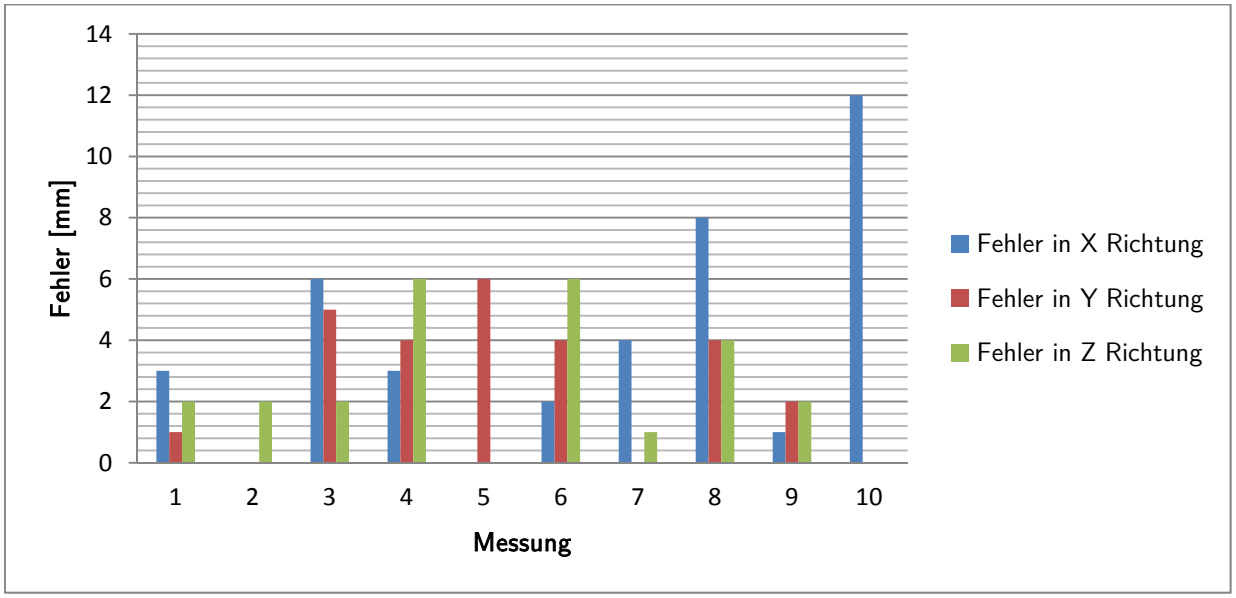


Abbildung 66: Wiederholgenauigkeitsfehler der Roboterkinematik

Bezogen auf die Wiederholgenauigkeit (mit welcher Genauigkeit fährt der Roboter aus einer gegebenen Startposition die Zielposition auf die gleichen Trajektorie/Bewegungsbahn an) ergeben sich die gleichen Aussagen, wie für die Positioniergenauigkeit. Die Wiederholgenauigkeit der Ist-Position ist ebenfalls bedingt durch die verwendeten Gelenke und deren Einschränkungen gering.

6.3 Hubkraft

Beim Hubkrafttest wurde eine leere Flasche schrittweise mit Wasser gefüllt und mit ausgestrecktem Arm angehoben, bis die Schulterschrittmotoren begonnen haben, einen Schrittverlust aufzuweisen. Die Flasche wurde jedes Mal gewogen und das Gewicht notiert.

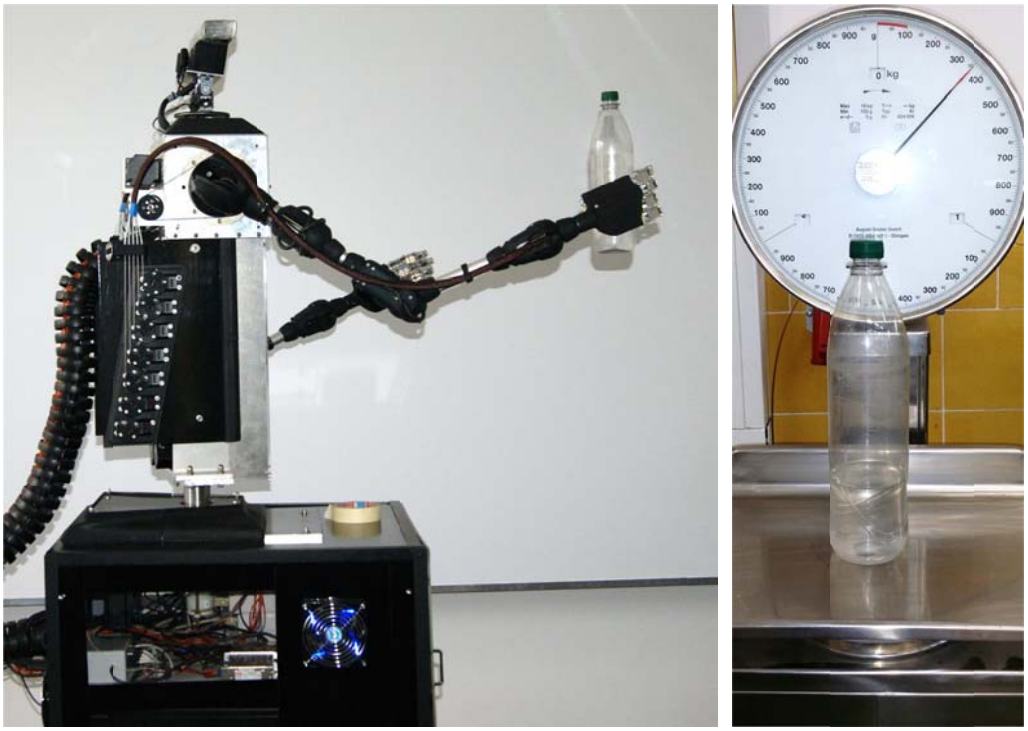


Abbildung 67: Hubkraft Test

Ergebnisse

Die theoretisch erreichbare Hubkraft der Arme bei idealen Reibkräften in den Bowdenzügen ist 5,29 Newton (540g) Durch Abnutzung der Zugseile und Verluste in den Umlenkrollen wurde nur eine Wert von 3,48 Newton (355g) erreicht.

6.4 Schwachstellen des Systems und Verbesserungsmaßnahmen

6.4.1 Schwachstellen des Systems

Durch die Verwendung von schwächeren mechanischen Komponenten ist die Positioniergenauigkeit von Roki mittelmäßig und seine Hubkraft zu gering.

Die verwendete Kamera hat eine Auflösung von 640x480 Pixel. Wegen dieser geringen Auflösung stößt die Körperskeletterfassung in einigen Situationen an ihre Grenzen. Die Kamera hat außerdem Probleme mit der Erfassung, sobald ein Gelenk verdeckt ist oder sich der Arm sehr nahe vor dem Körper des Bedieners befindet. Hier reichen die Tiefenunterschiede nicht mehr aus, um den Arm eindeutig zu identifizieren. Es kann hierdurch zu unkontrollierten Sprüngen in den Koordinaten der erkannten Gelenke kommen.
Ein weiteres Problem der Kinect Kamera und des zugehörigen SDK's (Software Development Kit) ist die fehlende Möglichkeit, Skelette anhand einer eindeutigen ID zu unterschei-

den. Hierdurch kann es zu einem ungewollten Wechsel des erfassten Skelettes kommen, sobald sich mehrere Personen vor der Kamera befinden. In einigen Fällen hat die Kamera auch Jacken oder andere Gegenstände, die einer Person ähnlich sehen, als Skelett erkannt.

6.4.2 Verbesserungsmaßnahmen

Das Wackeln und Überschwingen des Roboterarmes kann softwareseitig nicht ausgeglichen werden. Dieses Problem kann gelöst werden durch den Einsatz von hochwertigen und präzisen Robotergelenken.

Bekannte Schwachstellen, die durch die genutzte Kamera entstehen (Toleranzintervall [±5mm]) können mit Hilfe einer hochauflösenden Kamera (z.B. „Kinect 2.0“) oder hochauflösenden 3D-Scannern gelöst werden.
Eine zweite Maßnahme ist der Einsatz von mehr als einer Kamera zur Bewegungserfassung (Abbildung 68). Hierdurch kann der tote Winkel der Kamera verkleinert werden, wodurch die Problematik der verdeckten Gelenke verringert wird. Zudem kann durch den Einsatz mehrerer Kameras verhindert werden, dass sich ein Arm senkrecht vor der Kamera befindet und sich somit selbst verdeckt. Des Weiteren wird der unscharfe Bereich vor dem Körper des Bedieners verringert. Dieser Bereich wird durch den geringen Unterschied im Abstand zwischen Arm und Körper hervorgerufen. Gleichzeitig kann in Bereichen, die von mehreren Kameras erfasst werden, die Bewegungsgenauigkeit erhöht werden. Dies kann durch einen Vergleich der Kameradaten geschehen. Anschließend kann entweder der Mittelwert dieser Daten gebildet werden, oder mit Hilfe eines Algorithmus die für die Bewegungssituation besser positionierte Kamera genutzt werden.

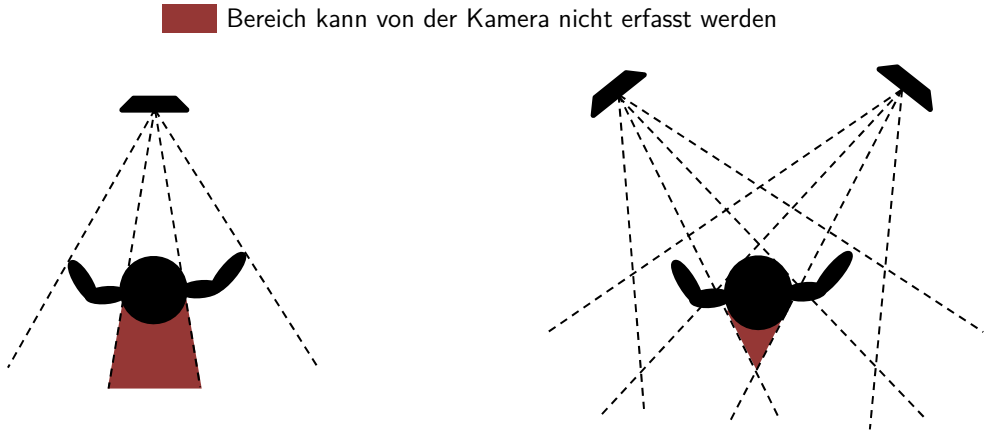


Abbildung 68: Schematischer Vergleich des Systemaufbaus mit einer und zwei Kame-

7 Praxisanwendung: Tauchroboter

Der erste konstruierte Prototyp des Humanoid-Roboters (*RoKi*) dient als Demonstrator für das entwickelte Steuerungssystem. Das System ist modular mit definierten Schnittstellen aufgebaut und kann sowohl für die Steuerung von Humanoid-Robotern als auch für andere Maschinen schnell angepasst werden.

Der erste Transfer in eine Praxisanwendung des Systems ist ein gestengesteuerter Tauchroboter zur Untersuchung von Unterwasserschweißnähten mittels Ultraschall.

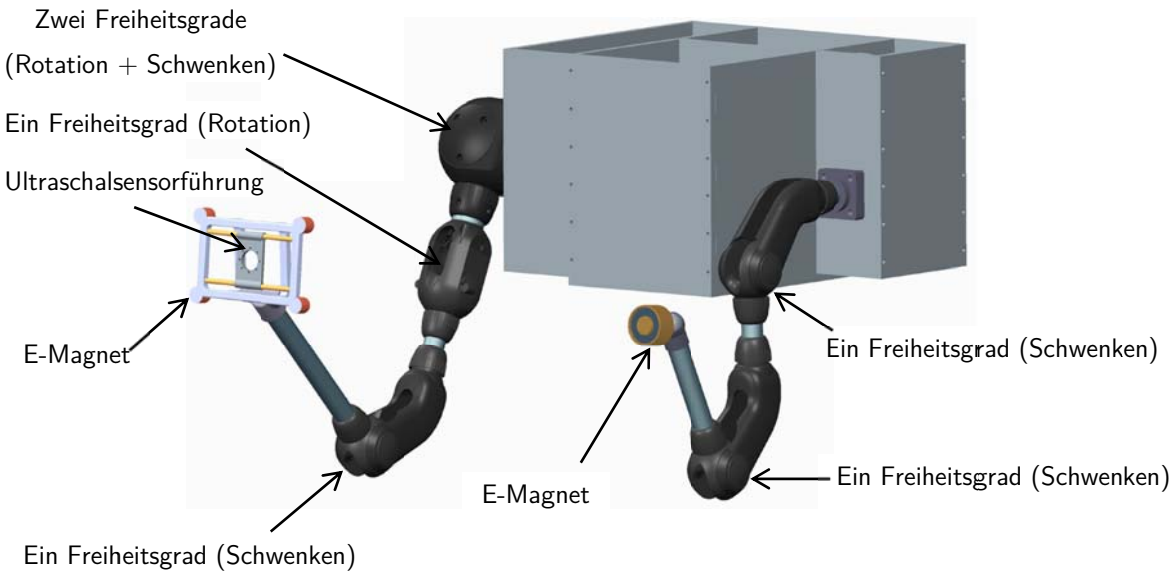


Abbildung 69: CAD Modell des Tauchroboters

7.1 Aufbau

Das System in Abbildung 70 verfügt über zwei Arme. Der linke Arm hat vier Freiheitsgrade und dient zur Positionierung des Ultraschalsensors, der rechte Arm hat zwei Freiheitsgrade und ist mit einem Elektromagnet ausgestattet.

Der rechte Arm ist für die Stabilisierung der Z Position des Tauchroboters zuständig.

Die Motoren sind alle in der Mitte, in einem wasserdichten Gehäuse angebracht. Die Datenübertragung zur Steuereinheit erfolgt über ein wasserdichtes Kabel.

Die Steuereinheit besteht neben einem Rechner, einem Monitor und einer Kinect Kamera aus zwei redundanten Robocontrol-Modulen sowie sieben Treibereinheiten für die Schrittmotoren.

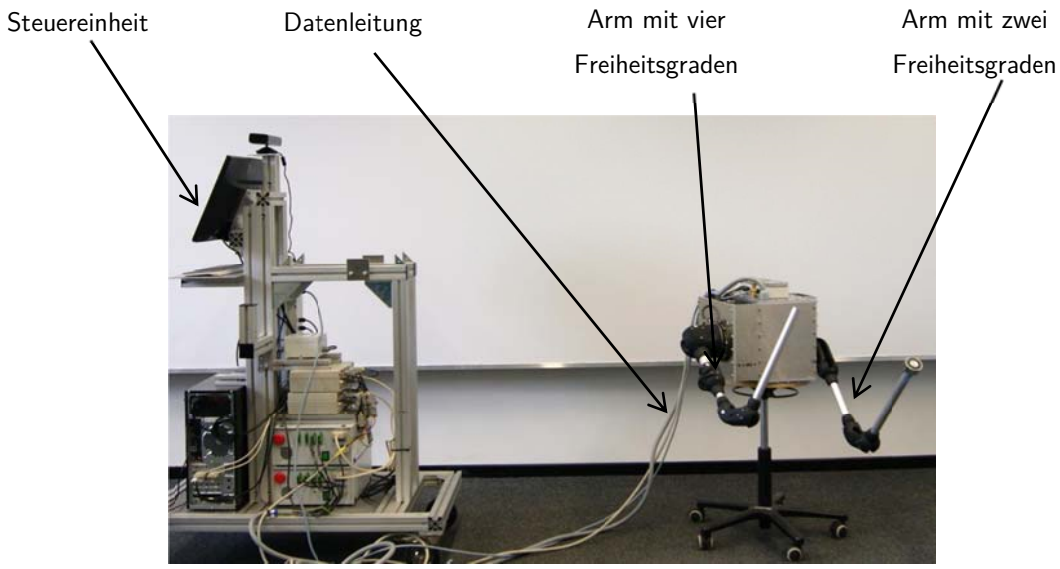


Abbildung 70: Aufbau des Tauchroboters

7.2 Tests und Validierung

Die Tests wurden im Trockenen durchgeführt um die Kalibrierung und Justierung der Gelenke einzustellen. Weitere Tests unter Wasser sind für die Zukunft geplant und können im Rahmen dieser Arbeit nicht mehr dokumentiert werden.

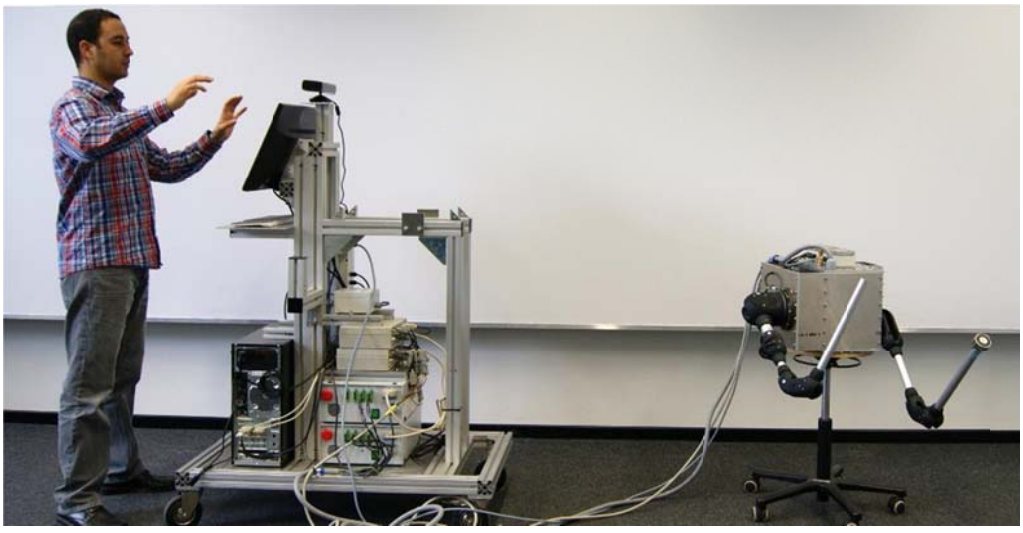


Abbildung 71: Trockentest des Tauchroboters

Der in der Abbildung 71 dargestellte Trockentest wird sowohl für die Positionier- als auch die Wiederholgenauigkeit durchgeführt. Weiterhin werden die zulässigen Arbeitsbereiche und Bewegungsgrenzen sowie die Nullpunkte der Gelenke an dieser Stelle definiert.

8 Zusammenfassung und Ausblick

Mit dem Ziel, eine intuitive Schnittstelle zur Steuerung von komplexen Maschinen auf Basis kostengünstiger Komponenten wie der Kinect Kamera zu entwickeln, wurde ein System aufgebaut, das dem Benutzer die Möglichkeit der virtuellen Teleoperation gibt. Das heißt, der Benutzer verrichtet die Arbeit in einer virtuell rekonstruierten Umgebung während ein Roboter die tatsächliche Arbeit simultan in der realen Umgebung am Einsatzort ausführt.

Für das System wurde sowohl Hard- als auch Software entwickelt, die in der Lage ist, einen Humanoid-Roboter, welcher die kinematischen Freiheitsgrade des menschlichen Oberkörpers nahezu komplett abbildet, durch Benutzereingaben über ein Kamerasystem zu steuern. Dabei wird die Kamera zur Erkennung der aktuellen Position und Armhaltung des Bedieners benutzt und der Roboter entsprechend nachgeführt. Bei der Entwicklung des Systems wurde auf eine Abtrennung der zeitkritischen Aufgaben von der übrigen Verarbeitungssoftware geachtet, um ein flüssige und störungsfreie Bewegung des Roboters zu garantieren.

Das System arbeitet bidirektional. Die erste Richtung wird für die Bewegungserfassung des Bedieners und die Umwandlung seiner Gesten in Maschinenbefehle genutzt. Die zweite Richtung besteht aus Informationen über die Umgebung des Roboters. Diese wird über Kamera- und Tiefensensoren erfasst und in Form eines 3D-Raumes für den Bediener virtuell rekonstruiert.

Das System ist modular mit definierten Schnittstellen aufgebaut und kann sowohl für die Steuerung von Humanoid-Robotern als auch für andere Maschinen (z.B. Fahrzeuge, Tauchroboter) schnell angepasst werden.

Um das System zu testen und zu validieren wurde der Humanoid-Roboter „RoKi“ entwickelt. RoKi dient als Demonstrator und ist nicht für den Einsatz in realen Situationen gedacht, deswegen wurden kostengünstige Komponenten verwendet, wie z.B. die Igus Robolink-Gelenke. Aus diesem Grund sind weder Positioniergenauigkeit noch Hubkraft des Roboters für den tatsächlichen Arbeitseinsatz geeignet.

Die erste Praxisanwendung des Systems ist ein gestengesteuerter Tauchroboter zur Untersuchung von Schweißnähten mittels Ultraschall. Im Gegensatz zum Demonstrator RoKi verfügt der Tauchroboter über wesentlich weniger Freiheitsgrade. Durch die Skalierbarkeit des Systems waren keine Anpassungen an Software oder Hardware, insbesondere Robocontrol Module, nötig, um den Tauchroboter mit diesem System zu steuern. Tests mit dem Tauch-

roboter haben gezeigt, dass das System trotz der anderen Dynamik innerhalb einer Unterwasser-Umgebung problemlos funktioniert.

Die im Rahmen dieser Arbeit entwickelte Steuerelektronik mit dazugehöriger Software kann somit als ausgereift für den Einsatz an anderen Maschinen und Geräte betrachtet werden. Insgesamt konnte gezeigt werden, dass das aufgebaute System universell einsetzbar ist und den gestellten Anforderungen entspricht.

Die Validierung anhand des aufgebauten Demonstrators RoKi konnte einige Verbesserungspotenziale des Systems sowie Herausforderungen für zukünftige Entwicklungen aufzeigen. Einer der Hauptaspekte betrifft die Qualität der Kamera. Hier verspricht die Nachfolgeversion der Kinect Kamera große Verbesserungen, da diese zum einen über eine wesentlich höhere Auflösung verfügt und neben den Gelenken und der Körperstellung des Nutzers ebenfalls über eine Fingererkennung verfügt. Darüber hinaus kann die Software zuverlässig bis zu sechs Personen verfolgen.

Die Auswertung der Kameradaten bietet weiteres Potenzial für weitere Entwicklungen. Ein grundsätzliches Problem besteht in der Gelenkerkennung in dem Fall, wenn Teile des Bedieners verdeckt sind und mit einer einzelnen Kamera nicht mehr sicher erfasst werden kann. Eine Kopplung von zwei Kameras kann hier Abhilfe schaffen.

Der nächste Schritt in der Entwicklung und dem Ausbau des Systems ist die Implementierung einer automatischen Kollisionserkennung, die über die aktuelle Definition der Wirkräume des Roboters hinausgeht und damit auch den sicheren Einsatz in hochdynamischen Umgebungen ermöglicht.

Um die parallele Steuerung von Roboter und virtueller Umgebung zu erleichtern, sollte das System weiterhin mit einem Sprachsteuerungsmodul versehen werden.

Softwareseitig ist die Entwicklung einer lernfähigen Software denkbar, die das Verhalten des Benutzers in einem gewissen Grad antizipieren kann und so bei Fehlern frühzeitig eingreift.

9 Literaturverzeichnis

[Alb07] Albitar, I. C., Graebling, P., & Doignon, C. (2007). *Robust Structured Light Coding for 3D Reconstruction.* Proc. IEEE 11th Int. Conf. Computer Vision ICCV.

[Arn10] Arne. (2010). http://botzeit.de/blog/2010_01_14_roboter_definitionen_des_begriffs.html (zugegriffen am 20.10.2014).

[Azi09] Azimi, M. (2009). *Skeletal Joint Smoothing White Paper.* http://msdn.microsoft.com/en-us/library/jj131429 (zugegriffen am 12.10.2014).

[Bai04] Baidyk, T., Kussul, E., Maekyev, O., Caballero, A., Ruiz, L., Carrera, G., et al. (2004). *Flat image recognition in the process of microdevice assembly.* USA: Pattern Recognition Letters 25.

[Ben09] Benra, J. T., & Halang, W. A. (2009). *Software-Entwicklung für Echtzeitsysteme.* Berlin: Springer.

[Ber05] Berekovic, M. (2005). *Eine skalierbare, verteilte Prozessor-Architektur mit simultanem Multi-Threading für Anwendungen der digitalen Signalverarbeitung.* Berlin: VDI Verlag.

[Bru07] Brudniok, S. (2007). *Methodische Entwicklung hochintegrierter mechatronischer Systeme am Beispiel eines humanoiden Roboters.* Karlsruhe: Karlsruhe Univ., Diss.

[Cer04] Cerda, R. (2004). *Pierce-gate oscillator crystal load calculation.* RF Design. Penton Media, Inc.

[Cla11] Clausing, A. (2011). *Programmiersprachen.* Heidelberg : Spektrum Akademischer Verl.

[Coh81] Cohen, D. (1981). *On Holy Wars and a Plea for Peace.* Computer 14 (1981), Nr. 10.

[Cor94] Corke, P. I. (1994). *Visual Control Of Robot Manipulators.* USA: Visual Servoing.

[Dab02] Dabek, F., Zeldovich, N., Kaashoek, F., Mazieres, D., & Morris, R. (2002). *Event-driven programming for robust software.* New York, NY, USA: ACM.

[Den55] Denavit, J., & Hartenberg, R. (1955). *A kinematic notation for lower-pair mechamechanisms.* Trans. of the ASME. Journal of Applied Mechanics 22.

[Ele69] Electronic Industries, A. (1969). *Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data interchange.*

[Far11] Farber, R., Green, T., Robyn, D., Miller, D., & Schaeffer, D. (2011). *CUDA Application design and development.* Elsevier.

[Gal11] Gallo, L., Placitelli, A. P., & Ciampi, M. (2011). *Controller-free exploration of medical image data: Experiencing the Kinect.* 24th Int Computer-Based Medical Systems (CBMS) Symp.

[Gil10] Giles, J. (2010). *Inside the race to hack the Kinect.* The New Scientist 208.

[Göp98] Göpfert, J. (1998). *Modulare Produktentwicklung zur gemeinsamen Gestaltung von Technik und Organisation.* Univ.-Verl., Wiesbaden.

[Gre11] Greuter, M., Rosenfelder, M., Blaich, M., & Bittel, O. (2011). *Object Detection with the 3D-Sensor Kinect.* International Conference on Research and Education in Robotics, Springer.

[Hei07] Heimann, B., Gerth, W., & Popp, K. (2007). *Mechatronik: Komponenten – Methoden – Beispiele. 3.* Fachbuchverl.Leipzig.

[Hei94] Heitmyer, C., & Lynch, N. (1994). *The generalized railroad crossing: a case study in formal verification of real-time systems.* Proc. Real-Time Systems Symp.

[Hes11] Hesse, S., & SchnellL, G. (2011). *Sensoren für die Prozess- und Fabrikautomation.* Vieweg + Teubner.

[Hon12] Honda Motor CO., L. (2012). *Humanoid robot ASIMO.* http://asimo.honda.com/Inside-ASIMO/,. 2012. – zugegriffen am 30.11.2012.

[Hop03] hopt+schuler. (2003). *Technische Daten Drehimpulsgeber 427.*

[Hor02] Horber, E. (2002). *Motion Capturing.* Universität Ulm Forschungsbericht.

[Huh10] Huhle, B., Schairer, T., Jenke, P., & W., S. (2010). *Fusion of range color images for denoising and resulution enhancement with a non-local filter; Computer and Vision Image Understanding 114.* Elsevier.

[Iza11] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., et al. (2011). *Kinect Fusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera.* Microsoft Research.

[Jac06] Jackel, D., Neubreither, S., & Wagner, F. (2006). *Methoden der Computeranimation.* Springer.

[Jäh12] Jähne, B. (2012). *Digitale Bildverarbeitung und Bildgewinnung .* Berlin: Springer.

[Jar72] JARA, J. R. (1972). *Industrial Robot.* www.jara.jp/e 2013 (zugegriffen am 10.10.2014).

[Kha07] Kharlamov, A., & Podlozhnyuk, V. (2007). *Image Denoising.* Nvidea Corporation.

[Kim09] Kim, J.-H. (2009). *Progress in robotics : FIRA RoboWorld Congress.* Korea: Federation of International Robosoccer Association.

[Kli08] Klimentjew, D., & Stroh, A. (2008). *Grundlagen und Methodik der 3D-Rekonstruktion und ihre Anwendung für landmarkenbasierte Selbstlokali-sierung humanoider Roboter.* Diplomarbeit, Universität Hamburg.

[Kre13] Kredel, R. (2013). *Dreidimensionale Bewegungserfassung mit Consumer-Highspeedkameras : Eine Systementwicklung unter besonderer Berücksichtigung der Messunsicherheit .* Konstanz, Univ.

[Küh11] Kühn, T. (2011). *The Kinect Sensor Platform.* Advances in Media Technology.

[Lav03] La Viola, J. J. (2003). *Double exponential smoothing: an alternative to Kalman filter-based predictive tracking. In: Proceedings of the workshop on Virtual environments.* New York, NY, USA: ACM.

[Liu73] Liu, C. L., & Layland, J. W. (1973). *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment.* Journal of the ACM 20.

[Mer07] Mergl, O. (2007). *Flexibilisierung von Baustrukturen durch Modularisierung zur Verbesserung des Nutzungspotenziales am Beispiel industrieller Produktionsstätten des Automobilbaus.* Kassel Univ. Pr,.

[Mer04] Merziger, G. (. (2004). *Formeln + Hilfen zur höheren Mathematik. 4. Aufl.* Springe.

[Mic08] Microchip. (2008). *MCP23018 / MCP23S18 – 16-Bit I/O Expander with Open-Drain Outputs.*

[Mic11] Microsoft. (2011). *http://www.microsoft.com/en-us/kinectforwindows/.* Microsoft (zugegriffen am 11.10.2014).

[Neu04] Neugebauer, R. (2004). *Parallel kinematic machines in research and practice : The 4th Chemnitz Parallel Kinematics Seminar.* Chemnitz: PKS.

[New11] Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., et al. (2011). *Kinect Fusion: Real-Time Dense Surface Mapping and Tracking.* Microsoft Research.

[Nie89] Niehhaus, H. A. *A fast square rooter combining algorithmic and lookup table techniques.* 1989: IEEE Southeastcon.

[Nüc06] Nüchter, A., Lingemann, K., & Hertzberg, J. (2006). *6D SLAM-3D Mapping Outdoor Enviroments.* Wiley InterScience.

[Nüh98] Nühermann, D. (1998). *Das große Werkbuch Elektronik. 7.* Poing : Franzis.

[Nvi14] Nvidia. (2014). *http://www.nvidia.de/object/cuda-parallel-computing-de.html.* (zugegriffen am 20.09.2014).

[Nxp11] NXP, B. (2011). *LPC1769/68/67/66/65/64/63 – Product data sheet. Rev. 8.*

[Nxp10] NXP, B. (2010). *UM10360 – LPC17xx User manual. Rev. 2.*

[Mar06] Osterloh, M. (2006). *Das Management von Strukturen und Prozessen.* Universität Zürich.

[Pau84] Paul, R. P. (1984). *Robot manipulators. Mathematics, programming, and control. the computer control of robot manipulators.* MIT Press series in artifical intelligence).

[Pfe87] Pfeiffer, F., & Reithmeier, E. (1987). *Roboterdynamik.* Teubner-Studienbücher.

[Pie23] Pierce, G. W. (1923). *Piezoelectric crystal resonators and crystal oscillators applied to the precision calibration of wavemeters.* Proceedings of the American Academy.

[Pri09] Prisacariu, V., Reid, I., & fastHOG. (09). *a real-time GPU implementation of OHG.* Oxford: Technical Report No. 2310/09; University of Oxford.

[Rob10] Robotic Industries, A. (2010). *Robot Terms and Definitions.* http://www.robotics.org/.

[Sch07] Scherer, M. (2007). *Kinematik des vollvariablen Ventiltriebs UniValve : inverse Berechnung der Kurvengeometrien und kinematische Simulation .* Hamburg: Diplomica-Verl.

[Sed05] Sedgewick, R. (2005). *Algorithmen in C.* München, PearsonStudium.

[Sho11] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., et al. (2011). *Real-time human pose recognition in parts from single depth images.* Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR).

[Sim03] Simon, C. *The Bip Buffer - The Circular Buffer with a Twist.* 2003.

[Ste01] Stefano Brusoni, A. P. (2001). *Unpacking the black box of modularity: Technologies, products and organizations.* Industrial and Corporate Change.

[Ste11] Steven W. Smith, P. (2011). *The Scientist and Engineer's Guide to Digital Signal Processing.*

[Sto11] Stowers, J., Hayes, M., & Bainbridge-Smith, A. (2011). *Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor.* Proc. IEEE Int Mechatronics (ICM) Conf.

[Tom98] Tomasi, C., & Manduchi, R. (1998). *Bilateral Filtering for Gray and Color Images.* IEEE International Conference on Computer Vision.

[Tön10] Tönnis, M. (2010). *Augmented Reality.* Springer-Verlag.

[Toy12] Toyota, T. (2012). *Toyota robot Humanoid and Robina.* http://www.toyota-global.com/innovation/partner_robot/family_2.html#h202,.2012. – zugegriffen am 30.11.2012.

[VDI2860] Verein Deutscher Ingenieure VDI 2860. (1990). *Montage- und Handhabungstechnik; Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbole.* VDI.

[Wan03] Wang, C., Thorpe, C., & Thrun, S. (2003). *Online Simultaneous Localization and Mapping with Detec-tion and Tracking of Moving Objects: Theory and Results from Ground Vehicle in Crowd-ed Urban Areas.* IEEE.

[Was12] Waseda, U. (2012). *TWENDY-ONE humanoid robot.* http://twendyone.com/index_e.html,. 2012. – zugegriffen am 30.11.2012.

[Web09] Weber, W. (2009). *Industrieroboter. 2.* Leipzig im Carl-Hanser-Verlag.

[Wey04] Weyrich, T., Pauly, M., Keiser, R., Heinzle, S., Scandella, S., & Gross, M. (2004). *Post-processing of Scanned 3D Surface Data.* The Eurographics Association.

[Wör05] Wörn, H., & Brinkschule, U. (. (2005). *Echtzeitsysteme.* Springer-Verlag Berlin Heidelberg.

Lebenslauf

Persönliche Daten

Name	Habib Nasri
Geburtstag	20.07.1982, Sidi Bouzid, Tunesien
Staatsangehörigkeit	Tunesisch & Deutsch
Familienstand	Ledig

Schulischer Werdegang

10.98 - 10.01	„Lycées technique 9 Avril“ (Elite Gymnasium), Sidi Bouzid/Tunesien Erwerb des „Baccalauréat en Technique“ (Allgemeine Hochschulreife, Fachbereich Technische Wissenschaft)
10.94 - 10.98	„College Abou Baker el Gammoudi“(Gymnasium), Sidi Bouzid/Tunesien
10.88 - 10.94	„Ecole Primaire bir el baye“ (Grundschule), Sidi Bouzid/Tunesien

Akademischer Werdegang

10.03 - 10.10	Dipl. Ing. Maschinenbau an der Leibniz Universität Hannover Schwerpunktfächer im Bereich der Produktionstechnik: Technologie der Fertigungsverfahren Produkt-Engineering und Logistik
08.02 - 06.03	Deutschkurs bei „Inligua“, Hannover Erwerb der deutschen Sprachprüfung für den Hochschulzugang ausländischer Studienbewerber (DSH)
09.01 - 08.02	„Ecole preparatoire d'Ingenieur Technique El Manar“ (Ingenieurstudium), Tunis/Tunesien

Berufliche Zusatzqualifikationen

Seit 04.2011	Wissenschaftlicher Mitarbeiter Im Institut für Produktentwicklung und Gerätebau, Leibniz Universität Hannover am Fachgebiet Rechner gestützte Produktentwicklung.
09.05 - 08.10	Aushilfe in der Firma „Beta Warenhandel GMBH & Co. KG“
10.06 - 10.08	Studentische Hilfskraft im Institut für Fertigungstechnik und Werkzeugmaschinen Projekt: „KosePro“
08.05 - 08.06	Studentische Hilfskraft im Institut für Integrierte Produktion Projekt: Beherrschung von Technologieänderung am Beispiel der Schmiedeindustrie