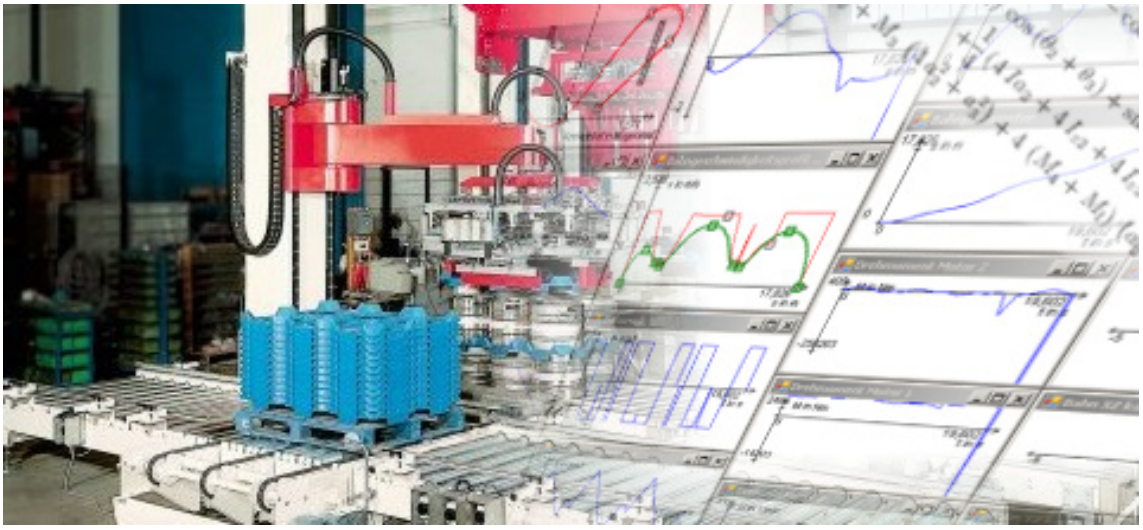
AG ROBOTERSYSTEME
FACHBEREICH INFORMATIK
AN DER UNIVERSITÄT KAISERSLAUTERN

Diplomarbeit



Zeitoptimale Bahnplanung für Industrieroboter

Ralf Kretschmann

18. Dezember 2007

Diplomarbeit

Zeitoptimale Bahnplanung für Industrieroboter

Arbeitsgruppe Robotersysteme
Fachbereich Informatik
UNIVERSITÄT KAISERSLAUTERN

Ralf Kretschmann
Studiengang Technoinformatik

Tag der Ausgabe : 27. Mai 2007
Tag der Abgabe : 18. Dezember 2007

Betreuer : Prof. Dr. Karsten Berns

Ich erkläre hiermit, die vorliegende Diplomarbeit selbständig verfasst zu haben. Die verwendeten Quellen und Hilfsmittel sind im Text kenntlich gemacht und im Literaturverzeichnis vollständig aufgeführt.

Kaiserslautern, den 18. Dezember 2007

(Ralf Kretschmann)

Vorwort

Die vorliegende Diplomarbeit entstand in der Zeit von 27.05.2007 bis 18.12.2007 im Competence Center der Elektro-Konstruktion der Firma KHS AG in Worms.

Besonderen Dank gebührt Herrn Prof. Dr. Karsten Berns für die Unterstützung und Betreuung der Diplomarbeit seitens der Technischen Universität Kaiserslautern. Durch seine Kooperationsfreudigkeit war es mir möglich, mit dieser Arbeit meine theoretischen Kenntnisse in der Industrie anzuwenden und sie um praktische Erfahrungen zu ergänzen.

Mein besonderer Dank gilt weiterhin für meine Betreuer Herrn Klaus Rau und Herrn Johann Mayer, die Leiter der Abteilungen Elektrische Konstruktion TBP-E bzw. Auftragsbezogene Elektrische Konstruktion PEK-E in der KHS Worms, die trotz angespannter Auftragslage viel Zeit und Geduld in mich investierten. Sie standen mir stets mit Rat und Tat zu Seite und verdeutlichten die Randbedingungen der praktischen Anwendbarkeit, ließen mir während der Arbeit jedoch viel Freiraum für eigene Ideen.

Meinen Dank möchte ich auch allen Kollegen aus der Abteilung, die mir bei technischen Fragen immer halfen, aussprechen. In gleichem Maße geht mein Dank an meinen Freundeskreis, der mich beim Korrekturlesen unterstütze, sowie an meine Familie, die mir während dieser Arbeit und auch des Studiums privat den Rücken freihielt.

Kaiserslautern, im Dezember 2007

Ralf Kretschmann

Inhaltsverzeichnis

Notationen

Die folgenden Vereinbarungen und Notationen werden in der Arbeit verwendet:

$(\vec{\ })$	Vektor
$(\underline{\ })$	Matrix
$(\dot{\ })$	zeitliche Ableitung
∂	partielle Differentiation
$\vec{c}$	Konfiguration
m	Drehmoment
q	generalisierte Koordinaten (unabhängig von Robotergeometrie)
s	Bahnparameter (z. B. Bogenlänge)
v	Geschwindigkeit
x,y,z	Koordinaten im kartesischen Raum
$\underline{A}$	Denavit-Hartenberg-Matrix
$\mathbb{C}$	Konfigurationsraum
$\underline{I}$	Trägheitsmatrix
L	Lagrangesche Energiedifferenzfunktion
M	Masse
$\mathcal{O}$	Komplexitätsklasse $f \in \mathcal{O}(g)$ bedeutet: f wächst höchstens so schnell wie g
Q	generalisierte Kraft
$\underline{R}$	Rotationsmatrix
S	Schwerpunkt
$\underline{T}$	Translationsmatrix
$\mathbb{W}$	Arbeitsraum
$\alpha, \beta, \gamma, \delta$	allgemeine Winkel
θ	Gelenkwinkel
ϕ, θ, ψ	Raumwinkel: Orientierung der Roboterhand

Indizes

a	Antrieb
g	Getriebe
i,j,k	allgemeine Indizes
kin	kinetisch
l	Last
m	Motor
pot	potentiell
r	Reibung
s	Schwerpunkt

Abkürzungen

CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CNC	Computerized Numerical Control
CP	Continous Path
DH	Denavit-Hartenberg Methode der Koordinatentranformation der Kinematik
DSE	Digitale Servoelektronik Leistungselektronik für die Aktuatoren
FBS	Funktionsbaustein-Sprache grafische SPS-Programmiersprache
GUI	Graphical User Interface
HMI	Human Machine Interface meist grafisches Bedienfeld des Roboters oder der Anlage
KHS	Klöckner Holstein Seitz
KRC	Kuka Robot Control Kuka Kern-System mit Steuerungssoftware für Kuka-Roboter
KMC	Kuka Motion Control Kuka CNC-Kern-System für individuelle Kinematiken
KCP	Kuka Control Panel Handbediengerät für Kuka Bahnplanung und Roboter
KRL	Kuka Robot Language Programmiersprache für KRC-Bahnplanung
MultiProg	IDE von Kuka und KW-Software für KMC SPS-Programme
OPC	OLE for Process Control Kommunikation zwischen Prozess- und Office-Systemen
ProConOS	Betriebssystem für Soft-SPS
PTP	Point-to-Point
RK	Knickarmroboter
RS	KHS-Hubsäulenroboter
SCARA	Selective Compliance Assembly Robot Arm
Soft-SPS	Software-Speicherprogrammierbare Steuerung auf PCs lauffähige Robotersteuerung
ST	Strukturierter Text an Hochsprachen angelehnte SPS-Programmiersprache
TCP	Tool Center Point
XM	Extended Motion KMC-Erweiterung mit flexibler Steuerung von Zusatzachsen

Übersicht

Die steigende Nachfrage nach individuellen Lösungen in der Automatisierungstechnik stellt die Entwickler von Bewegungsabläufen industrieller Maschinen vor großen Herausforderungen. Die Roboterbahnen sollen einerseits ohne hohen Programmieraufwand und langer empirischer Optimierung erstellt werden, andererseits müssen sie hohe Taktzahlen ermöglichen und individuelle Randbedingungen wie die Begrenzung des Rucks einhalten.

Die vorliegende Arbeit befasst sich mit Verfahren zur Bahnplanung und -optimierung im Kontext industrieller Handhabungsrobotern in Verpackungsanlagen. Im Fokus steht eine automatisierte Bahnplanung, die auf der Robotersteuerung aus Hindernisbeschreibungen eine kollisionsfreie Bahn erzeugt und diese direkt in die Bewegung umsetzt. Auf Basis dieser Bahn wird ein zeitoptimierter Geschwindigkeitsverlauf erstellt. Da die Geschwindigkeit von Randbedingungen des Roboters abhängt, werden zunächst die Roboterkinematik und -dynamik modelliert. Die dynamische Bewegungsgleichung fließt in die Wahl der Geschwindigkeiten ein. Für die Fahrt einer berechneten Trajektorie wird ein Bahnregler erstellt und die Verfahren so optimiert, dass sie mit den begrenzten Ressourcen einer SPS lauffähig sind.

Die vorgestellten Verfahren wurden an Industrierobotern der Fa. KHS mit der Steuerung KRC2 der Fa. Kuka erprobt. Versuche an den Maschinen wiesen nach, das die Ziele der Arbeit und die Anforderungen erfüllt wurden.

1. Einleitung

Moderne Fertigungsprozesse basieren seit Jahrzehnten auf Industrierobotern. Sie entgegnen den steigenden Ansprüchen nach Automatisierung, Qualitätssicherung und Flexibilität.

Nach Erhebungen der International Federation of Robotics (IFR) waren im Jahr 2004 weltweit 2 Mio. existierende Roboter im Einsatz, wovon die Industrieroboter 2005 einen Anteil von 770000 Stück aus machten. Hauptanwender von Industrieroboter sind Japan, Deutschland und die USA. Obwohl die aktuellen Investitionen der Automobilindustrie, welche mit ihren hohen Stückzahlen die Statistik am stärksten prägt, zurückhaltend sind, steigt die Nachfrage aus anderen Industriebereichen wie Kunststoff- und Gummiindustrie, Nahrungsmittel- und Verpackungsindustrie, Haushaltsgeräteindustrie, Holz- und Möbelindustrie, Glas- und Keramikprodukteindustrie. Die Zahl der Roboter in diesen Branchen zeigte 2006 einen Zuwachs von 25%. Für die Jahre 2008 bis 2010 soll nach Schätzungen der IFR das Wachstum des Gesamtbestandes der Industrieroboter durchschnittlich 4% betragen und bis 2010 zu 1,2 Mio. eingesetzten Maschinen führen.

Der praktische Ansatzpunkt dieser Arbeit sind die Industrieroboter aus der Nahrungsmittel- und Verpackungsindustrie der KHS AG. In dieser Arbeit ist der Begriff Industrieroboter an [Christaller 03] angelehnt:

Definition 1 (Industrieroboter) *Ein **Industrieroboter** ist ein universell einsetzbarer, sensumotorischer Bewegungsautomat basierend auf einer seriellen oder parallelen kinematischen Kette.*
Er besteht aus mechatronischen Komponenten, Sensoren und rechnerbasierten Kontroll- und Steuerungsfunktionen. Die Komplexität eines Industrieroboters unterscheidet sich deutlich von anderen Maschinen durch die größere Anzahl von Freiheitsgraden und die Vielfalt und den Umfang seiner Verhaltensformen.

Nach der Definition 1 ist weder eine mit Lichtschranken ausgestattete Rollenbahn (keine kinematische Kette) noch eine kontinuierlich laufende Abfüllstation (keine

Vielfalt der Verhaltensformen) ein Roboter.

Industrieroboter können zur Durchführung von Handhabungsaufgaben wie dem Greifen und Transportieren von Kisten eingesetzt werden. Die Roboterhand (Tool Center Point, TCP) ist dabei auf einer vorgegeben, kartesischen Trajektorie (CP) exakt zu bewegen. Alternativ wird bei der Punkt-zu-Punkt-Fahrt (PTP) eine Fahrt über vorgegebene Stützpunkte durchgeführt, wobei die Bahnkontur zwischen den Punkten beliebig sein darf. Diese Stützpunkte sind in der korrekten Reihenfolge und überschwingungsfrei zu erreichen.

Im Gegensatz zu PTP-Steuerungen muss bei der CP-Steuerung die Bahnkontur definiert und während der Fahrt abrufbar sein. Dies wird im Allgemeinen durch eine Transformation der kartesischen Bahn in den Raum der Gelenkwinkel erreicht. Zudem muss die Roboterhand dem in der Trajektorie enthaltenen Geschwindigkeitsprofil genau folgen.

Ein Bahnplanungssystem kann in die Funktionen der Bahnplanung mit Kollisionsvermeidung, der Erstellung der geometrischen Bahnkontur, der Berechnung des Geschwindigkeitsprofils und der Regelung zur Fahrt über die Trajektorie zerlegt werden. Diese Funktionen sollen zwar für die einfache, modularisierte Umsetzung getrennt betrachtet werden, die Optimierung fordert jedoch die Betrachtung der Komponenten im Kontext des Gesamtsystems.

Die Bahnplanung erzeugt aus der Beschreibung von Hindernissen die Stützpunkte für die kollisionsfreie geometrische Raumkurve. Auf Basis dieser Bahnbeschreibung kann das Zeitprofil berechnet werden. Falls die Berechnung die kinematischen und dynamischen Randbedingungen des Roboters berücksichtigt, werden nur solche Trajektorien erzeugt, die die Robotermechanik ohne Bahnfehler fahren kann. Wird zudem der Anstieg der Beschleunigung begrenzt, entstehen ruckbegrenzte Bewegungen zum schonenden Transport einer sensiblen Last wie Glasflaschen. Die Bahnregelung überträgt die Gelenkstellungen aus der Bahn an die Antriebssteuerung.

1.1 Motivation

Kommerzielle Robotersteuerungen bieten oft wenige oder umständliche Möglichkeiten, den Roboter auf einer beliebigen Trajektorie zu bewegen. Es können im CP-Betrieb Standardbahnen wie Kreise oder Geraden gefahren werden. Eine PTP-Steuerung erlaubt zwar das Folgen beliebiger Bahnen, aber diese sind nur über wenige Stützpunkte definiert. Soll der Bahnfehler möglichst gering sein, kann der PTP-Betrieb nicht verwendet werden.

Kommerzielle Systeme eilen dem Stand der Forschung hinterher. In wissenschaftlichen Veröffentlichungen wird oft ein idealisiertes und optimales Robotersystem als Basis der weiteren Untersuchungen zugrunde gelegt. Diese konzentrieren sich auf Teilaspekte wie Kollisionsvermeidung, Bahnoptimierung oder Sensorintegration und stützen sich auf theoretische Analysen oder Simulationen. Folglich ist die praktische Umsetzung der theoretischen Erkenntnisse mit Hindernissen konfrontiert.

Die wissenschaftliche Fragestellung dieser Arbeit zielt auf ein optimiertes und implementierbares Bahnplanungssystem ab. Dabei soll untersucht werden,

- welche Ansätze für die Lösung von Teilproblemen bereits existieren,
- wie diese Modelle integriert werden können und
- unter welchen Randbedingungen diese praktisch auf einen Roboter übertragbar sind.

Aus praktischer Sicht soll die vorhandene Einzelachssteuerung der KHS-Roboter in eine Mehrachssteuerung überführt werden. Bei KHS Anlagen kommen Knickarmroboter und Steuerungen der Kuka AG zum Einsatz. Die Roboterarme müssen mit selbst entwickelten KHS-Kinematiken zusammenarbeiten, die z. B. das Packen und Beladen der Paletten vornehmen. Die für diese Roboter ebenfalls verwendbare Kuka-Steuerung stammt aus dem CNC/CAM-Bereich (Computerized Numerical Control/Computer Aided Manufacturing) und hat keine kinematische Beschreibung der KHS-Roboter, so dass die Trajektorien für jeden Roboter erneut per Hand berechnet, aufwändig programmiert und empirisch optimiert werden müssen. Die Antriebe werden in Einzelachssteuerung getrennt voneinander geregelt. Eine Robotermodellierung existiert nicht und die Kollisionsvermeidung muss manuell durchgeführt werden.

Die Defizite des aktuellen Vorgehens der KHS mit Einzelachssteuerung sind

- hoher Programmieraufwand zur Erstellung der Bewegung
- das manuelle Programmieren von Bahnen erhöht die Zahl möglicher Programmierfehler
- die Erstellung von guten Bewegungsabläufen basiert hauptsächlich auf Erfahrung der Softwareentwickler und auf empirischer Optimierung

- die Leistungsgrenzen der Maschinen können nicht ausgereizt werden, weil Maschineneigenschaften vernachlässigt werden.

Diese Probleme können durch ein Bahnplanungssystem behoben werden.

1.2 Ziel der Arbeit

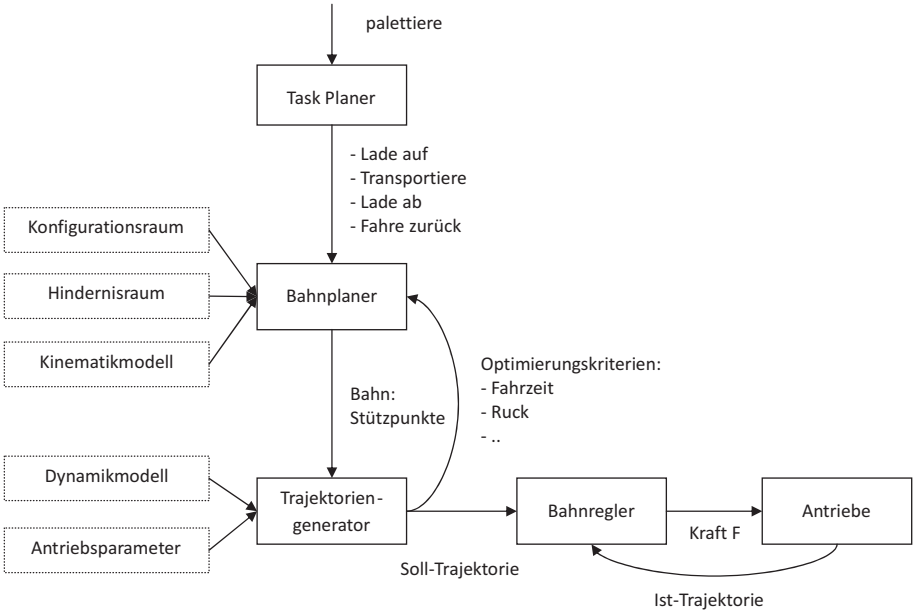


Abbildung 1.1: Bahnplanungssystem für Industrieroboter

Das Ziel dieser Arbeit ist die Entwicklung eines Bahnplanungssystems und seine experimentelle Implementierung auf einem KHS-Roboter unter der Verwendung einer hardwareseitigen Einzelachssteuerung. Zur Entwicklung des Systems müssen geeignete Verfahren der Bahnplanung, -optimierung und -regelung im Kontext des Gesamtsystems untersucht werden. Folgende Punkte sollen dabei berücksichtigt werden:

- Überführung der bisherigen Einzelachssteuerung in eine Mehrachssteuerung, so dass eine Bahn mit Stützpunkten beschrieben werden kann
- Geschwindigkeitsoptimierung der Trajektorie
- Beachtung von Benutzervorgaben wie begrenzter Bahnruck und mechanischen Randbedingungen wie maximale Motordrehmomente
- Untersuchung verschiedener Verfahren zur Bahnplanung im Arbeitsraum mit Hindernissen
- Implementierung der Steuerung auf einer SPS unter Berücksichtigung ihrer begrenzten Ressourcen

Abbildung 1.1 zeigt die Gesamtstruktur des Bahnplanungssystems. Aus dem jeweiligen Tasks ergeben sich Zielpunkte, die über die Bahnplanung kollisionsfrei erreicht werden sollen. Der geometrische Verlauf der Sollbahn kann mit Hilfe des Bahnplanungssystems automatisch berechnet oder vom Benutzer programmiert werden. Der Anwender soll von der Bahnkontur losgelöst das Geschwindigkeitsprofil programmieren oder automatisch optimieren können.

Die Erweiterung der KHS-Maschinensoftware um ein Bahnplanungssystem soll folgende praktischen Konsequenzen aufzeigen:

- Reduktion von Planungskosten aufgrund der Zeitersparnis bei der Programmierung des Bewegungsablaufs
- Betrachtung des Bahnplanungssystems als wiederverwendbare Black-Box: Bahnen bei neuen Anwendungen parametrieren statt programmieren
- Verringerung des Aufwands durch Simulationen und weniger empirische Optimierung an der Maschine
- Verbesserung des Maschinenverschleiß' durch glatte Bahnen
- höhere Taktzahlen durch eine automatisierte Optimierung

Für die experimentelle Untersuchung stand ein KHS RS3-Roboter und eine PC-basierte SPS-Steuerung Kuka KR C2 zur Verfügung. Die Entwicklung wurde auf einem PC mit den entsprechenden Software-Werkzeugen durchgeführt.

1.3 Aufbau der Arbeit

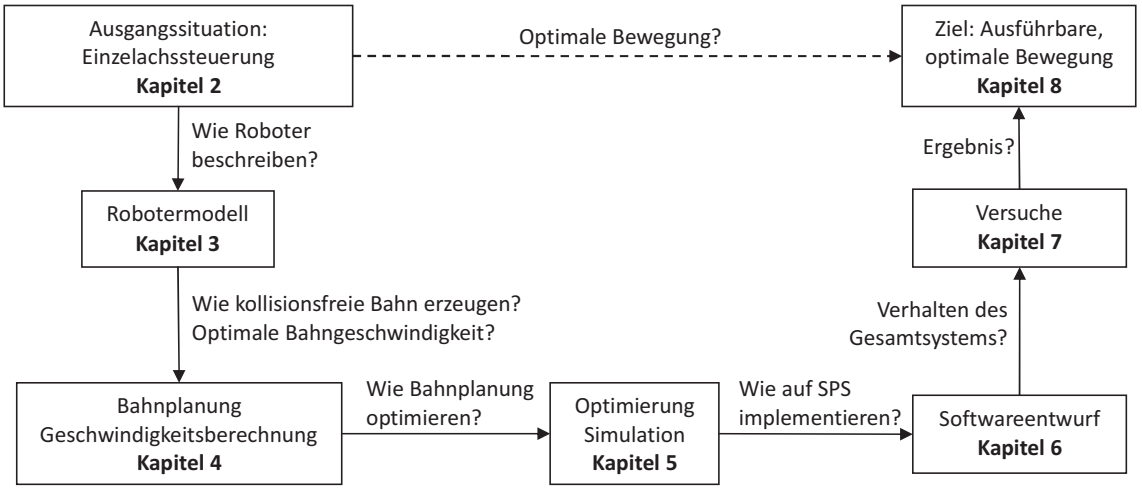


Abbildung 1.2: Konzept dieser Arbeit

Abbildung 1.2 ordnet den Kapiteln die einzelnen Fragestellungen und Entwicklungsschritte zur Lösung der Gesamtaufgabe zu. Die Kapitel besitzen folgen Inhalt:

Kapitel 2 geht auf die verwendete Technologie der Firma KHS ein und skizziert ihre Anwendungsgebiete. Es beschreibt den Stand der Technik der Industriemaschinen und stellt die Schwierigkeiten bei der Erstellung guter Bewegungsabläufe mit der aktuellen Einzelachssteuerung der Roboter dar. Es führt abschließend zu den Randbedingungen dieser Arbeit.

Das Ziel der Arbeit ist die Erstellung von zeitoptimalen Bahnen. Diese sind jedoch roboterspezifisch und setzen eine mathematische Beschreibung des Roboters voraus. Als Ausgangspunkt für die spätere Bahnoptimierung befasst sich **Kapitel 3** mit der Modellierung der Mechanik, Kinematik und Dynamik von Industrierobotern. Diese Modellierung wird auf eine SCARA-Kinematik und die KHS-Roboter angewandt.

Um den Roboter im Arbeitsraum mit Hindernissen kollisionsfrei zu bewegen, werden anschließend unterschiedliche Verfahren zur Bahnplanung und -interpolation in **Kapitel 4** auf ihre Anwendbarkeit für Industrieroboter untersucht. Mit den Ergebnissen der Robotermodellierung wird zu einer gegebenen Bahn die optimale Geschwindigkeit berechnet. Um die Abfahrt der gefundenen Trajektorie unter Angaben weiterer Randbedingungen zu ermöglichen, wird eine Bahnregelung entwickelt.

Kapitel 5 geht der Frage nach, mit welchen Optimierungsmechanismen die Fahrt weiter beschleunigt und die Modelle effizienter für die Implementierung auf einer SPS gestaltet werden können. Außerdem wird eine Robotersimulation vorgestellt, welche die Effekte der Bahnplanung aufzeigt und die Optimierung belegt.

Kapitel 6 beschäftigt sich mit der Übertragung der Modelle auf die Robotersteuerung. Es beschreibt den Entwicklungsprozess der Software und geht auf hardwarespezifische Randbedingungen ein.

Die praktische Anwendbarkeit der Bahnplanung wird in **Kapitel 7** erprobt. Durch Versuche mit einem Testroboter werden die Leistungsfähigkeit und die Einschränkungen der Bahnplanung bewertet.

Das Ergebnis der Arbeit und ein Ausblick auf zukünftige Arbeiten findet sich abschließend in **Kapitel 8**

Rechnungen, Programmdokumentation und die Beschreibung der Simulation sind im Anhang dargestellt.

2. Stand der Technik in der KHS

Die Firma KHS AG ist ein internationaler Anlagenbauer. Dieses Kapitel gibt einen Einblick in die Tätigkeit und die verwendeten Technologien der KHS. Es beschreibt, welche Roboter verwendet werden und stellt dar, wo Verbesserungspotential herrscht und liefert die Motivation für diese Arbeit.

Die KHS betätigt sich schwerpunktmäßig in verschiedenen Sektoren der Getränketechnik und auch in der Food- und Non-Food-Industrie. **Abschnitt 2.1** geht auf das Tätigkeitsfeld der KHS ein.

Abschnitt 2.2 beschreibt unterschiedliche Varianten der in KHS-Anlagen eingesetzten Robotern und Maschinen und führt die Maschinen ein, die Gegenstand dieser Arbeit sind.

Abschnitt 2.3 beschäftigt sich mit den Unterschieden zwischen Industrierobotern und aktuellen Forschungsprojekten der Robotik anhand eines konkreten Beispiels und begründet den Entwicklungsprozess und die Steuerungsarchitektur von Industrierobotern.

Die Steuerungsarchitektur beeinflusst maßgeblich die Freiheitsgrade der Softwareentwicklung: Die zu erstellende Software muss in den Rahmen der verwendeten Hardware, aber auch zu Betriebs- und Kommunikationssystemen passen. **Abschnitt 2.4** beleuchtet die Eigenschaften der KHS-Hardware und -Software.

Abschnitt 2.5 beschreibt die verwendeten Werkzeuge zur Softwareerstellung und die Entwicklungsprozesse.

Abschnitt 2.6 fasst die Abschnitte zusammen und stellt Eigenschaften dar, die mit dieser Arbeit verbessert werden können.

2.1 Tätigkeitsfeld und Anlagen der KHS

Der KHS Konzern mit dem Firmensitz in Dortmund und den Standorten aus Abbildung 2.1 ist mit 752 Mio. EUR Umsatz und 4681 Mitarbeitern im Jahr 2006 einer der führenden Anlagenbauer auf dem Gebiet der Verpackungs- und Abfülltechnik [KHS 06]. Die KHS AG konzentriert sich auf die Erstellung von Individualanlagen und das Betätigungsfeld umreißt die Themen Prozesstechnik, Abfüllung, Hygiene, Keg (Mehrweg-Fass), Reinigung, Inspektion, Etikettierung und Pasteurisierung.

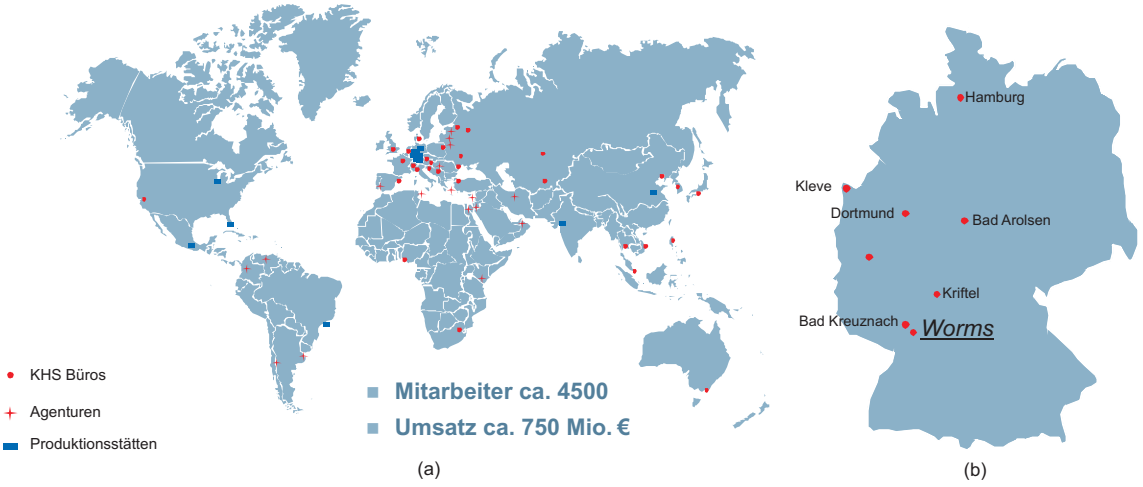


Abbildung 2.1: a) Weltweite Aufstellung der KHS AG und b) Standorte in Deutschland, aus [Kuka 07b] Rau S. 5

2.1.1 Planung kompletter Anlagen

Abbildung 2.2: Anlage mit Gebindetransportstecken im Vordergrund, aus [KHS 07a] Logistik S. 3

Die Anlagenplanung beinhaltet nach [KHS 07a] kundenspezifische Vorgaben wie Ausbringung pro Produkt und Jahr, Behältertypen und -sorten, Gebindetypen und -sorten, Ausstattungsmerkmale (Etiketten) und Produktionspläne. Zu den Herausforderungen zählt das Abfangen maschinenbedingter Ablaufstörungen. Es soll eine hohe Verfügbarkeit sichergestellt werden, als auch ein größtmöglicher Wirkungsgrad und ein optimales Kosten-Nutzenverhältnis. Entsprechend der Störcharakterisitika

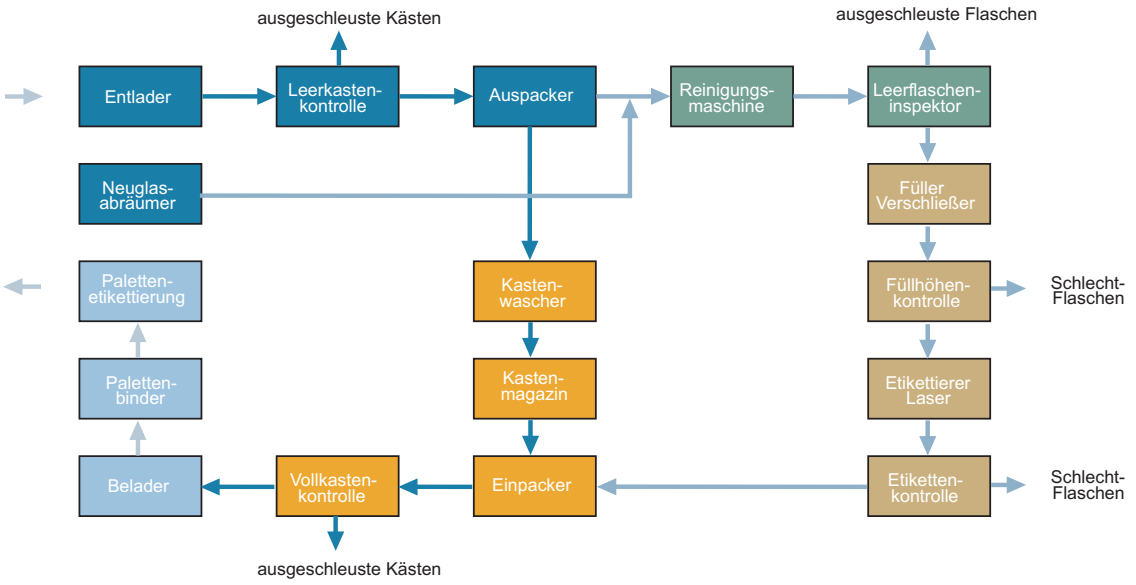


Abbildung 2.3: Stationen einer Abfülllinie, aus [KHS 07a] AIS S.7

der Anlage werden Pufferkapazitäten bestimmt und das Anlagenlayout erstellt. Es richtet sich an bestimmende Umfeldbedingungen wie die Ausbringungsvorgaben des Betreibers, die zur Verfügung stehende Produktionsfläche, die notwendige oder gewünschte Funktionalität und die gewünschten Einzelmaschinen. Eine KHS-Anlage aus Abbildung 2.2 kann grob in folgende Klassen eingeordnet werden: Mehrweg (Glas, PET), Einweg (Glas, PET, ACF, Zweiweg-PET, Dose), Sortierung (Sortierzentrale, Inline-Sortierung) und Sonstiges (Keg, Party-Keg, NGI), wobei multifunktionale Anlagen eine Mischung der Produkttypen zulassen.

Das Flussdiagramm aus Abbildung 2.3 stellt die Standardkomponenten und den Materialfluss einer Abfüllanlage dar. Gebrauchtes Mehrwegmaterial wird im Entlader von der Palette geholt, aus den Kästen ausgepackt und mit Neumaterial gemischt. Die Flaschen werden anschließend gereinigt, befüllt und etikettiert. Nach dem Einpacken in gereinigte Kästen wird das Material auf Paletten gestapelt. Mehrere Kontrollstationen überwachen die Produktionslinien und sortieren den Ausschuss aus.

Für die Anlagenoptimierung werden Simulationsmodelle eingesetzt. Sie bieten wie in Abbildung 2.4 gezeigt ein grobe Sicht auf die komplette Anlage sowie die Leistungsdaten der einzelnen Komponenten und zeigen Schwachstellen auf. Anlagenleistung, Logistik und die Lärmbelastung sind Gegenstand der Optimierung.

2.1.2 Einordnung des Standorts Worms

Diese Arbeit wurde in dem Bereich Pack- und Palettiertechnik in der Abteilung Elektrische Konstruktion / Competence Center an dem Standort Worms durchgeführt. Das Competence Center beschäftigt sich unter anderem mit der nichtauftragsspezifischen Entwicklung von Palettieranlagen, Packmaschinen, Gebinde- und Palettentransport. Es ist wie in Abbildung 2.5 dargestellt in der Projektierung von komplexen Anlagen involviert und dient als Ansprechpartner für den Vertrieb

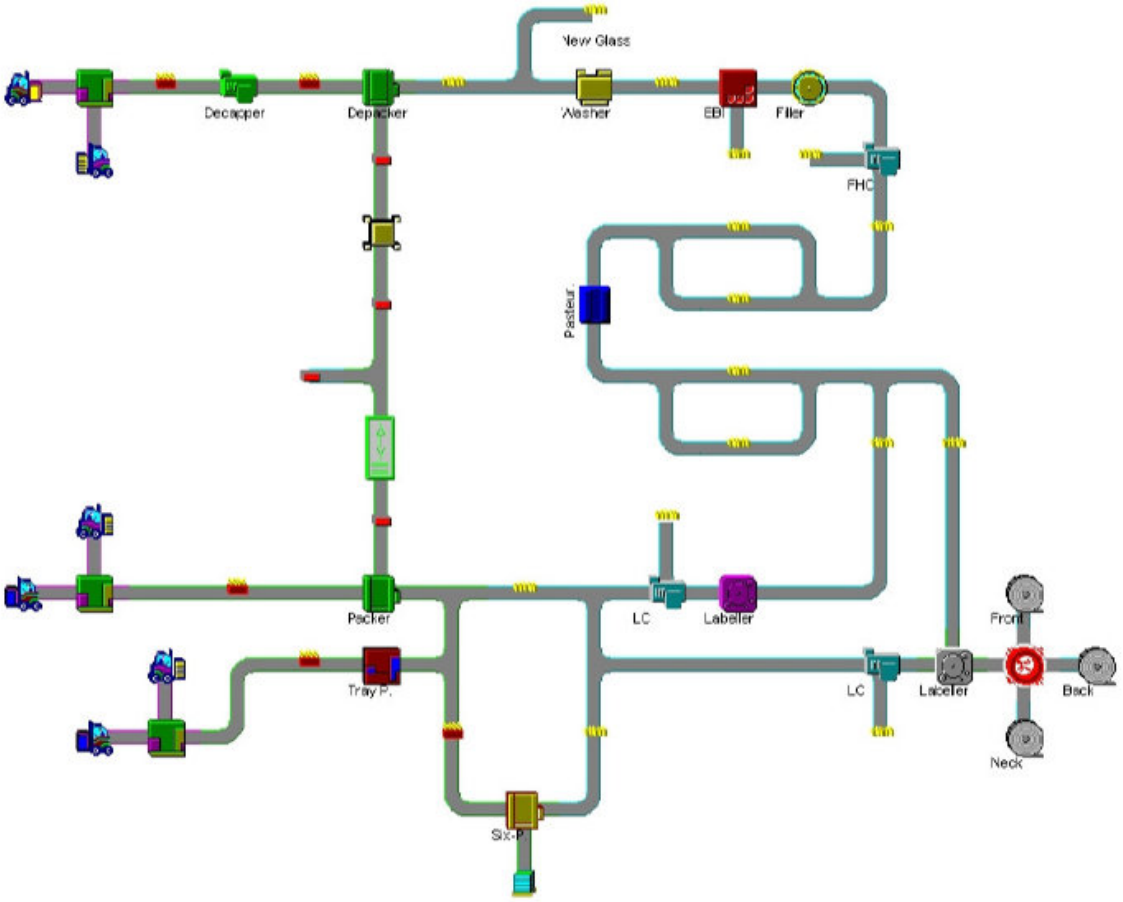


Abbildung 2.4: Simulation einer Anlage mit Pufferstrecken und den Stationen Abräumer, Deckel-Abschrauber, Wascher, Flaschen-Kontrolle, Füller, Füllhöhen-Kontrolle, Pasteurisierer, Etikettierer, Etiketten-Kontrolle, Packer und Belader, aus [KHS 07a] Anlagenoptimierung S. 18

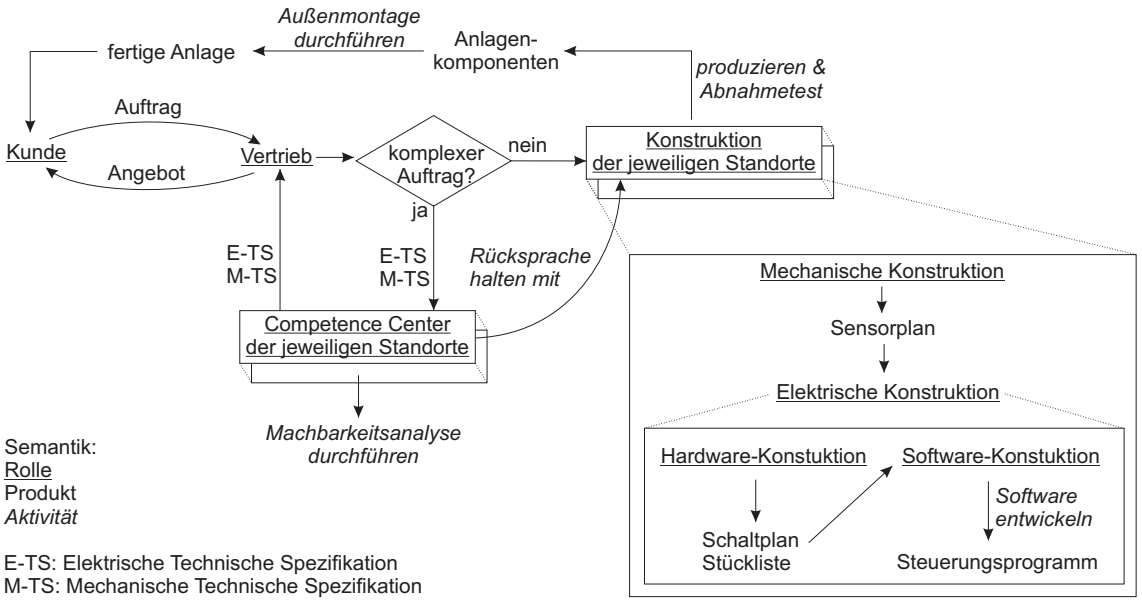


Abbildung 2.5: Vereinfachtes Prozessmodell der Anlagenentwicklung

bei Detailproblemen zu Standort-spezifischen Produkten.

Die in Worms hergestellten Anlagenkomponenten sind Trockenteile. Sie unterscheiden sich von den Nassteil-Komponenten anderer Standorte: Füller oder Wascher sind kontinuierlich laufende Maschinen. Dagegen sind Pack-, Palettier- und Handlingsmaschinen aus Worms meist Taktmaschinen. Sie führen diskrete Bewegungen aus und basieren auf Robotertechnologie. Diese Maschinen werden in den nächsten Abschnitten näher beschrieben.

2.2 Robotertypen und -anwendungen in Verpackungsanlagen

Für den Gebindetransport aus Abbildung 2.6 kommen je nach Einweg- oder Mehrweg-Typ unterschiedliche Transportmittel (Laufbänder), Drehvorrichtungen, Verteiler und Gruppierstationen zum Einsatz. Die Gebinde entstehen aus dem Einschrumpfen mehrerer Einweg-Flaschen mit Plastikfolie. Bei Mehrwegware werden Ein-/Auspackmodule verwendet und Kästen transportiert. Die KHS verwendet SPS-Systeme verschiedener Anbieter zur Steuerung der Anlagen. Für die Vielfalt an Aktuatoren und Roboter werden je nach technischen Anforderungen und Kundenwunsch Siemens S7, Rockwell oder eine Steuerung der Fa. Kuka eingesetzt. S7 hat in der Europa die größte Verbreitung, im amerikanischen Raum sind Rockwell-Steuerungen der Standard in der Automatisierungstechnik. Kuka hebt sich von der Konkurrenz durch eine leistungsfähige PC-basierte Steuerung, die u. a. eine SPS beinhaltet, hervor.
Die Ein-/Auspacker wie in Abbildung 2.7 dargestellt arbeiten mit Unterdruck- oder Klemmgreifern und besitzen teilweise eine Flaschenausrichtung, so dass das Etikett in Kisten eingepackter Flaschen stets von außen lesbar ist.
Verpackte Gebinde laufen vor der Pallettierung in eine Gruppierstation mit Einlauf

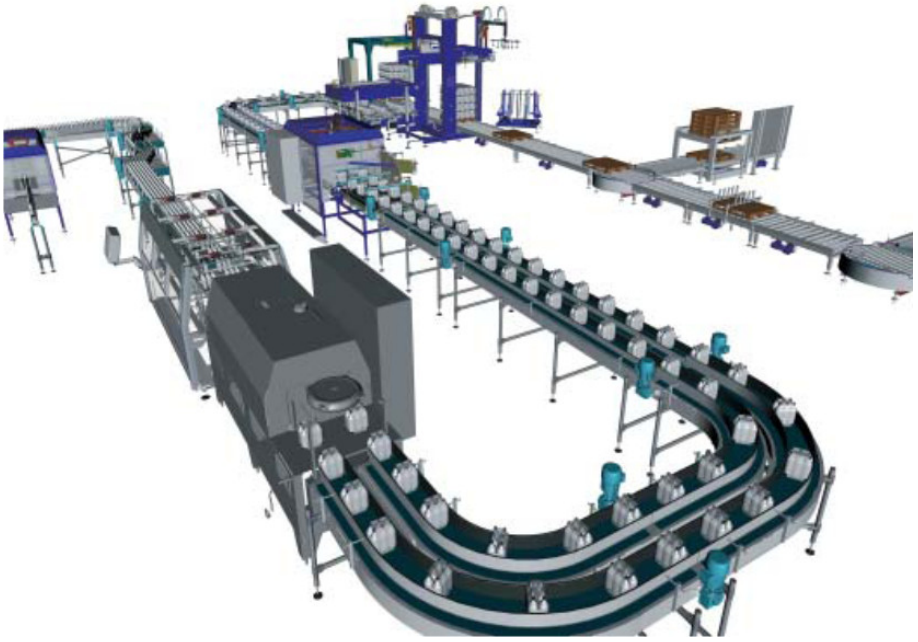


Abbildung 2.6: Schrumpftunnel und Transportanlage für Gebinde mit Palettiermodul am Ende, aus [Kuka 07b] Transportanlagen S. 6

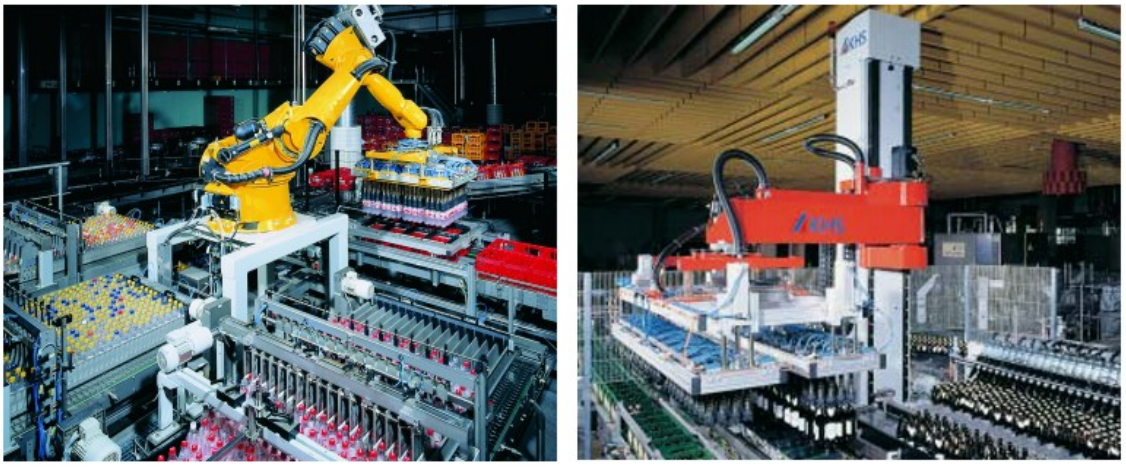


Abbildung 2.7: Einpacken von sortierten Flaschen mit Knickarmroboter des Typs Innopack RK (links) und Säulenroboter Innopack RS (rechts), aus [KHS 07a] Innopack S. 6, Innopack S. 16

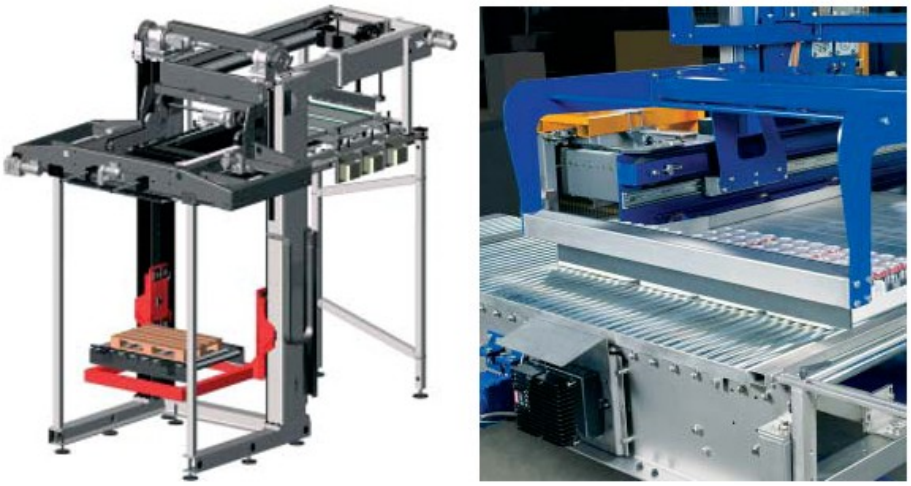


Abbildung 2.8: Gruppiermodul und Schieber, der aus einlaufenden Gebinden eine Lage für die Palettierung bildet, aus [KHS 07a] Innopal S. 7

Abbildung 2.9: Palettieren von Kästen und Gebinden mit einem Knickarmroboter Innopal RK mit Rollenteppich und Säulenzentrierung Innopal RSZ und Palettierung mit einem Säulenroboter Innopal RS, aus [KHS 07a] Innopal S. 18, Innopack S. 5

und Schieber wie in Abbildung 2.8 ein. Dort werden die Gebinde zu einer Lage formiert. Die Lagen besitzen unterschiedliche Anordnung der Gebinde, um sie stabil auf Paletten stapeln zu können. Nachdem Lagen formiert wurden, werden sie u. a. mit Palettierrobotern aus Abbildung 2.9 auf Paletten gestapelt.
Im Gegensatz zu von Krieger [Krieger 04] beschriebene Systeme mit Mensch-Roboter-Kooperation kann in Palettiermodulen auf aufwendige Sensorik verzichtet werden. Zur Erfassung der Umwelt werden Lichtschranken, Kameras und Antriebssensorik eingesetzt. Durch die Servotechnologie kann die Stellung der Motorachsen genau erfasst werden. Die wenigen Lichtschranken werden nur zur Positionsbestimmung von Material und aus Sicherheitsgründen eingesetzt. Eine Bildverarbeitung zur Lageerkennung wie in Abbildung 2.10 ist bei Handhabungsaufgaben zum Greifen der KEG-Fässern nötig. Je nach Anordnung der Fässer kann der Roboter ein Fass oder mehrere Fässer zusammen greifen und umsetzen. Sonst wird auf den Einsatz von Sensoren wegen der Fehleranfälligkeit und höheren Wartungskosten bewusst verzichtet.

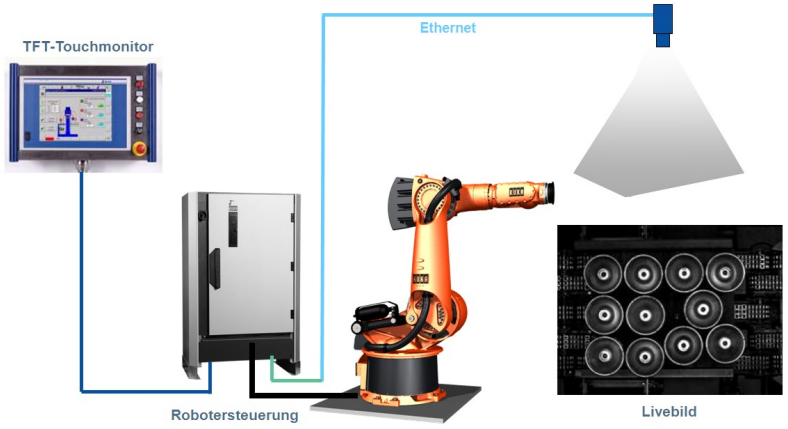


Abbildung 2.10: Kameragestütze Steuerung zum Ermitteln der Zugriffs- und Bewegungsführung für das Umsetzen von KEG-Fässern, aus [Kuka 07b] Rau S. 30

Maschinentyp	Traglast Nutzlast	Arbeitsbereich	Arbeitszyklen/h Arbeitstakt	Lagen/h bei Vollgut
RK 4-KR 180	210 kg 100 kg	360°	500-720 7,2-5,0 s	250-360 l/h
RK 6-KR 350/2	360 kg 190 kg	360°	425 8,5 s	425 l/h
RS 3 (Armlänge 1400)	500 kg 300 kg	180°	520 6,9 s	520 l/h
RS 3 (Armlänge 1600)	450 kg 250 kg	180°	450 8,0 s	450 l/h

Tabelle 2.1: Systemvergleich von Knickarm- (RK) und Säulenrobotern (RS) bei Kunststoffverarbeitung mit Hakengreifer und Glasflaschen mittels Packkopf, aus [KHS 07a]

Während des Betriebs dürfen sich keine Menschen im Arbeitsraum der KHS-Roboter aufhalten. Im allgemeinen können Bediener oder Arbeiter in industriellen Anlagen direkt mit Robotern zusammenarbeiten. Krieger [Krieger 04] befasst sich mit einem Schutzsystem für Menschen im industriellen Roboterarbeitsraum, der in überwachte Sicherheitszonen eingeteilt wird. Die Arbeitsraumüberwachung geschieht über Sensorik wie Laser-Scanner, Ultraschall und Lichtgitter. Als Koordinator wird ein Sicherheits-SPS eingesetzt. Das System kann nur wenige, einfache Ausweichbewegungen durchführen und enthält keine dynamische Bahnplanung.

Die Hubsäulenroboter können verschiedene Köpfe als Greifer aufnehmen. Der Kopfwechsel kann auch dynamisch geschehen. Abbildung 2.11 zeigt eine Auswahl. Der Lagenklemmkopf klemmt eine Lage mit Gebinden ein, der Packtulpenkopf arbeitet als Unterdruckgreifer oder Klemmgreifer, der Hakengreiferkopf kann sich in Kästen einhaken und der Rollenteppichkopf bekommt eine Lage seitlich eingeschoben und setzt sie durch Zurückziehen des Teppichs nach unten ab.
Tabelle 2.1 gibt eine Übersicht der Eigenschaften der Palettierroboter. Die Kuka Knickarmroboter RK können sich zwar schneller bewegen und sind wegen ihres grö-

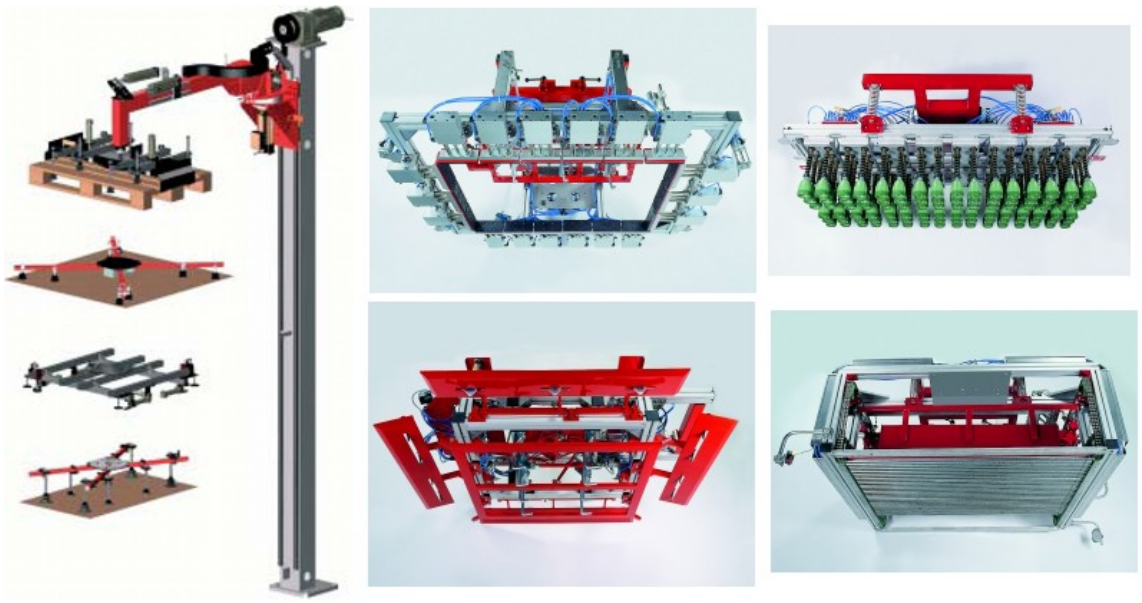


Abbildung 2.11: Säulenroboter mit austauschbaren Köpfen: Lagenklemmkopf, Packtulpenkopf, Hakengreiferkopf und Rollenteppichkopf, aus [KHS 07a] Innopal S. 23, Innopack S. 24-25

ßeren Arbeitsraums flexibler einsetzbar, aber die KHS Hubsäulenroboter RS können schwerere Lasten tragen und erreichen einen höheren Durchsatz an Material.

Die Abbildung 2.12 zeigt unterschiedliche Varianten von KHS-Standard-Maschinen. Palettenbelader (PB) laden bis zu 300 Lagen pro Stunde auf Paletten. Das Material kann aus Kunststoff-Getränkekisten, Kartons, Trays, Schrumpfgebinde, Einzelpäckchen oder Säcken bestehen. Der Niveauausgleich ist bei Anordnungen mit unterschiedlichen Höhen des Materialzulaufs (z. B. Gebindebahn, Gruppierstation) und des Ablaufs (z. B. Rollenbahn) erforderlich.
Die Abräumer (AS) schieben Neuglasartikel, PET-Artikel und Dosen mit maximaler Artikelhöhe von 400 mm von einer Palette ab. Die Maschinen wiegen bis zu 7 t und ihre Leistung beträgt bis zu 400 Lagen Dosen pro Stunde. Der Zwischenlagenabheber / -einleger mit einer Traglast von max. 50 kg hebt die Palettierungshilfen wie Zwischenlagen (Kunststoff, Pappe, etc.), Stülpdeckel, Paletten und Abdeckrahmen ab und legt sie in ein Magazin.
Ein- und Auspacker (SP) besitzen eine Leistung von bis zu 60000 Flaschen oder Gläser pro Stunde. Das Material wird in Kunststoffkästen oder Faltkartons verpackt bzw. aus den den Kästen ausgepackt. Für höhere Leistungen wird die Zweisäulen-Maschine (PPZ) mit 420 Arbeitszyklen bei zweibahnigem Materialablauf verwendet.
Die Kartonauffalter und -verschließer (CA, CV) sind den kontinuierlich laufenden Maschinen und nicht den Robotern zuzuordnen. Sie erreichen eine Leistung von 3300 Kartons pro Stunde.

Die Randbedingungen wie die Anordnung der Maschinen, Lasten und Hindernisse richten sich individuell an den Kundenanforderungen, den baulichen Gegebenheiten und dem Kundenmaterial aus. Ein Verpackungsmodul besteht aus Standardkom-

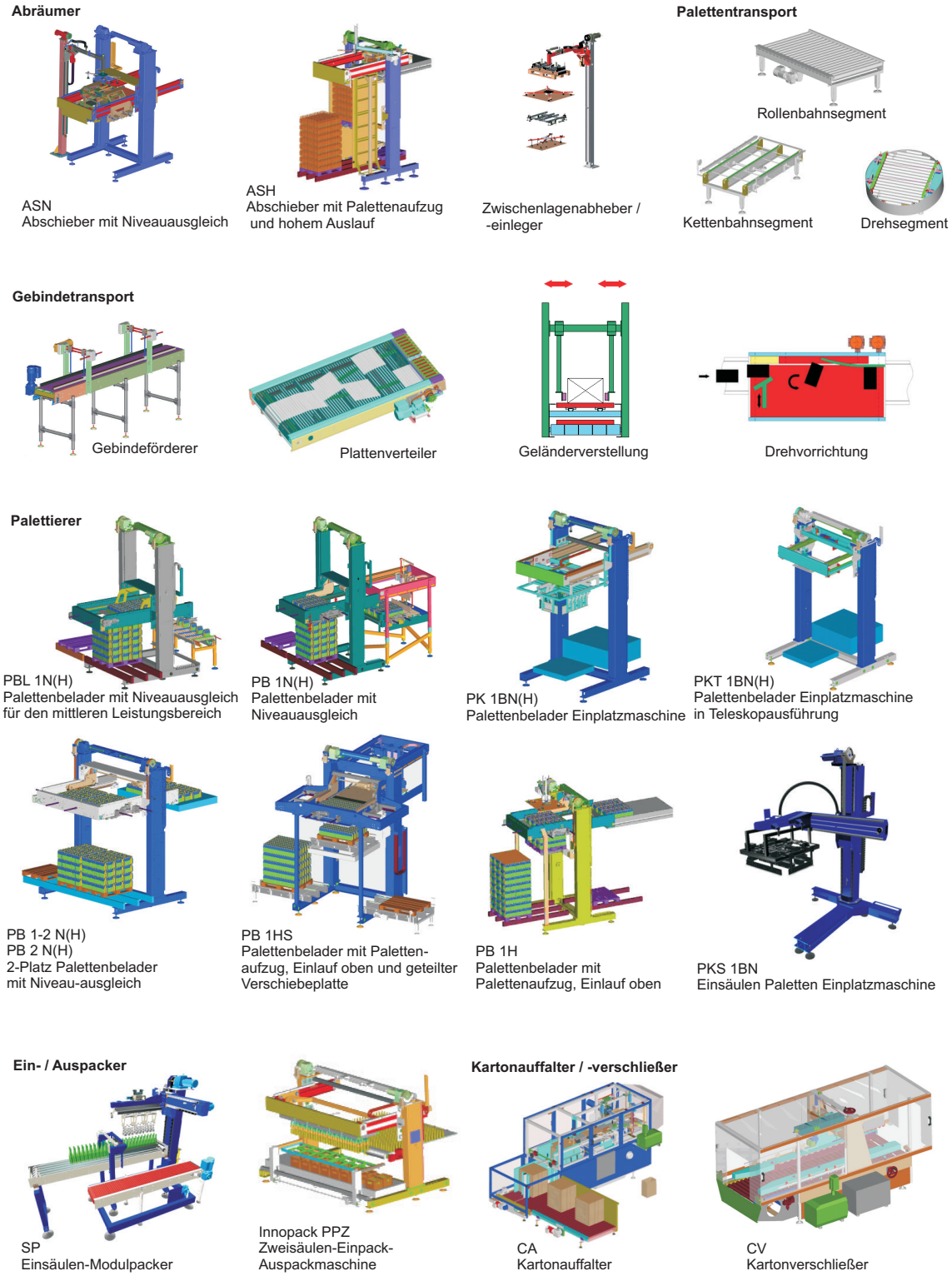


Abbildung 2.12: Auswahl an Standard-Palettier-, Pack- und Transportmaschinen der KHS Worms, aus [KHS 07b], [KHS 07c], [KHS 07a]

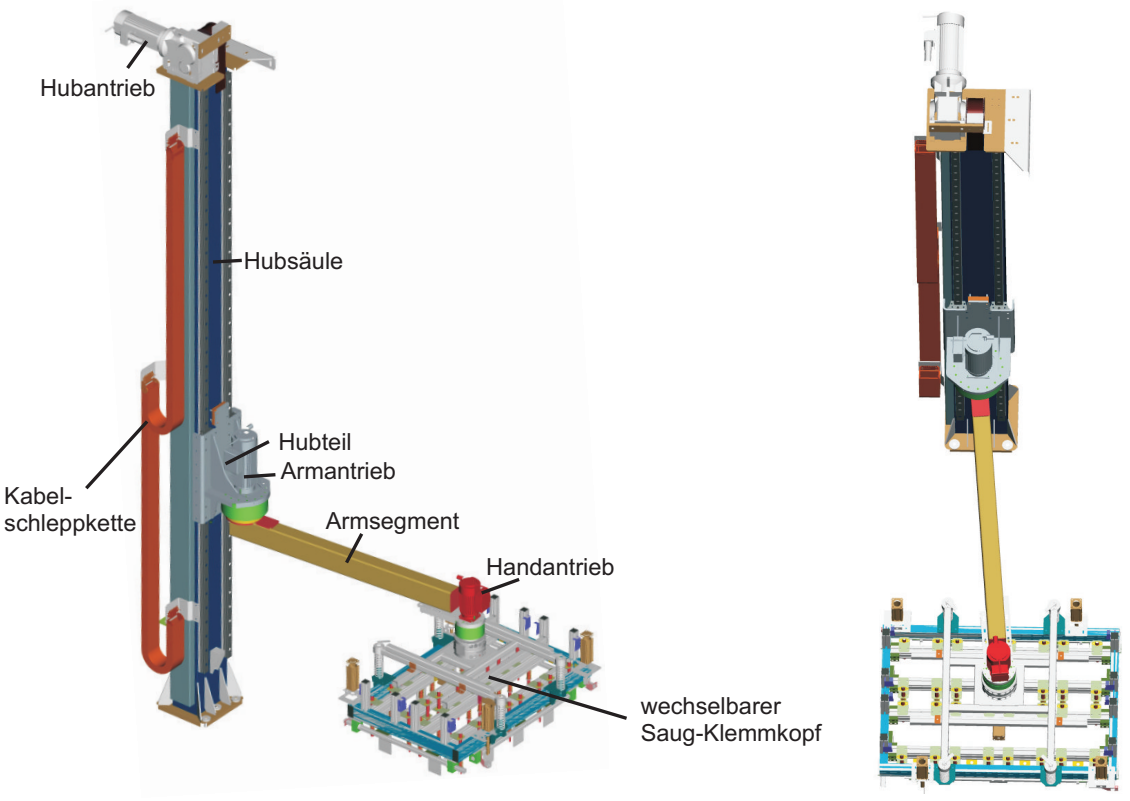


Abbildung 2.13: Geometrisches Modell des 3-achsigen Testroboters (von KHS)

ponenten, die für jeden Auftrag individuell kombiniert und ggf. um RS- und RK-Roboter erweitert werden müssen. Komplexe Handhabungsaufgaben lassen sich aus mechanischer Sicht leicht erfüllen, indem z. B. an eine Hubsäule weitere Roboter angebracht werden. Jedoch muss eine angepasste Robotersteuerung entwickelt werden, welche die Mechanik effizient nutzen kann.

Die Mechanik wird mit CAD-Programmen (Computer Aided Design) geplant und kann Massen und Trägheiten berechnen. Zur Antriebsauslegung werden die Werkzeuge der Motorhersteller verwendet. Neue Maschinen werden einmalig mit Berechnungen und Messungen optimiert und dann mit ihren Varianten (z. B. Greifer mit unterschiedlichen Massen) als Baukastensystem verwendet. Die Maschinenparameter der Varianten werden dann hauptsächlich empirisch z. B. auf Grundlage der Erfahrungen der KHS Mitarbeiter ermittelt, weniger durch Berechnung oder Optimierung.

Als Steuerungen kommen SPS von Siemens, Rockwell oder Kuka zum Einsatz. Sie befinden sich zusammen mit Motorreglern, BUS-Bausteinen, Bausteinen für Lichtschranken und sonstiger Peripherie in Schaltschränken. Die Motorregler bieten Einstellmöglichkeiten wie max. Drehzahl, Strom, Anlaufzeiten, Lastträgheit, Parameter der mechanischen Bremse, Abtastintervalle, Kalibrierung des Resolvers, Filter- und Reglerkoeffizienten zur Anpassung an den Motor und an die Mechanik. Mit einem externen Programmiergerät können die Parameter programmiert, Ist-Größen erfasst sowie dargestellt werden. Die PC-basierten Kuka-Systeme benötigen kein Program-

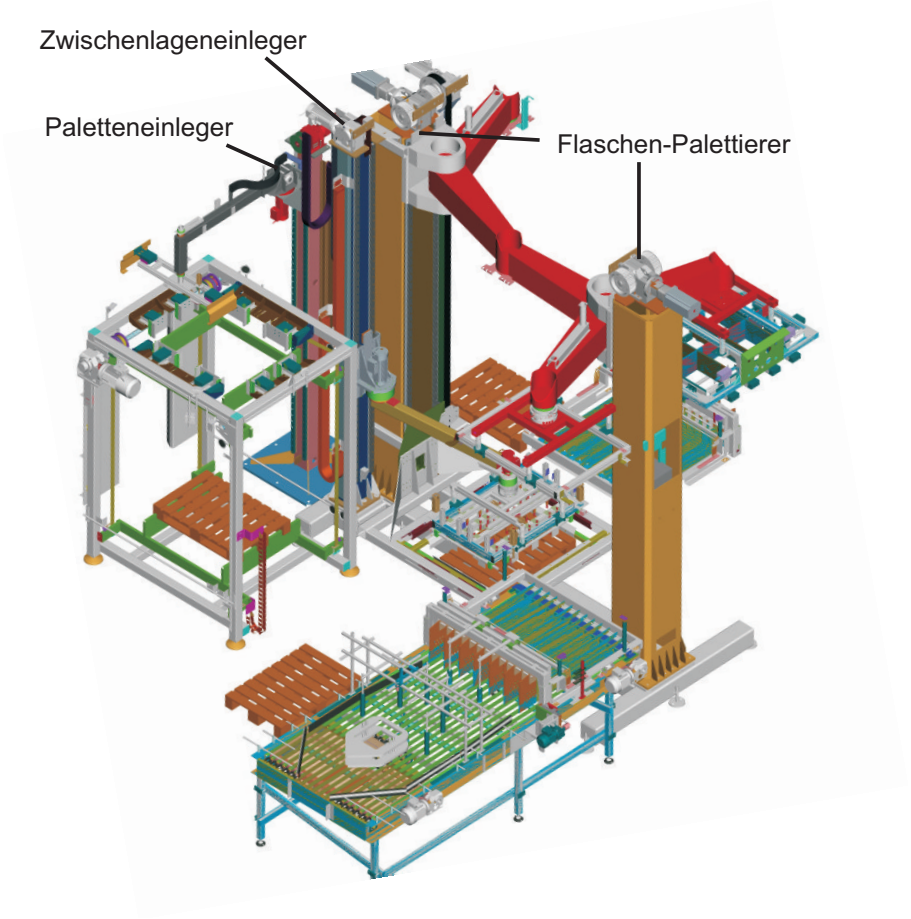


Abbildung 2.14: Modell einer Palettieranlage mit 4 Hubsäulenrobotern. Der Testroboter kommt als Zwischenlageneinleger zum Einsatz (von KHS)

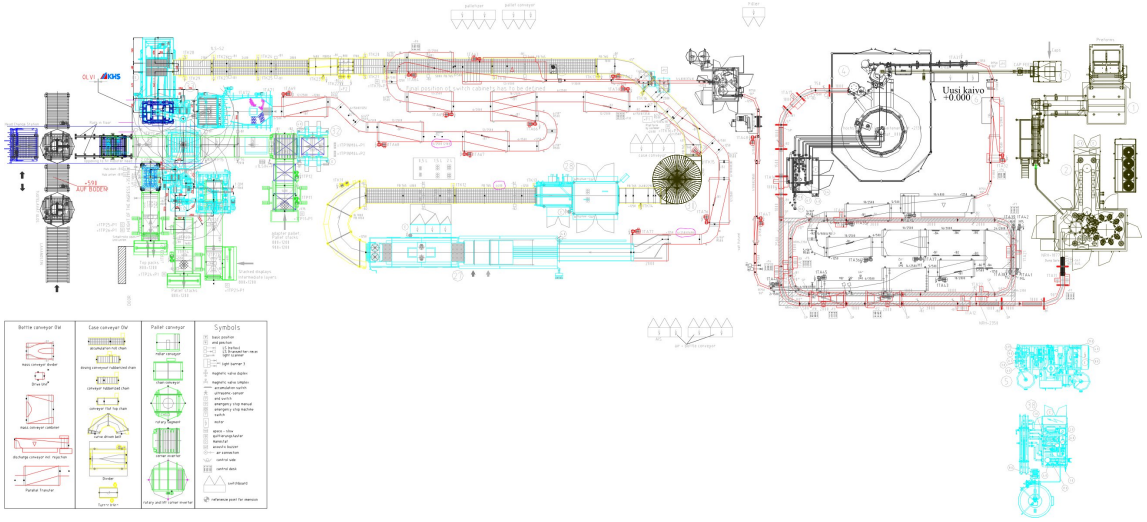


Abbildung 2.15: Einordnung des Palettiermoduls (hellblau, links) am Ende des Fertigungsprozesses (von KHS)

Abbildung 2.16: Robotermodul aus Abbildung 2.14 während der Produktion, nur 3 der 5 Säulenroboter sind in dieser Perspektive sichtbar, Betrieb der kompletten Anlage mit einer gemeinsamen Kuka-Steuerung

miergerät, da Einstellungen einfach über Dateien verwaltet und geändert werden können.

Gegenstand dieser Arbeit sind die von der KHS hergestellten Roboter für Palettierungsaufgaben. Als Testroboter kommt u. a. ein 3-achsiger Palettierer aus Abbildung 2.13 zum Einsatz. Der Roboter ist Teil der Palettieranlage aus Abbildung 2.14.

Das Palettiermodul mit dem Zwischenlagen-Einleger ist Teil der Multifunktionsanlage aus Abbildung 2.15. Die Anlage befüllt Kunstofftrays mit PET-Flaschen und palettiert die Trays. Abbildung 2.16 zeigt den Aufbau der Verpackungsmoduls während der Produktion. Die Herausforderung solcher Maschinensysteme liegt in der Erstellung der koordinierten Bewegung der vielen Achsen. Eine Kuka-Steuerung kann die komplette Anlage betreiben, jedoch können die Motoren nur getrennt im Einzelachsbetrieb (externe Servoachsen) angesteuert werden.

Um die zu erstellende Bahnplanung und -optimierung allgemein zu gestalten und ihre Leistungsfähigkeit zu demonstrieren, wird in dieser Arbeit die Modellierung eines SCARA-Roboters mit vier Achsen durchgeführt. Ein SCARA kann aufgrund seines zusätzlichen Ellenbogengelenks die Hand im kartesischen Raum frei bewegen und sich so z. B. Corioliskräfte zur zusätzlichen Beschleunigung nutzen. Die Modelle lassen sich leicht auf einen 3- oder 2-achsigen KHS-Säulenroboter reduzieren.

2.3 Anforderungen an Roboter aus Industriesicht

Die umfassende mathematische Beschreibung und Synthese von Industrierobotern stellt Ingenieure aus dem Maschinenbau, der Elektrotechnik, der Informatik und weiteren Disziplinen vor viele Herausforderungen: Bahnplanung, Dynamik, Lebensdauerberechnung, Schwingungsverhalten, Mehrziel-Optimierung etc. fließen in den Entwurf mit ein.

> *„Für die komplette Berechnung der Roboterdynamik eines Kuka-Roboters werden ca. 6 Mannjahre verwendet.“*,

nach Aussage eines Kuka-Mitarbeiters.

Im Gegensatz zu Kuka fertigt die KHS keine Serienroboter in hohen Stückzahlen, sondern Individual-Anlagen in kurzen Lieferzeiten. Eine aufwendige Dynamik-Optimierung jedes einzelnen Roboters ist somit nicht anwendbar.

Projekte der aktuellen Forschung gehen über die Industrierobotik hinaus und betrachten Systeme, die intelligent auf eine unbekannte Umwelt reagieren können. Solche Systeme müssen im Gegensatz zu Industrierobotern über entsprechende Sensorik zur Erfassung der Umwelt verfügen. Weitere Randbedingungen wie Autonomie, Mobilität und Autarkheit verschärfen die Anforderungen bei dem Entwurf der Robotersysteme. Gegenstand der aktuellen Forschung sind kooperierende, mobile Roboter, die im Team Aufgaben gemeinsam lösen. Kretschmann [Kretschmann 06a] skizziert ein solches Robotersystem.
Als Beispiel für ein Forschungprojekt wird RAVON (Robust Autonomous Vehicle for Off-road Navigation, [Berns 06]) der Arbeitsgruppe Robotersysteme, TU Kaiserslautern aus Abbildung 2.17 verwendet:

> *„Er dient als Versuchsplattform für die Erforschung verhaltensbasierter Bewegungs-, Lokalisations- und Navigationsstrategien in rauem, unebenem Gelände. Die Basiskonstruktion der Maschine wurde von Robosoft hergestellt.*
> *Die Welt von heute konfrontiert uns mit einer Häufung von Naturkatastrophen, Großunfällen und terroristischen Aktivitäten. Brandkatastrophen, Wirbelstürme, Flugzeugunglücke und Angriffe auf Chemiefabriken oder Atomkraftwerke stellen lediglich eine kleine Auswahl möglicher Szenarien dar. Autonome Fahrzeuge könnten an Grenzen patrouillieren, industrielle Anlagen schützen, routinemäßige Messungen in vordefinierten Bereichen durchführen, Aufklärungsaufgaben in lebensgefährlichen Umgebungen übernehmen oder bei Aufräumarbeiten nach schweren Unfällen oder Naturkatastrophen helfen.*
> *Um zukünftigen Anforderungen im Bereich Outdoor-Robotik gerecht zu werden, hat sich die AG Robotersysteme die Entwicklung eines vollautonomen Fahrzeuges für raues und bewachsenes Gelände als langfristiges Ziel gesetzt. Das reproduzierbare Abfahren von durch Wegpunkte abgesteckten Routen sowie die selbstständige Erkundung und Kartierung unbekannter Umgebungen sind als Schlüsselfähigkeiten definiert worden.“*

Abbildung 2.17: Mobile Versuchsplattform RAVON der AG Robotersysteme[1]

nach [Robotersysteme 07].

Tabelle 2.2 fasst die Randbedingungen einer KHS-Verpackungsanlage zusammen und stellt wesentliche Eigenschaften eines Forschungsprojekts denen eines kommerziellen Anlagenmoduls gegenüber. Aus ihr folgt, dass aus Zeit- und Haftungsgründen für die Industrierobotik hauptsächlich Komponenten zugekauft und nur wenige selbst entwickelt werden. Die Anlagen müssen Sicherheitsnormen einhalten und vorhandene Standards umsetzten, so dass ein sicherer und verlässlicher Betrieb auch durch weniger geschultes Personal gewährleistet werden kann.
Die Einführung selbst erstellter Technologien führt zu Kosten und Risiken, da Mitarbeiter und Kunden neu geschult werden müssen. Neue Technologien werden oft so gestaltet, dass sie die Funktionalität älterer Technologien weiterhin anbieten können, wie z. B. SPS-Emulation auf Kuka-Industrie-PCs mit Software-SPS. Die Herausforderung industrieller Steuerungen ist die Optimierung der Taktzeiten, obwohl die Kommunikation mit vielen Busteilnehmern notwendig ist. Zudem muss die Steuerung leicht verständlich und verlässlicht sein. Die eingesetzten Recheneinheiten besitzen meist eine höherere Leistung als benötigt, denn die Folgekosten einer nachträglich notwendigen Umrüstung und Anpassung, vor allem auf der Außenbaustelle beim Kunden, würde zu immensen Kosten führen. Dies begründet, warum für Industrierobotersteuerungen überdimensionierte Rechenleistung in großen Schaltschränken eingesetzt werden, obwohl ein mobiler Roboter wie RAVON weitaus rechenintensivere Funktionen wie Bildverarbeitung, Sensorauswertung und -fusion, intelligente Verhalten etc. mit kompakterer Hardware und effizienterer Software online umsetzen kann.

[1] Quelle agrosy.informatik.uni-kl.de

Kriterium	RAVON	Palettiermodul
Zahl Aktuatoren	6 (4x Motor, 2x Lenkung)	3 bis 30
Sensorik	Stereo-Kamera, Laser-Scanner, IR, GPS, Intertialsystem, taktil	Lichtschranke, Winkelencoder, Stromsensor, taktil
Freiheitsgrade der Systemgestaltung	beliebig	Beschränkung auf kommerziell verfügbare Technologien
Technologien	proprietär	KHS, Kuka, ISG, SEW, etc.
Rechner	Mini-PC, Laptop	SPS, Industrie-PC
Systemsoftware	Open-Source RTAI-LINUX	VxWorks, ProConOS, Windows
Software-Architektur	homogen: Open-Source MCA	heterogen: Kuka + kommerzielle Erweiterungen
Programmiersprache	C++	SPS-/KRL-Sprachen
IDE	Open-Source, beliebig	Kuka/ KW-Software MultiProg
BUS	CAN, WLAN	Profibus, DeviceNet, etc.
Energieversorgung	autark (10 W - 2 kW)	beliebig viel Leistung (bis 20 kW)
Herausforderungen	Bewegungs-Intelligenz, konsistente Umwelterfassung, Mobilität	Taktzahl, Verlässlichkeit, Verständlichkeit der Steuerung
Optimierung	konstruktiv, analytisch	empirisch
Verhalten	autonom	vorprogrammiert
Menschen mit Kontakt zu Maschine	wenig	sehr viele
Sicherheit, Einhaltung von Normen	unwichtig	wichtig
Haftung bei Fehlern	/	relevant
Umwelt	unbekannt, dynamisch	exakt definiert, dynamisch
Umweltkenntnis	unvollständig	exakt
Steuerungs-architektur	verhaltensbasiert	funktionsbasiert
Budget	gering	groß
Zeitrahmen	beliebig	3 Monate zwischen Auftrag und Auslieferung
Eigenentwicklung	Mechanik, Elektronik, SW	teilweise Mechanik + SW
Ziel	neue Erkenntnisse	finanzieller Gewinn

Tabelle 2.2: Gegenüberstellung 2er Produkte der Robotik: wissenschaftliches Fahrzeug RAVON und industrielles Palettiermodul der KHS

Obgleich in der industriellen Umgebung die Umwelt oberflächlich als exakt bekannt gilt, ist eine genaue Modellierung und Simulation der Anlage aufgrund des Zeitdrucks nicht sinnvoll. Ein einzelner Roboter kann zwar simuliert und optimiert werden, er zeigt jedoch seine Funktionalität nur im Zusammenspiel mit den übrigen Anlagenkomponenten und dem zu verarbeitenden Material. Bei einem Unterdruck-Palettierer für eingeschweißte Gebinde hängt beispielsweise die Armgeschwindigkeit von der Roboterkinematik, -dynamik, Taktzeit der Anlage, Unterdruck der Sauggreifer, Größe der Öffnung der Sauger, Masse der Gebinde, geometrische Form eines Gebindes, Lagenbild der Gebinde mit wechselseitigen Kräften, Materialbeschaffenheit der Plastikfolie, Temperatur, Feuchtigkeit, etc. ab. Die für eine konstuktive Optimierung der Armgeschwindigkeit notwendige exakte Modellierung müsste somit die komplette Anlage und ihre Umgebung erfassen. Das ist in einem engen Zeitrahmen schwer möglich.

Eine industrielle Steuerung ist zwar nicht durch eine aufwändige Sensorverarbeitung und die Erzeugung eines intelligenten Verhaltens wie bei dem vorgestellten mobilen Roboter ausgelastet, aber durch die Einhaltung von Sicherheitsaspekten und die hohe Zahl der Busteilnehmer und der zu bewegenden Achsen. Wenn die Bahnplanung mehrere Roboter in einer gemeinsamen Steuerung bewegen soll, sollte sie möglichst geringe Anforderungen an Rechenleistung und Speicher stellen. Sie muss genügend Stellschrauben für eine empirische Optimierung anbieten. Eine weitere Randbedingung ist die problemlose Integration in die vorhandene Steuerungsarchitektur, die im nächsten Abschnitt beschrieben wird.

2.4 Architektur der Robotersteuerung

Die Steuerungsarchitektur beschreibt das funktionale Zusammenwirken der Hard- und Softwarekomponenten der Robotersteuerung. Die Wahl der Steuerung setzt bereits die ersten Randbedingungen für die Gestaltung der Software fest. Im Gegensatz zu einfachen Steuerungsaufgaben wird eine Bahnplanung hohe Anforderungen an Speichergröße, Rechenleistung und an Ausdrucksfähigkeit der Implementierungssprache stellen. Als Alternativen stehen in der KHS die Steuerungssysteme von Siemens, Rockwell und Kuka zur Auswahl. Das PC-basierte System von Kuka ist das leistungsfähigste (ca. 5 μs/1000 Anweisungen in SPS-Sprache AWL) und besitzt umfassende Programmier-Bibliotheken. Es wird deshalb ausgewählt und im Folgenden näher beschrieben.

2.4.1 Hardware des Kuka-Schaltschranks

Für die Hardware wird seit 8 Jahren eine vereinheitlichte Lösung der Fa. Kuka aus Abbildung 2.18 eingesetzt. Der projektspezifische Projektierungs- und Konstruktionsaufwand mit unterschiedlichen Komponenten von Kuka, Siemens, Rockwell, Mitshubishi, Danfoss, etc. war zuvor schwierig zu beherrschen und stellte den weltweiten Kundenservice vor große Herausforderungen im Know-How-Transfer an Servicetechniker und Kunden.

Der Kuka Schaltschrank KR C2 wie in Abbildung 2.19 ist das Steuerungszentrum

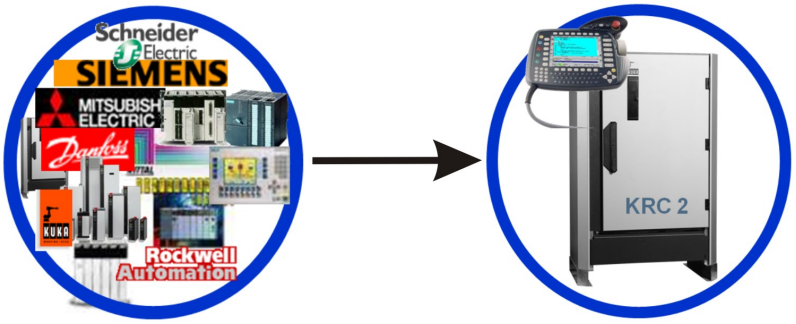


Abbildung 2.18: Die Vielfalt der Steuerungskomponenten wird in der KHS je nach Anforderungen der Anwendung durch die einheitliche Plattform der Kuka-Systeme ersetzt, aus [Kuka 07b] Rau S. 16-17

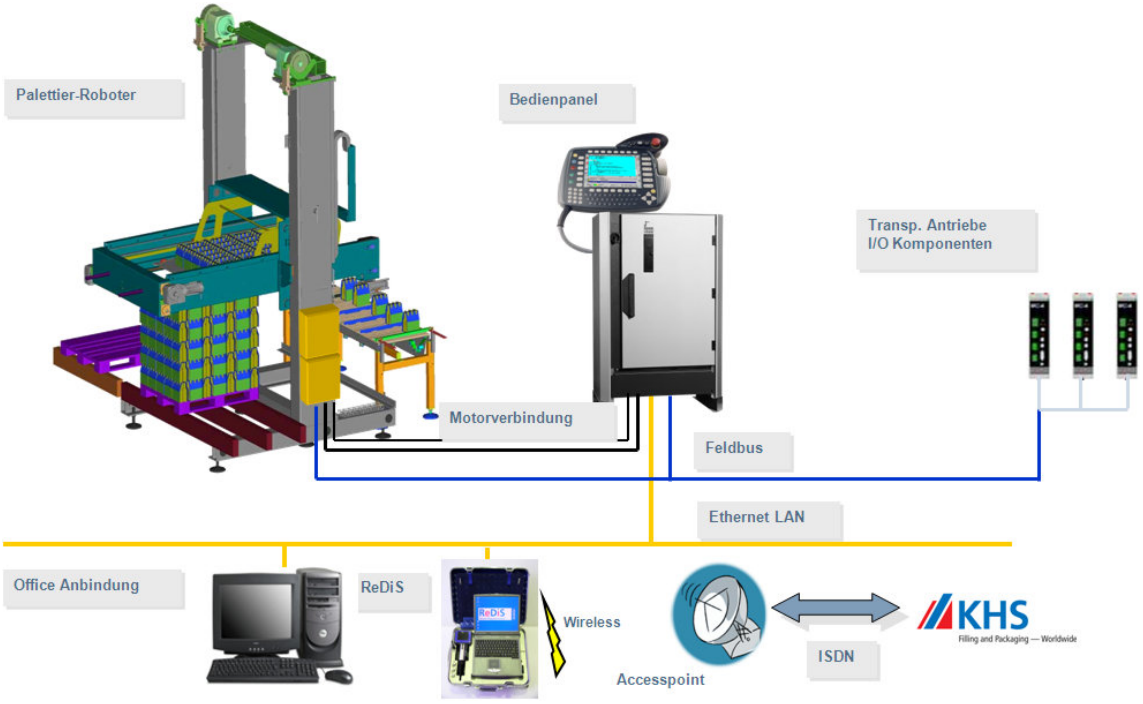


Abbildung 2.19: Konnektivität der Kuka-Steuerung: Maschinenunabhängige Installation durch konfektionierte Motorverbindungen, Einsatz von verschiedenen Feldbussystemen (Profibus, Devicenet, ...), Homogene Kommunikation mit Ethernet (Office, HMI, ReDiS, Motion-Control), aus [KHS 07c] Verpackungstechnik Worms S. 17

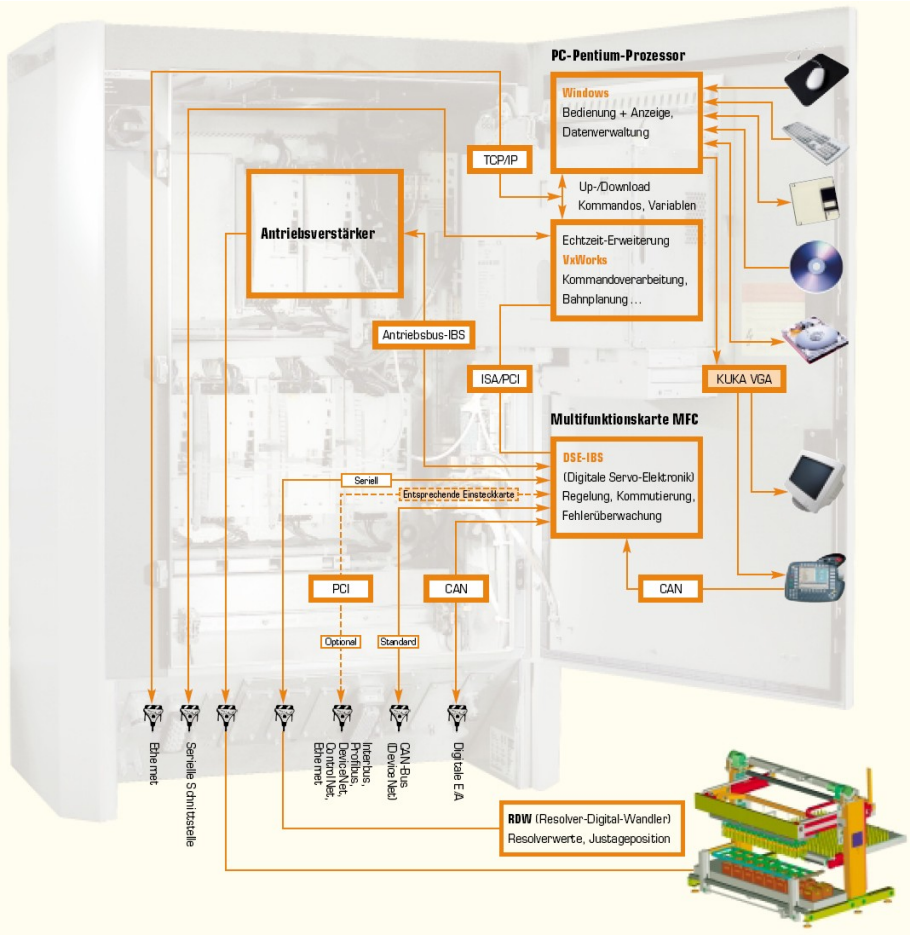


Abbildung 2.20: Schaltschrank mit Hardware der Steuerung: Einschübe für 8 Servoregler (oben), konfigurierbares Steckerfeld (unten), Industrie-PC (oben in Tür), anwenderspezifische Einbauten (unten in Tür), aus [Kuka 07a] KMC - KUKA Motion Control S. 10

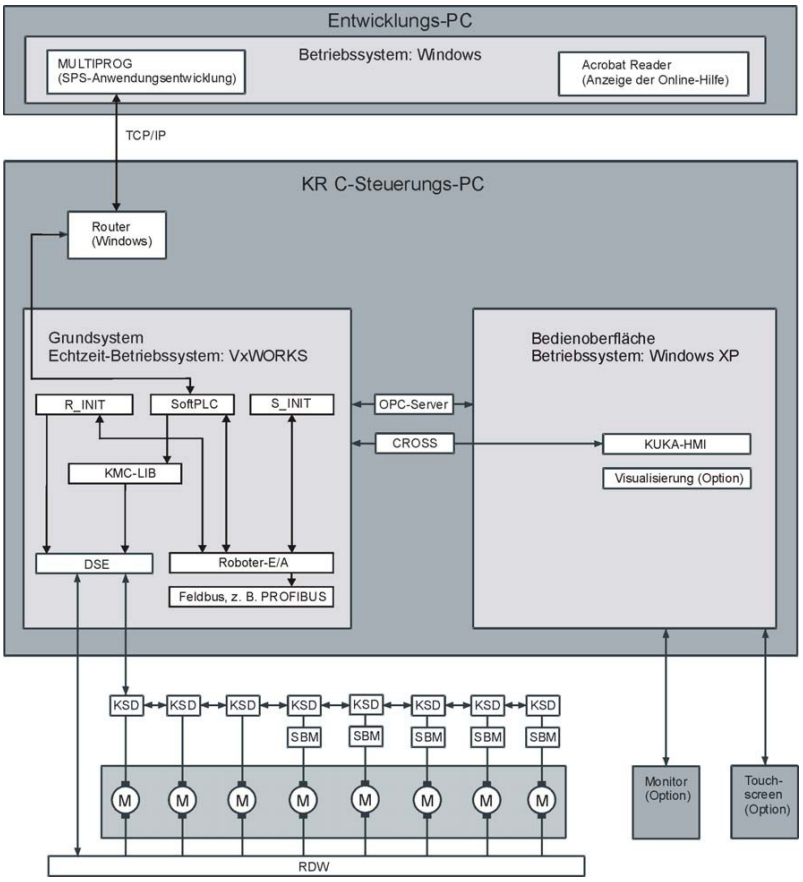


Abbildung 2.21: Softwarearchitektur der Steuerung: Windows XP zur Visualisierung und Eingabe, VxWorks als Echtzeitbetriebssystem für die Robotersteuerung, aus [Kuka 07a] KUKA Motion Control (KMC) S.16

eines Anlagenabschnitts. Peripheriegeräte lassen sich über die gängigen BUS- und Netzwerksysteme integrieren. Abbildung 2.20 zeigt Einschübe für die Leistungselektronik (Motorregler), Kommunikationskarten, einen Industrie-PC und kundenspezifische Erweiterungen. Die im Schrank eingebauten Kuka-Motorregler werden über Dateien parametriert und an die Motoren angepasst. Die Programmierung der Regler mit einem externen Gerät wie bei Reglern der Marke SEW oder Danfoss, die vor allem in Kombination mit S7-Steuerungen eingesetzt werden, entfällt. Sollen mehr Achsen gesteuert werden als Kuka-Regler (max. 16) in einem Schaltschrank vorhanden sind, kann der PC um Steckkarten für Bussysteme wie CAN, Profinet, ASI, etc. erweitert werden und z. B. Danfoss-Regler über den Bus ansteuern.
Als Kommunikationsschnittstelle für den Bediener (HMI) kommt ein grafisches Handbediengerät (KCP) zum Einsatz. Als PC-System bietet es Ethernet, VGA, USB und weitere Schnittstellen, die Test und Betrieb erleichtern und die Kommunikation mit Datenverwaltungssystemen ermöglichen.

2.4.2 Softwarekomponenten des Industrie-PCs

Abbildung 2.21 beschreibt die Softwarearchitektur des Systems. Auf dem Industrie-PC mit bis zu 2 GHz und 1 GB RAM laufen 2 Betriebssysteme: Das Echtzeitbe-

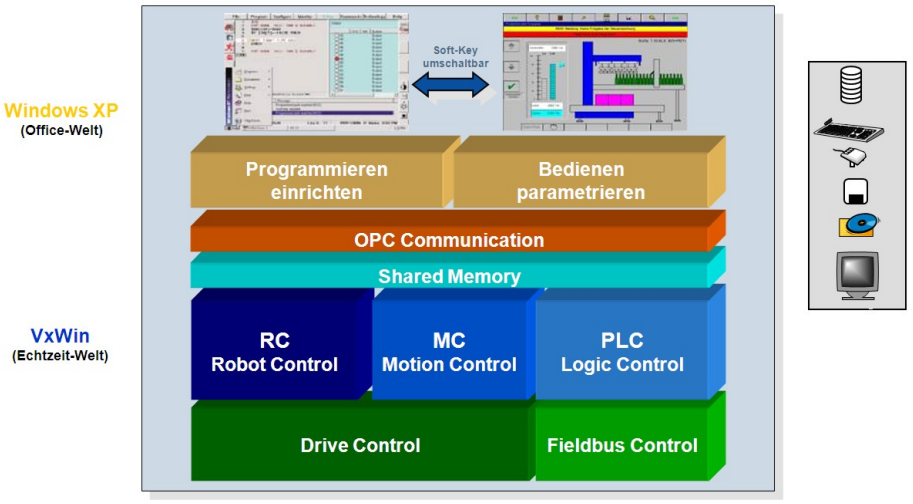


Abbildung 2.22: Windows XP läuft auf auf VxWorks, VxWorks beinhaltet RC, MC und SPS, aus [Kuka 07b] Rau S. 21

triebssystem VxWorks und Windows XP. Windows XP dient zur Visualisierung und Kommunikation mit dem Anwender. In VxWorks läuft Windows als Prozess. Somit kann Windows Ressourcen entzogen werden, falls die Robotersteuerung dies erfordert. Außerdem laufen in VxWorks eine SPS-Emulation (SoftPLC), die Bibliotheken der Kuka-Steuerung (KMC-Lib), die Verwaltung der digitalen Servoelektronik (DSE) und E/A- und BUS-Funktionen. Windows XP und VxWorks sind aus Sicherheitsgründen abgeschottet und benötigen den OPC-Server (OLE for Process Control) zur Kommunikation. OPC ist nicht echtzeitfähig und kann deshalb nicht zur Steuerung sondern nur zur Prozessdatenerfassung verwendet werden. Mit Windows XP stehen alle Vorteile eines normal PCs dem Industrie-PC zur Verfügung, wie z. B. der Anschluss weiterer Monitore und Eingabegeräte und auch die Ausführung gängiger Standardsoftware wie Microsoft Office, vor allem Excel und Access zur Verwaltung von Fehlermeldungen und Betriebszustände.

Die SPS-Programme werden über die Netzwerkverbindung vom Entwicklungs-PC auf den Steuerungs-PC übertragen. Außerhalb des PCs befinden sich im Roboter die Servoantriebe (Kuka Servo Drive KSD), die Bremsmodule (Single Brake Module SBM) und die Antriebssensorik (Resover Digital Module RDM).

Abbildung 2.22 liefert eine andere Sicht auf die VxWorks-Steuerung. In VxWin befinden sich die Funktionalitäten zur Steuerung eines Kuka-Roboters mit **Kuka Robot Control** (KRC). KRC kann nur einen Roboter als synchrones System ansteuern. Dieser kann Kuka-Knickarm-Roboter sein oder einer der vordefinierten Typen wie SCARA, Portal-Roboter, etc. . KRC beinhaltet eine kartesische Bahnplanung und die Programmiersprache KRL (Kuka Robot Language). Mit ihr können im Teach-In-Verfahren Stützpunkte angefahren und eine Bahn generiert werden. Punkte auf dieser Bahn werden Programmen zugeordnet, so dass über den Aufruf von Programmen bestimmte Bewegungen ausgeführt werden.

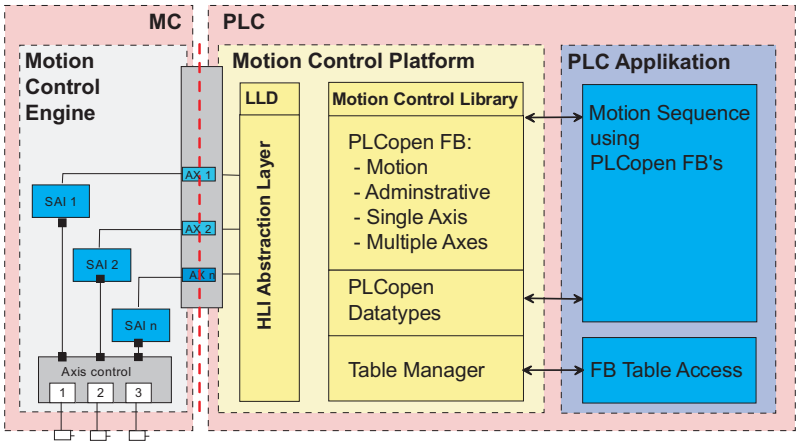


Abbildung 2.23: Schnittstelle der Motion Control Plattform als SPS-Biblothek, angelehnt an [ISG 01] S. 8

Eine weitere Komponente ist **Kuka Motion Control** (KMC). KMC dient zur Einzelachssteuerung wie sie für Werkzeuge, Bohrer, Greifer, Geländerverschiebungen und sonstige Aktuatoren benötigt wird. Im Gegensatz zu KRC ist in KMC keine Kinematik modelliert und alle Motorachsen werden getrennt angesteuert. KMC stammt aus dem CNC-Bereich und bietet keine Bahnplanung sondern Bibliotheken für die Einzelachssteuerung an. Zudem können mehrere Achsen in einem Achsverband miteinander gekoppelt Bewegungen ausführen. Die Kopplung kann durch virtuelle Getriebe oder Kurvenscheiben erfolgen.

Das Anwenderprogramm der KMC-Steuerung befindet sich in **Programmable Logic Device** (PLC, speicherprogrammierbare Steuerung, SPS). Dort emuliert das Betriebssystem ProConOS eine SPS-Umgebung, so dass SPS-Programme auf einem Industrie-PC ausgeführt werden können.

2.4.3 Einbettung der SPS in die Steuerung

Von der SPS-Umgebung kann über eine Speicherschnittstelle (High Level Interface, HLI) auf KMC zugegriffen werden. Dazu kommt die Motion Control Plattform aus Abbildung 2.23 zum Einsatz. KMC wird somit in einer SPS-Bibliothek für den Anwendungsprogrammierer gekapselt. Diese Bibliothek stellt Funktionsbausteine zum Ansteuern der Motoren als Einzelachsen oder als Achsverband zur Verfügung.

Da auf dem Markt viele Lösungen für Motion Control mit SPS existieren, versucht das PLCOpen-Kommitee [PLCOpen 01] diese Schnittstelle für SPS-Programmiersprachen zu standardisieren, um die Wiederverwendbarkeit und Kompatibilität zwischen SPS-Programmen zu erhöhen. Die in der SPS-Bibliothek enthaltenen Funktionsbausteine implementieren diese Standard-Schnittstelle.

Die Funktionsbausteinsprache ist eine in IEC 61131-3 definierte SPS-Sprache und

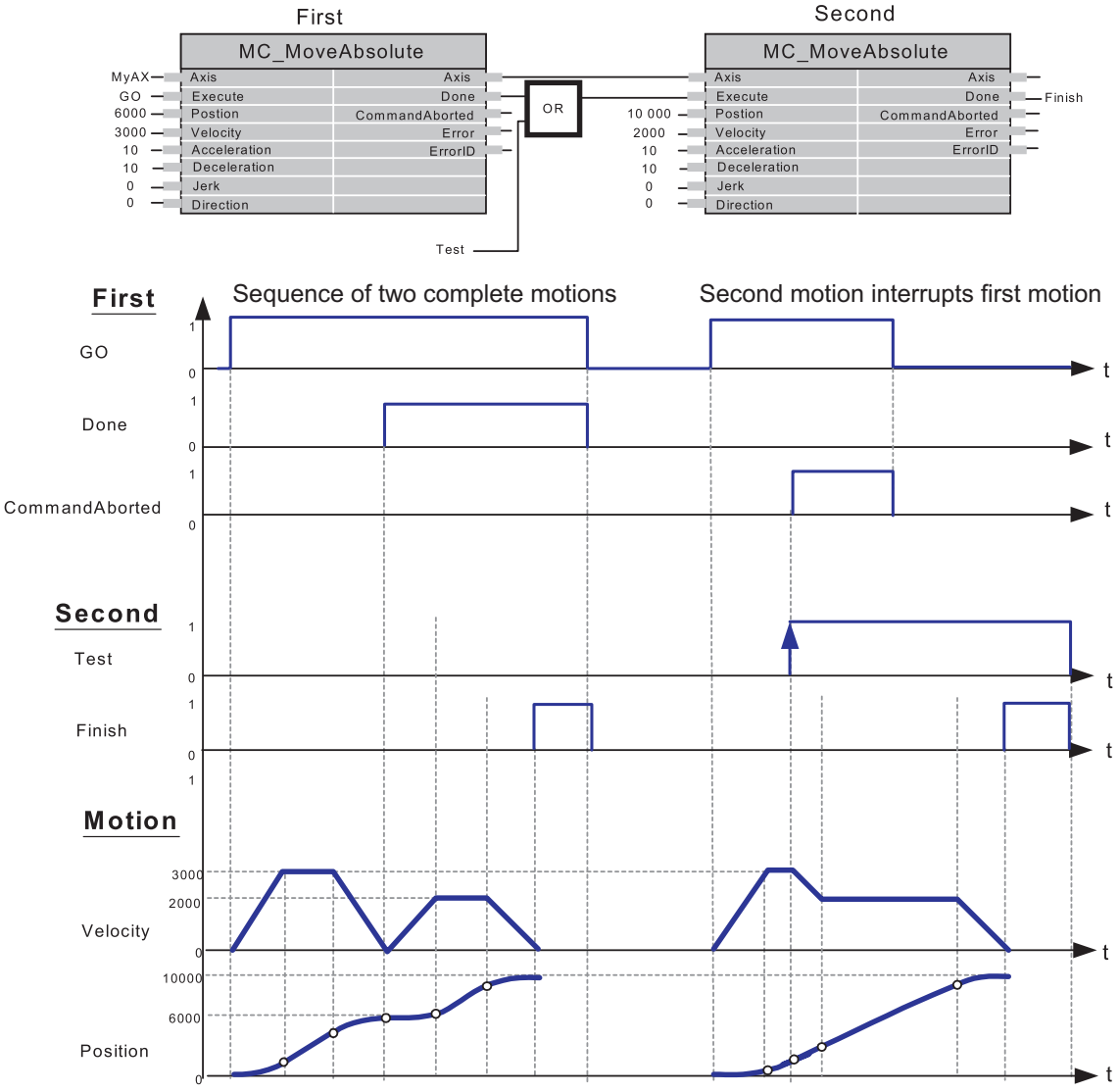


Abbildung 2.24: Zeitdiagramme des MoveAbsolute-Funktionsbaustein für die Einzelachssteuerung, aus [PLCOpen 01] S. 20

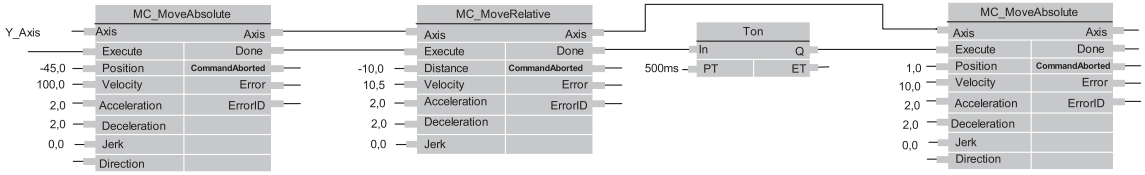


Abbildung 2.25: Programm einer Einzelachs-Anwendung mit verknüpften Funktionsblöcken für eine Bohraufgabe, aus [PLCOpen 01] S. 54

weit verbreitet in der Automatisierungstechnik. Das Verhalten aller PLCOpen--Funktionsbausteine wird durch ein gemeinsames Zustandsdiagramm definiert. Es wird zwischen administrativen und Motion-Bausteinen unterschieden. Administrative Bausteine beinhalten Funktionen zum Aktivieren der Motoren (setzen von Motorparametern, etc.). Motion-Bausteine führen konkrete Fahrbefehle für einzelne Achsen aus. Achsen können über ein virtuelles Getriebe miteinander gekoppelt werden, um komplexe Bewegungen zu realisieren. Ebenso ist die kombinierte Fahrt aller Achsen auf Basis einer CAM-Tabelle möglich.

Abbildung 2.24 zeigt die Bewegung einer Achse mit dem Funktionsbaustein MoveAbsolute. Eine Flanke am *Execute*-Eingang startet die Fahrt. Die Achse wird auf die angegebene Position mit definiertem Maximaleinstellungen für Ruck, Beschleunigung und Geschwindigkeit gefahren. Ist das Ziel erreicht, teilt dies der Baustein mit dem Ausgang *Done* mit.

Bausteine können synchron ausgeführt werden. Regeln für die resultierende Bewegung der Motorachse sind in [PLCOpen 01] näher spezifiziert.

Die Verknüpfung von Funktionsbausteinen führt zu einem kompletten Steuerungsprogramm. Für die Bohraufgabe aus Abbildung 2.25 nähert sich der Bohrer schnell dem Werkstück, danach bohrt er langsam das Loch, wartet in dem Loch um Bohrspäne zu entfernen und fährt wieder langsam zurück.

Die Steuerungsarchitektur ist von einer heterogenen Softwareinfrastruktur geprägt. Zum Einsatz kommen die Betriebssysteme VxWorks, Windows XP und ProConOS, die Steuerungsmodule KRC, KMC und PLC und weitere Softwareschnittstellen von Kuka und weiteren Herstellern. Außerdem befinden sich Komponenten der Entwicklungsumgebung, BUS-Treiber und Standardsoftware auf dem Industrie-PC. Zudem soll auf der Soft-SPS die bisherige Anlagensteuerung weiterhin laufen. ProConOS bietet viele Stellschrauben zur Beeinflussung einzelner Systemkomponenten an, zum Beispiel beim Scheduling und Speichermanagement.
Deswegen ist eine konstruktive Berechnung der Laufzeitkomplexität des kompletten Bahnplanungsalgorithmus' unter Einbeziehung aller Randbedingungen der Steuerungsarchitektur mit vertretbarem Aufwand nicht möglich. Stattdessen muss die Bahnplanung vorerst für stark begrenzte Rechen- und Speicherressourcen ausgelegt sein. Falls empirsche Analysen nach der Implementierung nicht benötige Ressourcen liefern, wird die Bahnplanung zu Lasten des Ressourcenverbrauchs optimiert.

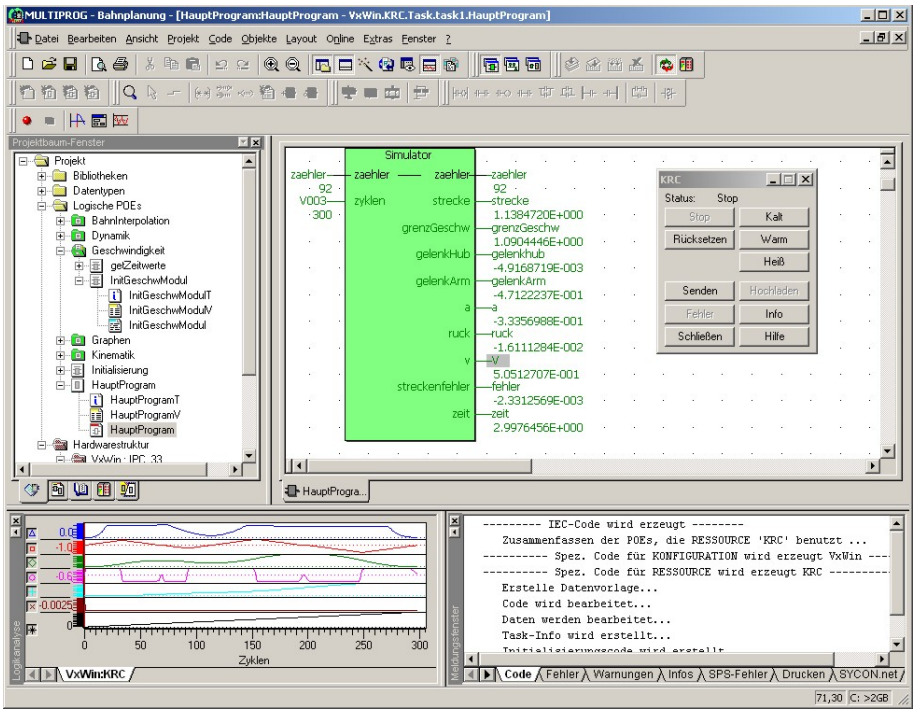


Abbildung 2.26: Die SPS-Entwicklungsumgebung MultiProg

2.5 Entwicklungsprozesse und -werkzeuge

Die Entwicklungswerkzeuge mit ihren unterstützen Programmiersprachen und Bibliotheken stellen die Randbedingungen für die Lösung eines Problems und nehmen teilweise Entwurfsentscheidungen vorweg. Als Entwicklungsumgebung der SPS-Programme kommt MultiProg der Fa. KW-Software zum Einsatz. Abbildung 2.26 zeigt die IDE mit einem SPS-Simulator im Debug-Modus. MultiProg kann zwar die SPS-Umgebung simulieren und den online-Wert von Variablen anzeigen, jedoch bietet sie keine Simulation der Roboterkinematik oder gar eine 3D-Visualisierung.

MultiProg unterstützt die IEC 61131-3 Programmiersprachen Anweisungsliste (AWL), Strukturierten Text (ST), Funktionsbausteinsprache (FBS), Kontaktplan (KOP) und Ablaufsprache (AS).
Nach Lepers [Lepers 05] ist **AWL** eine stackorientierte Sprache und ähnelt Assemblersprachen. Sie kann für einfache Programme zur Verknüpfung von Ein- und Ausgängen verwendet werden und erzeugt einen kleinen Code. Bei komplexeren Problemstellungen führt die Anwendung dieser Sprache aber zu Unübersichtlichkeit.
ST ist mit seiner Pascal-ähnlichen Syntax an Hochsprachen angelehnt und bietet Konstrukte für Schleifen, Verzweigungen und die Definition von Datentypen an. Der Objektcode von ST benötigt jedoch mehr Speicher als AWL-Programme.
FBS ist eine grafische Sprache. Die syntaktischen Elemente sind Funktionen, Verbindungen und Variablen. Sie gilt als Anfängersprache, da der Kontrollfluss leicht durch die grafische Darstellung verständlich ist.
KOP stammt aus dem Bereich der Verknüpfungssteuerungen und ist an Stromlaufpläne der Elektrotechnik angelehnt.
AS modelliert Petri-Netze.

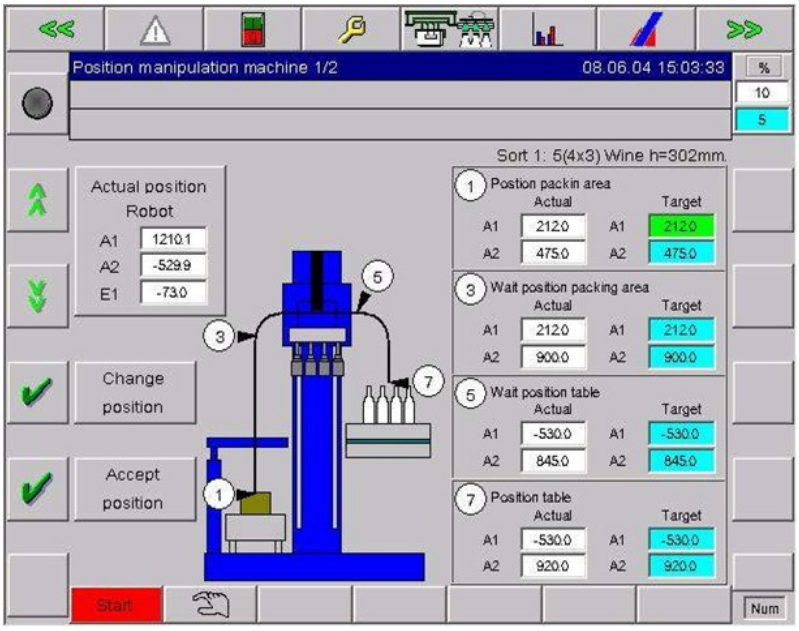


Abbildung 2.27: Anzeige der KHS-HMI auf dem KCP zur Parametrierung der Roboterbewegung, aus [KHS 07c] HMI S. 11

Robotersimulationen werden zur Unterstützung der Softwareentwicklung nur sporadisch eingesetzt. Aufgrund der kurzen Entwicklungszyklen ist die konsistente Verwaltung von Simulation und Anlage zu aufwendig. Viele Parameter können erst empirisch an der Maschine im Zusammenspiel mit anderen Anlagenteilen und dem Testmaterial des Kunden ermittelt werden. Das mögliche Kollisionspotential von zusammen arbeitenden Aggregaten muss durch das Anwenderprogramm beachtet und beseitigt werden. Kollisionen treten bei der Inbebetriebnahme der Anlage selten und während des Produktionsbetriebs beim Kunden nicht auf.

Die HMI der KHS-Maschinen wie in Abbildung 2.27 dargestellt ermöglicht dem Bediener die Kontrolle der Roboterbewegung und auch die manuelle Parametrierung der Bahn beim Wechsel des Produktionsmaterials mit z. B. veränderter Palettenhöhe. Diese Bahnplanung mit der Vorgabe spezieller Positionen der Hand ist zwar einfach zu verwenden, jedoch ist eine komplette Änderung der Bahn nur mit Programmierung und erheblichen Aufwand möglich und nicht durch einen Bediener sondern dem Softwareentwickler umzusetzen. Der Aufwand zur Erstellung einer umfassenden, komplexen Benutzerschnittstelle, die auch die Erstellung komplett neuer Bahnen gestattet, steht in der KHS nicht in Relation zu dem Nutzen. Außerdem fehlt die Bahnplanung im Hintergrund, welche die Bahn auf Plausibilität prüft und sie optimiert, so dass auch ein Bediener ohne Programmieraufwand gute Bahnen erstellen kann. Diese Problemstellung trägt ebenfalls zur Motivation dieser Arbeit bei.

Die HMI wird mit dem Visualisierungs-Werkzeug Iconics und einer an Visual Basic angelehnten Implementierungssprache erstellt und läuft auf Windows XP. Die An-

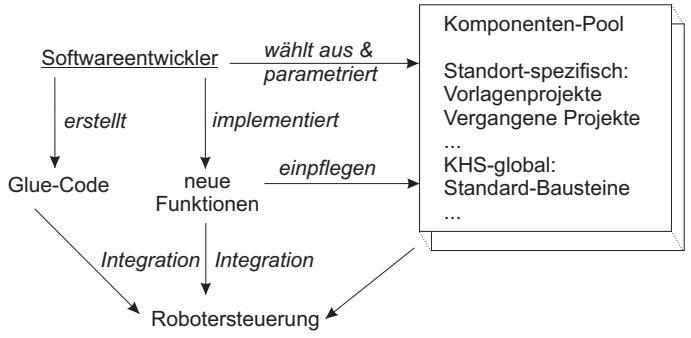


Abbildung 2.28: Vorgehen mit komponentenbasierter Entwicklung der Steuerungssoftware

zeige wird auf dem KCP oder einem gesonderten Display darstellt.

Die Philosophie der Softwareerstellung der KHS lautet *„Parametrieren statt Programmieren“*. Abbildung 2.28 stellt den Entwicklungsprozess dar. In allen Entwicklungsphasen wird nach wiederverwendbarer Software im Komponenten-Pool gesucht und lediglich nicht vorhandene Funktionen neu implementiert. Der Pool ist breit gefächert und enthält auch Varianten gleicher Maschinen. Dadurch können lauffähige und getestete Komponenten, in denen auch Erfahrungswerte liegen, schnell für ein neues Projekt wiederverwendet werden. Die Programme müssen so gestaltet sein, dass Monteure und Servicetechniker intuitiv Änderungen an der Steuerung ohne Kenntnisse der Programme und Änderung des Quellcodes durchführen können. Kernfunktionen, die selten verändert werden müssen, sind in ST implementiert, während anwendernahe Programme und die Parametrierung der Maschinen und Mechanik in FBS geschrieben werden.

Die Steuerung der Hauptmaschinen und Säulenroboter über KRC und KMC kann je nach Anzahl der Achsen über einen gemeinsamen Schaltschrank geschehen oder auf mehrere Schaltschränke verteilt sein. In der verteilten Lösung werden unter anderem Lichtschranken zur Synchronisation verwendet.

MultiProg bietet die Möglichkeit zur Integration externer C-Programmfragmente als SPS-Bibliotheken. Dies ist allerdings erst ab einer neueren Version als die in der KHS verwendeten möglich und somit noch nicht für Kuka-Plattformen verfügbar. Die Bahnplanung kann in SPS-Sprachen geschrieben und anschließend als Bibliothek zur Verfügung gestellt werden. Eingriffe in die Entwicklungsumgebung sind nicht unbedingt erforderlich.

Zur Überprüfung des Roboterverhaltens können sowohl über die Bedienoberfläche des KRC2 als auch im Debug-Modus von MultiProg Soll- und Ist-Größen wie Strom, Gelenkstellung, Geschwindigkeit, Schleppfehler, etc. ausgelesen und für die anschließende Auswertung aufgezeichnet werden. Für die Verbindung externer Sensoren wie Inertialsensor oder Drehmomentmessdosen mit der SPS liegen keine Erfahrungen vor. Hier könnten autarke Systeme eingesetzt werden, falls die Antriebssensorik nicht

Abbildung 2.29: Werkzeug für die automatisierte Lagenbildung (links), Umsetzung der Lagenbilder durch eine Robotergruppierung mit Knickarmroboter (mitte) und gemischtes Lagenbild auf der Palette (rechts), aus [Kuka 07b] Rau S. 34, 45, 46

genügend Informationen liefert.

2.5.1 Online-Bahnberechnung statt Teach-In

Die Bahn des Roboters ergibt sich aus seiner Aufgabe. In der Großserienproduktion der Automobilindustrie führen die Maschinen stets die gleichen Bewegungen für Schweißen, Lackieren und Transportieren aus. Hier wird das Teach-In-Verfahren angewendet: Die Bahn wird vom Bediener abgefahren und Stützpunkte gespeichert. Die Robotersteuerung kann aus den Punkten die Bahn rekonstruieren. Eine Änderung der Bahn kann nur manuell mit hohem Aufwand durchgeführt werden.

Verpackungsmodule der KHS sind multifunktional gestaltet: Je nach Kundenwunsch muss das Verpackungsmodul verschiedene Palettentypen (Euro, Halb-Euro, Industrie, Dolly, ..), automatischen Kopfwechsel (bis zu 50 Sensoren extra), automatische Geländerverstellung, Wahl der Formation (Anordnung der Gebinde in einer Palettenlage), Wahl des Flaschentyps (2l, 1,5l, 1l, 0,5l, Dosen, PET, Glas, etc.), Mischung von Produktsorten auf einer Palette ohne eine mechanische Nachjustierung unterstützen. Die Zahl der Bahnen wächst exponentiell mit den Anwendungsfällen. Da ein Teach-In aller Achsen nicht anwendbar ist, werden die Endpositionen der Achsen berechnet. Sie ergeben sich aus den Handhabungsaufgaben und den Eigenschaften des aktuell zu verarbeitenden Materials.
Abbildung 2.29 zeigt ein Werkzeug zur automatisierten Lagenbildung auf dem PC. Die erstellten Lagenbilder werden in die SPS geladen und können über die HMI angewählt werden. Ein oder mehrere Knickarmroboter greifen das Material und positionieren es auf dem Zulauf des Beladers.

Um für Maschinen mit der Einzelachssteuerung glatte Bahnen zu erzeugen, werden wie in Abbildung 2.30 die Ecken verschliffen: Vor der Endstellung des Hubantriebs wird ein Überschleiffenster definiert. Sobald die Auslenkung des Hubgelenks in das Fenster gelangt, wird der Armmotor in Bewegung gesetzt. Die Überlagerung der beiden Einzelbewegungen führt zu einer abgerundeten Bahnecke.

Für alle unterschiedlichen KHS-Maschinen existieren Programme, welche die Ausführung von Standardbewegungen implementieren. Diese müssen lediglich entspre-

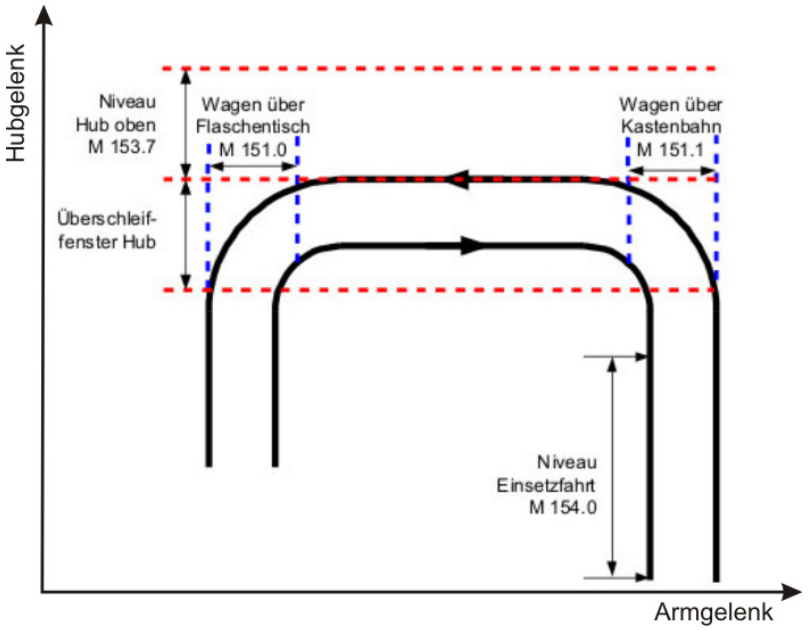


Abbildung 2.30: Überschleifen der Ecken in Einzelachssteuerung eines Einpackers zur Erhöhung der Bahngeschwindigkeit, angelehnt an [KHS 07b] Programmstruktur PPZ / SP S. 9

chend der Anforderungen eines neuen Auftrags parametriert werden und führen schnell und einfach zu einer lauffähigen Maschinensoftware. Die so entstandenen Bewegungen sind jedoch nicht optimal auf die Maschine und ihre spezielle Aufgabe abgestimmt. Hier entsteht die Forderung nach einer optimierenden und einfach wiederverwendbaren Bahnplanung.

2.5.2 Synchronisierung in Multi-Robotersystemen

Der Bewegungsablauf eines Aggregats oder Roboters wird mit einem Zustandsmodell (Schrittkette) ausgedrückt. Der Zustandswechsel erfolgt nach Erfüllung bestimmter Bedingungen und führt zu neuen Aktionen wie in Abbildung 2.31 gezeigt. Jede Maschine befindet sich in einer eigenen Schrittkette. Die Synchronisation der Schrittketten geschieht über eine gegenseitige Verriegelung durch Bedingungen zum Wechsel in den Folgezustand.

2.6 Probleme und Verbesserungspotential

Die Robotersteuerung Kuka-KRC2 bietet eine Bahnplanung an, jedoch unterstützt sie nur einen Roboter und ist nur für Kuka-Roboter optimiert. In KHS-Palettieranlagen kommt eine Vielzahl von Antrieben zum Einsatz, die über KMC und Einzelachssteuerung betrieben werden, wie z. B. in Geländerverstellungen, Laufbändern und Greifern. Die Erweiterungsmöglichkeiten des KRC2 gestatten es, über einen Bus neue Motorregler für die Einzelachssteuerung weiterer Antriebe mit nur einem gemeinsamen Kuka-Schaltschank KRC2 anzusteuern. Dies erlaubt den Aufbau von einfachen und günstigen Steuerungen.

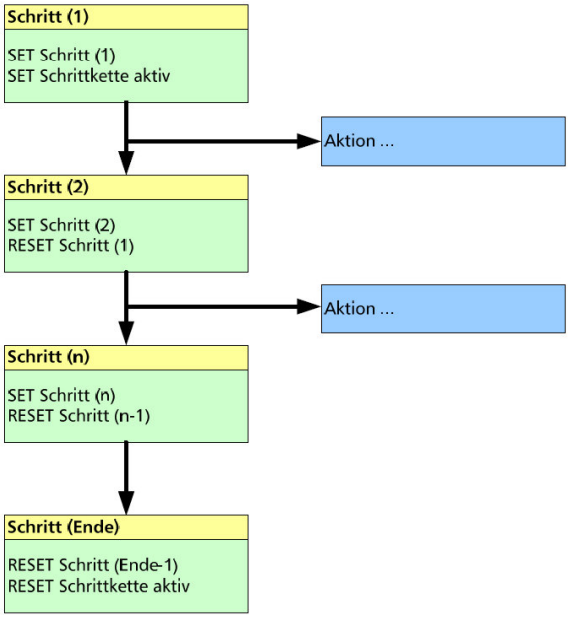


Abbildung 2.31: Schrittkette mit Abfolge von Aktionen für eine Maschine, aus [KHS 07b] Allgemeine Programmstruktur S. 6

Abbildung 2.32: Beladermodul bestehend aus Zentrierer und Zwischenlangen-Einleger als externe Servoachsen an der Hubsäule (links) und Kuka-Knickarmroboter (rechts). Betrieb über eine gemeinsame Kuka-Steuerung, aus [Kuka 07b] Rau S. 9

In der KHS werden über KMC auch verkettete Kinematiken wie die vorgestellten 3-achsigen Hubsäulenroboter RS3, Einleger, Portalroboter und Schieber angesteuert. Die Schwierigkeiten der Entwicklung und des Betriebs von Anwendungen aus Abbildung 2.32 sind Bahnplanung und die Kollisionsvermeidung, die zu Fuß durchgeführt werden müssen. Die aus der Einzelachssteuerung entstandenen Trajektorien sind nicht optimal, fehleranfällig und eine materialschonende, ruckbegrenzte Fahrt ist schwer zu realisieren. Eine Optimierung der Bahnen zum Erreichen der gewünschten Taktrate der Anlage geschieht ausschließlich empirisch und ist somit langwierig.

Diese Probleme können durch eine automatisierte Berechnung der Trajektorie behoben werden. Basierend auf Hindernisinformationen und Robotereigenschaften soll die Bahn über KMC kollisionsfrei und optimal abfahrbar sein. Die bisherige Einzelachssteuerung für KHS-Roboter soll in eine Mehrachssteuerung überführt werden. Die Optimierungskriterien Fahrdauer, Ruck, Verschleiß, Energieverbrauch, genaue Bahnverfolgung, etc. müssen in Betracht gezogen und ggf. gegeneinander abgewogen werden. Das Bahnplanungssystems soll einerseits einen hohen Automatisierungsgrad für die Erstellung von Trajektorien ermöglichen, andererseits muss sie genügend Stellschrauben für die Inbetriebnahme anbieten und in vorhandene Steuerungen einfach integrierbar sein. Zudem muss sie eine hohe Wiederverwendbarkeit garantieren, so dass sie durch Parametrierung schnell auf neue Anwendungsfälle übertragen werden kann.

3. Modellierung des Industrieroboters

Die Gestalt einer zeitoptimalen Trajektorie ist abhängig vom verwendeten Roboter. In einem 6-achsigen Knickarmroboter treten andere Kräftearten als in einem 2-achsigen Schieber oder Säulenroboter der KHS auf. Eine optimierende Bahnplanung muss diese Kinematik- und Dynamik-spezifischen Kräfte berücksichtigen und nutzen. Deshalb ist eine mathematische Beschreibung des Roboters und der auftretenden Kräfte notwendig.

Ein Industrieroboter ist ein Mehrkörpersystem. Der Grundkörper ist mit dem Intertialsystem verankert, die übrigen Körper werden meist über elektrische oder pneumatische Antriebe bewegt. Entlang der kinematischen Kette treten dadurch nicht-lineare Wechselwirkungen mit komplexen Zusammenhänge für Kinematik und Dynamik auf.

Im Folgenden wird das Modell eines Roboters erstellt. Um die Modellierung allgemeingültig zu halten, soll die Kinematik einerseits alle Eigenschaften eines komplexen Roboters wie Zentrifugal-, Coriolis- und Abstützkräfte aufweisen, andererseits soll die Modellierung überschaubar bleiben. Ein SCARA-Roboter mit vier Bewegungsfreiheitsgraden erfüllt diese Anforderungen. Seine Modellierung kann einfach auf einen 2- oder 3-achsigen KHS-Säulenroboter übertragen werden.

Der Ausgangspunkt der Robotermodellierung ist seine Mechanik. **Abschnitt 3.1** geht kurz auf mechanische Beschreibungstechniken ein.

Die direkte Kinematik bildet die Gelenkstellungen auf die Lage der Roboterhand im kartesischen Raum ab. Die Umkehrung der Abbildung ist bei redundanten Kinematiken wie einem SCARA-Roboter nicht eindeutig. **Abschnitt 3.2** geht auf beide Themen mit der Denavit-Hartenberg-Methode ein.

Ausgehend von der Kinematik betrachtet die Dynamik die zeitliche Bewegung und stellt den Zusammenhang zwischen Kräften und Geschwindigkeiten der Armseg-

menten dar. In **Abschnitt 3.3** wird die Bewegungsgleichung mit der Lagrangeschen Energiefunktion L hergeleitet.

3.1 Modelle der Mechanik

Die Modellierung eines Roboters wird in zwei Stufen durchgeführt. Zuerst wird von unwesentlichen Bestandteilen des Roboters abstrahiert, danach wird ein Modell für die Bewegungsgleichung hergeleitet. Die Bewegung ergibt sich aus der Überlagerung einer gewollten Führungsbewegung mit elastischen Verformungen und Schwingungen.

Die mechanischen Eigenschaften eines Manipulators wie Trägheit, Elastizität, Reibung, Stellkräfte und Momente sind mit idealisierten Bauteilen zu modellieren. Hier stehen Starrkörper, elastische Körper, Feder, Dämpfer, Gelenke und Stellglieder zur Auswahl.

Kreuzer et al. [Truckenbrodt 94] führen drei Verfahren für Modellierung von Systemen an.

- In **Mehrkörpersystemen** sind Armsegmente als starre Glieder und Antriebselastizitäten als Feder beschrieben. Elastische Bewegungen können mit über Federn verbundene Balkenelemente modelliert werden.
  Der Vorteil dieses Verfahrens liegt in den einfachen Modellen. Nachteilig ist die schwierige Modellierung und die Parameterbestimmung von Federkonstanten, wenn Federn armsegment-übergreifend angeordnet sind. Werden solche Federkonstanten vernachlässigt, sind nur Grundschwingungen und keine Oberschwingungen erfassbar.

- Die Zerlegung eines Armsegments in viele gleichartige Elemente führt zu **Finite-Element-Systemen**. Anschließend werden die Kontaktpunkte dieser Elemente untersucht. Die Zusammenfassung der Elemente führt zum Gesamtkörper.
  Für statische Lastfälle und kleine Verformungen ist diese Methode gut geeignet. Sie ist jedoch wegen der hohen Zahl der Freiheitsgrade und des resultierenden Rechenaufwands schlecht bei Führungsbewegungen.

- In **kontinuierlichen Systemen** werden Differentialgleichungen für die exakte Beschreibung einfacher, elastischer Balkenkörper verwendet. Jedoch ist eine aufwendige mathematische Diskretisierung nötig und eine exakte Beschreibung komplexerer Körper schwierig.

Industrieroboter sind durch die Überdimensionierung der Armsegmente, der resultierenden Festigkeit und durch die Genauigkeit der Getriebe ausgezeichnet. Vor allem ist die SCARA-Konfiguration steif und belastbar, da die Bewegung der Armsegmente nur in horizontaler Ebene statt findet. Diese Ebene steht senkrecht zu der Bewegung des Hubteils. Deshalb ist die nur durch das Eigengewicht und Nutzlast hervorgerufene statische Biegebelastungen konstant und erlaubt eine hohe Genauigkeit.

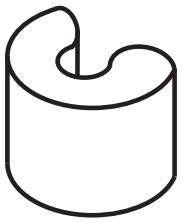


Abbildung 3.1: Arbeitsraum eines SCARA-Roboters

Die KHS verwendet zur Konstruktion der eigenen Roboter und Maschinen aus Kapitel 2 das Finite-Element-System, um die Stabilität der Armsegmente zu überprüfen. Kritische Bauteile werden mit dem FEM Programm ProMechanika berechnet und optimiert. Weniger kritische Teile werden mit Erfahrungswerten und auf Basis empirischer Optimierung ausgewählt. Mit den vorhanden Software-Werkzeugen nicht ermittelbare Eigenschaften der Mechanik wie Reibungswerte werden vernachlässigt.

Für die einfache analytische Modellierung des KHS Roboters mit Vernachlässigung der mechanischen Biegungen wird für diese Arbeit das Mehrkörpersystem gewählt.

3.2 Modelle der Kinematik

Die Kinematik beschreibt die Lage des Endeffektors im Intertialsystem. Abbildung 3.1 zeigt den Arbeitsraum des Robters für diesen Abschnitt.
Die Position und Orientierung des Endeffektors im kartesischen Raum wird über die homogene Transformation der Gelenkstellungen einfach durchgeführt. Die inverse Kinematik, die Bestimmung der Gelenkwinkel aus der Sollposition im kartesischen Raum, ist jedoch schwierig und nicht immer analytisch zu lösen.

Als Basis für die Berechnung der Kinematik dient eine allgemeine Beschreibung eines Roboters als Mehrkörpersystem, das mit Dreh- und Lineargelenken verbunden ist.

3.2.1 Universalkoordinaten und DH-Methode

Um die Lage der Roboterhand über die Ortskoordinatensysteme der kinematischen Kette im Inertialsystem auszudrücken, sind Translationen und Rotationen nötig. Translationen werden über Vektoren ausgedrückt, Rotationen über Rotationssmatrizen. Beide Operationen können vereint durch die harmonische 4×4-Matrix beschrieben werden.

Für einfache Roboter kann das Kinematikmodell durch eine geometrische Betrachtung und trigonometrische Beziehungen der Gelenkstellungen hergeleitet werden. Bei komplexeren Kinematiken existieren zwei Alternativen zur Herleitung der Lage eines Ortskoordinatensystems im Inertialsystem: Universalkoordinaten und die Denavit-Hartenberg-Methode.

Universalkoordinaten

Die Transformation der Manipulatorkoordinaten in Weltkoordinaten geschieht durch die Translation des Ortskoordinatensystems O_i entlang eines Armsegments zum nächsten Ortskoordinatensystem O_{i+1}. Nach dieser Abbildung der Ursprünge der Koordinatensystem aufeinander folgt eine Rotation, um die Orientierungen anzugleichen. Dieses Verfahren wird iterativ vom Fuß bis zur Hand angewandt. Jedes Glied der kinematischen Kette wird mit einem körperfesten Koordinatensystem verbunden. Die Koordinatentransformation zwischen zwei benachbarten Gliedern hängt nur von der Koordinate des verbindenden Gelenks ab.
Sechs Parameter werden pro Transformation benötigt: Drei für die Translation und drei für die Rotation.

Denavit-Hartenberg (DH)

Denavit und Hartenberg [Hartenberg 55] stellen fest, dass zur allgemeinen Beschreibung der Lage einer Geraden im Raum sechs Parameter verwendet werden können (drei für den Stützpunktvektor und drei für den Richtungsvektor), aber nur vier Parameter sind nötig, falls die Gerade mit den Gleichungen

$$\begin{aligned} x &= A\,z + B \\ y &= C\,z + D \end{aligned} \tag{3.1}$$

definiert wird.

Dieses Prinzip wird in [Hartenberg 55] zur Darstellung der Ortskoordinatensysteme von Gelenken angewandt und wird zu einer symbolischen Schreibweise mit homogenen Matrizen, die mit Matrixalgebra verarbeitet wird, weiterentwickelt.

Bei dem DH-Verfahren ist die Lage eines Koordinatensystems in der kinematischen Kette nicht beliebig wählbar, sondern es muss den DH-Konventionen entsprechen. Für die Anwendung dieser Modellierung sind Erfahrung oder eine Umrechnung in Universalkoordinaten erforderlich.

Die Kinematik der KHS-Roboter besitzt wenige Bewegungsfreiheitsgrade und lässt sich mit DH-Parametern und der DH-Konvention einfach beschreiben. Die KHS verwendet numerische Software-Tools und Tabellen zur Überprüfung der Kinematik während der Konstruktion. Diese liefern aber keine explizite Formeln. Deshalb werden in den folgenden Abschnitten die kinematischen Gleichungen für einen 4-achsigen SCARA-Roboter und den 3- und 2-achsigen KHS-Roboter hergeleitet.

3.2.2 Vorwärtskinematik

Die Vorwärtskinematik bildet die Gelenkstellungen auf den kartesischen Raum ab. Der Ausgangspunkt der folgenden Betrachtung ist die SCARA-Grundkonfiguration aus Abbildung 3.2.

Gegeben sind die Gelenkwinkel der Rotationsgelenke und die Auslenkung der Lineargelenke. Gesucht ist die Lage der Roboterhand in Weltkoordinaten. Für das

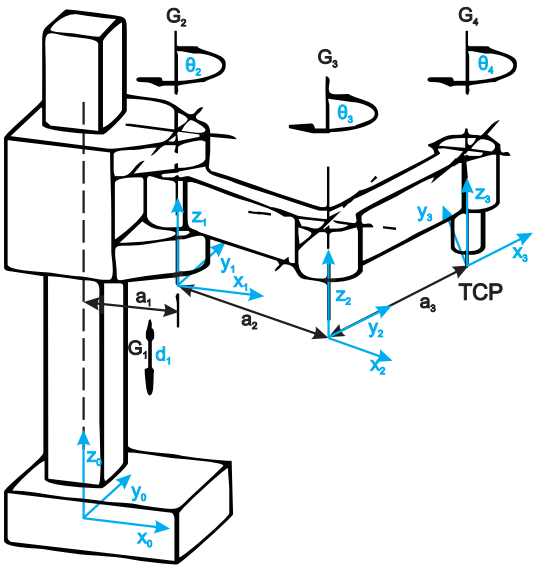


Abbildung 3.2: Festlegung der Koordinatensysteme nach der Denavit-Hartenberg-Konvention zur Beschreibung der Kinematik eines 4-achsigen SCARA Roboters

Gelenk	a_i	α_i	d_i	θ_i
1	a_1	0	d_1	0
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3
4	0	0	0	θ_4

Tabelle 3.1: DH-Parameter der SCARA-Kinematik

Gelenk	a_i	α_i	d_i	θ_i
1	0,5 m	0	$d_1 \in$ [1,5m; 5m]	0
2	1,4 m	0	0	$\theta_2 \in$ [-90°, 90°]
3	0	0	0	$\theta_3 \in$ [-180°, 180°]

Tabelle 3.2: DH-Parameter der Kinematik des 3-achsigen KHS-RS3

direkte kinematische Problem sind die Ansätze mit Universalkoordinaten und DH gleichwertig und führen zu dem gleichen Ergebnis.

Aus den Parametertabellen 3.1 und 3.2.2 leiten sich die homogenen Transformationsmatrizen nach [Hartenberg 55] ab.

Aus der Parameterbeschreibung folgt die Lage des Tool Center Points im kartesischen Raum[1]:

SCARA mit 4 Achsen:

$$\underline{X}_{TCP,SCARA}\left(\vec{\theta},\vec{d}\right) = \left[\underline{X}_{TCP,1},\, \underline{X}_{TCP,2},\, \underline{X}_{TCP,3},\, \underline{X}_{TCP,4}\right] \tag{3.2}$$

mit

$$\begin{aligned}
\underline{X}_{TCP,1} &= \begin{pmatrix} \cos\theta_2 \cdot (\cos\theta_3 \cdot \cos\theta_4 - \sin\theta_3 \cdot \sin\theta_4) - \sin\theta_2 \cdot (\cos\theta_3 \cdot \sin\theta_4 + \sin\theta_3 \cdot \cos\theta_4) \\ \cos\theta_2 \cdot (\cos\theta_3 \cdot \sin\theta_4 + \sin\theta_3 \cdot \cos\theta_4) + \sin\theta_2 \cdot (\cos\theta_3 \cdot \cos\theta_4 - \sin\theta_3 \cdot \sin\theta_4) \\ 0 \\ 0 \end{pmatrix} \\
\underline{X}_{TCP,2} &= \begin{pmatrix} \sin\theta_2 \cdot (\sin\theta_3 \cdot \sin\theta_4 - \cos\theta_3 \cdot \cos\theta_4) - \cos\theta_2 \cdot (\cos\theta_3 \cdot \sin\theta_4 + \sin\theta_3 \cdot \cos\theta_4) \\ \cos\theta_2 \cdot (\cos\theta_3 \cdot \cos\theta_4 - \sin\theta_3 \cdot \sin\theta_4) - \sin\theta_2 \cdot (\cos\theta_3 \cdot \cos\theta_4 - \sin\theta_3 \cdot \sin\theta_4) \\ 0 \\ 0 \end{pmatrix} \\
\underline{X}_{TCP,3} &= \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\
\underline{X}_{TCP,4} &= \begin{pmatrix} \cos\theta_2 \cdot (a_3 \cdot \cos\theta_3 + a_2) - a_3 \cdot \sin\theta_2 \cdot \sin\theta_3 + a_1 \\ a_3 \cdot \cos\theta_2 \cdot \sin\theta_3 + \sin\theta_2 \cdot (a_3 \cdot \cos\theta_3 + a_2) \\ d_1 \\ 1 \end{pmatrix} \\
&= {}_{Additionstheorem} \begin{pmatrix} a_3 \cdot \cos(\theta_2 + \theta_3) + a_2 \cdot \cos\theta_2 + a_1 \\ a_3 \cdot \sin(\theta_2 + \theta_3) + a_2 \cdot \sin\theta_2 \\ d_1 \\ 1 \end{pmatrix}
\end{aligned} \tag{3.3}$$

Die Hubsäulenroboter der KHS besitzen kein Ellenbogengelenk. Kinematik für den KHS-Einleger mit 3 Achsen und den RS3-Roboter:

$$\underline{X}_{TCP,RS3}\left(\vec{\theta},\vec{d}\right) = \begin{bmatrix} \cos\theta_2 \cdot \cos\theta_3 - \sin\theta_2 \cdot \sin\theta_3 & -\cos\theta_2 \cdot \sin\theta_3 - \sin\theta_2 \cdot \cos\theta_3 & 0 & a_2 \cdot \cos\theta_2 + a1 \\ \cos\theta_2 \cdot \sin\theta_3 + \sin\theta_2 \cdot \cos\theta_3 & \cos\theta_2 \cdot \cos\theta_3 - \sin\theta_2 \cdot \sin\theta_3 & 0 & a_2 \cdot \sin\theta_2 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

[1] Ausfürliche Herleitung der Denavit-Hartenberg Matrizen im Anhang A

Für die Variante des KHS-Einlegers mit 2 Achsen ohne Ellenbogen- und Handgelenk:

$$\underline{X}_{TCP,2-Achser}\left(\vec{\theta},\vec{d}\right) = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_2\cdot\cos\theta_2 + a1 \\ \sin\varphi_2 & \cos\theta_2 & 0 & a_2\cdot\sin\theta_2 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

3.2.3 Rückwärtskinematik

Die Trajektorie eines Roboters soll möglichst unabhängig von seiner konkreten Roboterkonfiguration erstellt werden können. Dazu wird die Bahn im kartesischen Raum erstellt und anschließend auf Gelenkwinkelkoordinaten rücktransformiert.
Die inverse Kinematik beschreibt ein komplexeres Problem, dessen Lösung trigonometrisch, analytisch, numerisch oder mit einer Kombinationen dieser Verfahren erstellt wird.

Analytische und geschlossene Lösungen sind bei komplexen kinematischen Strukturen schwer oder zum Teil gar nicht zu finden.
In solchen Fällen bieten sich **numerische** Lösungsverfahren an. Das Prinzip besteht darin die kinematische Struktur in einem Arbeitspunkt, in der Regel ist das die aktuelle Robotergelenkstellung, zu linearisieren.
Die **trigonometrische** Herangehensweise betrachtet Winkel und stellt trigonometrische Beziehungen her. Sie ist einfach zu realisieren, wobei für jede Kinematik ein individuelles Vorgehen erforderlich ist. Sie ist komplex bei viel-achsigen Roboterkonfigurationen und zudem sind trigonometrische Umkehrfunktionen nicht eindeutig.

Im Folgenden werden das analytische und das numerische Verfahren näher betrachtet.

Analytisch

Der Ausgangspunkt ist die Solllage $\underline{X}_{TCP}$. Die Linksmultiplikation der DH-Matrizen führt zu

$$\underline{X}_{TCP} = \underline{A}_{0,n} = \prod_{i=1}^{n} \underline{A}_{i-1,i}\,. \tag{3.6}$$

Die Anzahl der Matrizengleichungen entspricht der Zahl der Bewegungsachsen. Z. B. gilt für 3 Bewegungsachsen

$$\underline{X}_{TCP} = \underline{A}_{0,3} \tag{3.7}$$

$$\underline{A}_{0,1}^{-1}\cdot\underline{X}_{TCP} = \underline{A}_{1,3} \tag{3.8}$$

$$\underline{A}_{1,2}^{-1}\cdot\underline{A}_{0,1}^{-1}\cdot\underline{X}_{TCP} = \underline{A}_{2,3}\,. \tag{3.9}$$

Die linke Seite der Gleichungen besteht aus Funktionen von $\underline{X}_{TCP}$ und den ersten n Bewegungsachsen. Die rechte Seite aus Funktionen der $n+1$ bis 3 Bewegungsachsen. Jede Matrizengleichung führt zu 12 nicht-trivialen Gleichungen. Insgesamt sind 36

Gleichungen zu untersuchen, um die Gelenkvariablen freizustellen.

Der Vorteil dieses Verfahrens ist die festgelegte Vorgehensweise. Für komplexe Manipulatoren entsteht jedoch ein hoher Rechenaufwand. Varianten dieses Verfahrens arbeiten mit Vereinfachung oder lösen sich von den DH-Konventionen los. Sie führen aber nur zu geringen Effizienzsteigerungen.

Numerisch

Der numerische Lösungsweg geht von einem Startwert aus und approximiert mit einem Iterationsverfahren die Zielstellung $\vec{X} = [x,\ y,\ z,\ \phi,\ \theta,\ \psi]^T$ über den Zusammenhang $\vec{X} = \vec{f}\,(\vec{q})$ mit den Gelenkstellungen $\vec{q}$.
Der Zusammenhang zwischen Gelenkgeschwindigkeit und Ableitung von $\vec{X}$ berechnet sich zu

$$\begin{aligned} \dot{\vec{q}} &= J^{-1}\,(\vec{q}) \cdot \dot{\vec{X}} \\ \text{falls Jakobimatrix } J\,(\vec{q}) &= \frac{\partial \vec{f}}{\partial \vec{q}} \text{ regulär} \\ \text{Quantisierung} : \Delta\vec{q} &= J^{-1}\,(\vec{q}) \cdot \Delta\vec{X}\ , \end{aligned} \tag{3.10}$$

mit dem Newton-Verfahren wird anschließend q approximiert.

Das numerische Verfahren erfordert nur eine einfache Gleichungsaufstellung und ist auch für komplexe Konfigurationen anwendbar. Nachteilig ist die lange Rechenzeit.

Allgemeine Probleme der Rückwärtskinematik

Mehrdeutigkeiten treten bei überbestimmten Robotern wie in Abbildung 3.3 auf. Sie besitzen mehrere unterschiedliche Gelenkstellungen um die gleiche vorgegebene Lage mit dem Endeffektor anzufahren. Die Lösung des Matrizengleichungssystems (3.9) ist mehrdeutig. Das numerische Verfahren liefert eine eindeutige Lösung: Sie liegt dem Startwert am nächsten.
Bei der SCARA-Kinematik treten Mehrdeutigkeiten auf, aber für den KHS-RS3 und den Einleger nicht.

Speziallagen sind Lagen, in denen die Änderung einer Gelenkstellung (nicht Handgelenke) keinen Einfluss auf die Position des TCP hat. Singularitäten sind ebenfalls Speziallagen. Sie treten bei dem Sprung von π nach $-\pi$ bei unbeschränkt drehbaren Gelenken auf.
Nur für numerische Verfahren stellen Speziallagen ein Problem dar. Wenn sie in die Nähe dieser Lage kommen, entstehen Fehler.

Für SCARA treten Singularitäten nur im Handgelenk auf, da andere Gelenkwinkel mechanisch begrenzt sind.

Kreuzer et al. [Truckenbrodt 94] schätzen die Laufzeitkomplexität und die Häufigkeit der Verwendung von mathematischen Funktionen bei der Berechnung der Rückwärtskinematik ab. Die Abbildung 3.4 zeigt die Ergebnisse und stellt den praktischen

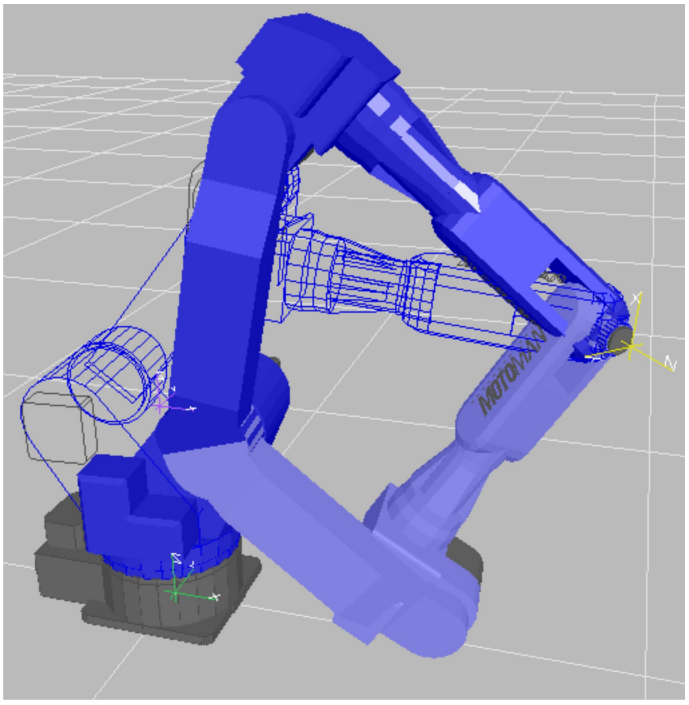


Abbildung 3.3: Mehrdeutigkeit bei einem kinematisch überbestimmten Manipulator, aus [Anton 04] S. 4

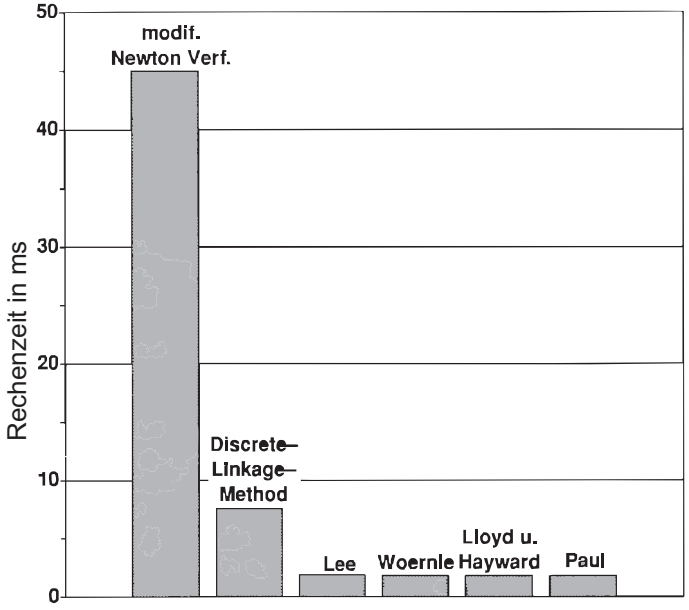


Abbildung 3.4: Aufwandsbetrachtung von analytischen und numerischen Verfahren der inversen Kinematik. Die numerische Methode nach Newton besitzt die schlechteste Ausführungszeit, aus [Truckenbrodt 94] S. 55

Vergleich der Laufzeiten der vorgestellten Verfahren auf einem HP1000-Rechner dar. Sie bestätigt, dass numerische Methoden aufgrund ihrer langen Laufzeit nur angewendet werden sollten, wenn auf dem analytischen Weg bei nicht-redundanten Manipulatoren keine Lösung gefunden wird oder wenn ein überbestimmter Roboter vorliegt.

Aufgrund der einfachen Roboterkonfiguration des SCARA und der KHS-Roboter wird das trigonometrische Verfahren angewendet. Nach der geometrischer Betrachtung der SCARA-Kinematik[1] folgt für die inverse Kinematik (vgl. Abbildung 3.2)

$$\begin{aligned} d_1 &= z \\ \theta_2 &= \arccos\left(-\frac{a_3^2 - a_2^2 - (x-a_1)^2 - y^2}{2\,a_2\sqrt{(x-a_1)^2+y^2}}\right) + \operatorname{atan2}(y, x-a_1) \\ \theta_3 &= \alpha - \pi = \arccos\left(-\frac{(x-a_1)^2 + y^2 - a_2^2 - a_3^2}{2\,a_2\,a_3}\right) - \pi \\ \theta_4 &= \psi - \theta_2 - \theta_3\,. \end{aligned} \tag{3.11}$$

Für den KHS RS3 mit 3 Achsen ohne Ellenbogengelenk gilt

$$\begin{aligned} d_1 &= z \\ \theta_2 &= \operatorname{atan2}(y, x-a_1) \\ \theta_3 &= \psi - \theta_2\,. \end{aligned} \tag{3.12}$$

Für den 2-achsigen Einleger ohne Ellenbogen- und Handgelenk gilt

$$\begin{aligned} d_1 &= z \\ \theta_2 &= \operatorname{atan2}(y, x-a_1)\ . \end{aligned} \tag{3.13}$$

3.3 Modelle der Dynamik

Im vorherigen Abschnitt wurde die Kinematikmodellierung eingeführt, um Gelenkwinkelstellungen auf die Lage der Roboterhand abzubilden und umgekehrt. Die Dynamik erweitert die Roboterbeschreibung um zeitabhängige Größen: Kräfte, Drehmomente, Geschwindigkeiten und Beschleunigungen.
Im Roboter als kinematische Kette treten nicht-lineare Wechselwirkungen zwischen den Armsegmenten auf und führen zu Störungen in der Regelung. Die genaue Kenntnis der Dynamik ist die Grundlage für den Entwurf der Roboterregelung, die Erstellung des Geschwindigkeitsprofils und die Bahnoptimierung. Die Dynamik liefert Antworten auf zwei Probleme: Was sind die besten Roboterparameter wie Massen,

[1] Herleitung der inversen SCARA-Kinematik im Anhang B

Armlängen, Motorisierung zur Lösung einer Aufgabe und welche Stellkräfte sind nötig, um bestimmte Gelenkgeschwindigkeiten zu erreichen.

Analog zur direkten und inversen Kinematik existiert das Problem der direkten Dynamik mit der Abbildung von Stellkräften auf die Bewegung der Gelenke und der inversen Dynamik mit der Ermittlung der nötigen Stellkräfte auf Basis der gewünschten Gelenkbewegung.

Es existieren zwei prinzipielle Ansätze zur Herleitung des Dynamik-Modells: **Analytische** Verfahren nach Lagrange und **rekursive** Methoden wie das Newton-Euler-Verfahren. Während Lagrange mit der Energiebetrachtung ein analytisch auswertbares und geschlossenes Modell hervorbringt, bedient sich Newton-Euler des D'Alembertisches Prinzips mit dem Impuls- und Drallsatz, schneidet Massen frei und führt eine rekursive Berechnung durch. Die Lagrange-Berechnung beinhaltet eine teilweise komplizierte Differenzierung der Energieausdrücke, liegt in $\mathcal{O}(n^4)$ und liefert nur Antriebsmomente. Newton-Euler besitzt einen $\mathcal{O}(n)$-Aufwand, verwendet aber Rekursion.

Das Ziel der Dynamikmodellierung ist die Feststellung des Zusammenhangs zwischen Kraft und hervorgerufener Bewegung. Dieser wird durch die Bewegungsgleichung ausgedrückt und kann nach Berns [Berns 05] wie folgt dargestellt werden:

$$\vec{Q} = \underline{M}(\vec{q})\,\ddot{\vec{q}} + \vec{n}(\vec{q},\dot{\vec{q}}) + \vec{g}(\vec{q}) + \underline{R}\,\dot{\vec{q}} \tag{3.14}$$

mit

$\vec{Q}$	Stellkräfte und -momente der Gelenke
$\underline{M}$	Massenträgheit
$\vec{n}$	Zentrifugal- und Corioliskomponenten
$\vec{g}$	Gravitationskomponenten
$\underline{R}$	Diagonalmatrix der Reibung
$\vec{q}$	Winkellage und Auslenkung der Gelenke .

Die Bewegungsgleichung wird auf Basis der Roboterparameter wie in Tabelle 3.3 und den Antriebsparametern wie in Abbildung 3.5 erstellt. Die genaue Erfassung aller Roboterparameter stellt sowohl für die Kinematik als auch für die Dynamik ein Problem dar. Aurnhammer [Aurnhammer 04] verwendet für die Beschreibung der Ungenauigkeiten ein durchgängig stochastisches Robotermodell.
In der KHS sind nur Werte exakt bekannt, die entsprechende Planungsprogramme liefern. Trägheitsmomente, Reibungskoeffizienten und Steifigkeit werden empirisch ermittelt, geschätzt oder bei der Konstruktion vernachlässigt.

Eine Kraft stellt eine Wechselwirkungen zwischen Gliedern der kinematischen Kette dar. Sie ist lageabhänging und eingeprägt. Bei schnellen Bewegungen kommen Zentrifugal- und Corioliskräfte hinzu.

Roboterachse	1 Hub	2 Arm	3 Kopf
Motorbezeichnung von Fa. SEW je -B/HR/TH	DY90L	DFY90L	DFY71S
Arbeitsbereich φ in $^\circ$	-	240	190
max. Geschwindigkeit $\dot{\varphi}$ in $^\circ/s$	-	ableitbar	ableitbar
max. Beschleunigung $\ddot{\varphi}$ in $^\circ/s^2$	-	ableitbar	ableitbar
Arbeitsbereich d in m	3,5	-	-
max. Geschwindigkeit $\dot{d}$ in m/s	ableitbar	-	-
max. Beschleunigung $\ddot{d}$ in m/s^2	ableitbar	-	-
Getriebeübersetzung i	13,52	141	141
Verhältnis Motor-Gelenk in m	100:1		
Trägheitsmoment Motor I_i in $kg\,m^2$	$32,6\,10^{-4}$	$32,6\,10^{-4}$	$5,46\,10^{-4}$
Trägheitsmoment Getriebe I_i in $kg\,m^2$	$17,82\,10^{-4}$	$6,35\,10^{-4}$	$0,638\,10^{-4}$
Trägheitsmoment um Gelenkachse J_{bew} in kg m^2	-	845,6534	53,5269
Masse in kg	50	109	210
Träge Masse in kg	100	109	210
Schwerpunkte von Drehachse entf. in m		1,5084	0,025
maximales Motormoment $\widehat{M_i}$ in Nm	55	55	7,5
effektives max. Dauermoment M_i in Nm	18	18	2,5
effektives Dauermoment mit Kühlung M_i in Nm	28	28	4
Motor-Nenndrehzahl n in s^{-1}	3000	3000	3000
statische Gelenkreibung			
geschw. prop. Reibung			

Tabelle 3.3: Dynamische Kenngrößen des KHS-Einlegers, aus [SEW 07], KHS-Angaben

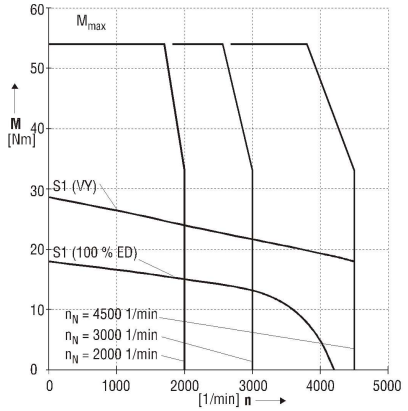


Abbildung 3.5: Kennlinie des Synchromotors DFY 90L der Fa. SEW, aus [SEW 07]

Kräfte lassen sich allgemein wie folgt klassifizieren:

- **äußere Kraft**: einfaches Vorkommen
- **innere Kraft**: paarweises Auftreten, Kompensation bei der Summation
- **eingeprägt** nach Kraftgesetz: elastische Bauteile, Dämpfer, Motoren, Gleitreibung
- **reaktiv** nach Bindung

Die Gewichtskraft ist beispielsweise eine äußere, eingeprägte Kraft, das Antriebsmoment dagegen ist eine innere, eingeprägte Kraft.

3.3.1 Bewegungsgleichungen nach Lagrange

Die Herleitung und Echtzeit-Auswertung der Bewegungsgleichung in schnellen Taktraten stellt hohe Anforderungen an die Leistung der Steuerung. Ramos und Khosla [Khosla 88] analysieren die Hardwareanforderungen für rekursive Lagrange-Euler- und Newton-Euler-Verfahren. Die Algorithmen werden durch eine mathematische Dekomposition für Parallelverarbeitung auf VLSI (ASIC)- und Multiprozessor-Rechnerarchitekturen optimiert. Der Einsatz von Newton-Euler in Kombination mit einem auf Gernal Purpose-CPUs basierendem Multiprozessor-Rechner brachte die besten Ergebnisse.
Durch die hohe Rechenleistung aktueller Standard-Systeme, vor allem der PC-basierten Kuka SoftSPS, ist der Einsatz einer spezialisierten Rechnerarchitektur nicht nötig. Dennoch wird die Bewegungsgleichung für die SCARA-Kinematik in den folgenden Abschnitten nach Lagrange analytisch hergeleitet, so dass die online-Berechnung während der Fahrt auf die Auswertung reduziert werden kann.

Die Lagrangesche Energiefunktion L berechnet sich zu

$$L = E_{kin} - E_{pot} \tag{3.15}$$

$$Q_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \tag{3.16}$$

mit Q: verallgemeinerte Kräfte, q: verallgemeinerte Lagekoordinaten.

Auf den SCARA-Roboter angewendet: Q: Drehmomente der Rotationsantriebe bzw. Kraft des Hubantriebs, q: Gelenkwinkel von Arme und Hand bzw. Gelenkstellung des Hubsegments.

3.3.2 Herleitung der Energien

Wird ein starrer Körper im Inertialsystem bewegt, so beträgt seine kinetische Energie

$$E_{kin} = \frac{1}{2}M\vec{v}^2 + M\vec{v}\,(\vec{\omega} \times \vec{r}_s) + \frac{1}{2}\underline{I}\vec{\omega}^2\,, \tag{3.17}$$

mit

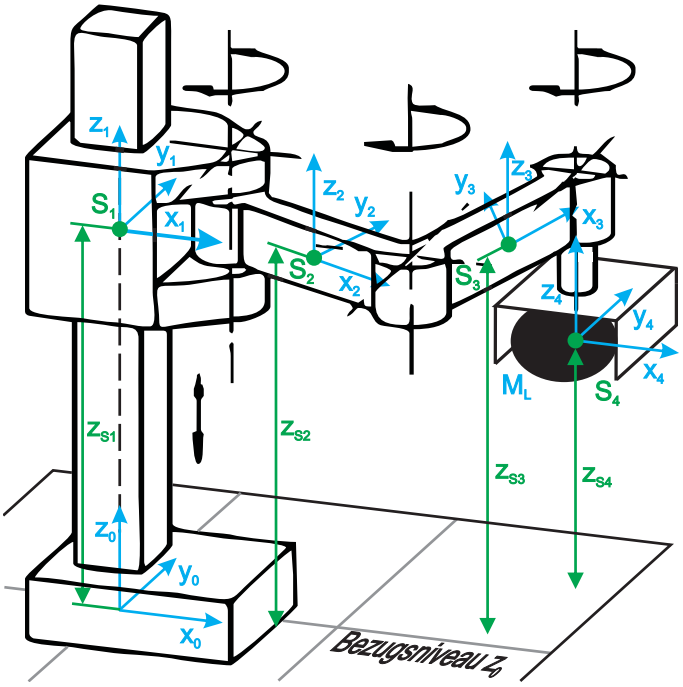


Abbildung 3.6: Festlegung der körperfesten Koordinatensysteme in den Schwerpunkten für die Beschreibung der SCARA-Dynamik

v	Geschwindigkeit des körperfesten Ursprungs im Inertialsystem
M	Masse des Körpers
$\vec{r}_s$	Schwerpunktvektor
ω	Winkelgeschwindigkeit
$\underline{I}$	Trägheitsmatrix bezogen auf körperfesten Ursprung.

$\underline{I}$ ist nur auf der Hauptdiagonalen besetzt, wenn Koordinatenachsen mit den Hauptträgheitsachsen übereinstimmt. Die Lageenergie bei gegebener Höhe h lässt sich zu

$$E_{pot} = M\,g\,h \tag{3.18}$$

berechnen.

Abbildung 3.6 zeigt die Festlegung der Koordinatensysteme. Im Gegensatz zur kinematischen Beschreibung aus Abbildung 3.2 befinden sich die Ursprünge in den Schwerpunkten der Robotersegmente. Die beiden Armsegmente lassen sich mit hinreichender Genauigkeit als Balkenkörper mit homogener Massenverteilung modellieren. Der Schwerpunkt entspricht dem Mittelpunkt. Die Roboterhand mit Last sei achsensymmetrisch zur Drehachse z_4 und der Hubkörper habe seinen Schwerpunkt auf der Achse z_1. Somit reduziert sich die Trägheitsmatrix der Armsegmente und der Hand mit Last auf die Elemente der Hauptdiagonale. Da der Hubkörper nicht rotiert wird, ist nur die Höhe seines Schwerpunktes von Bedeutung.

Energie des Hubkörpers, Lineargelenk

Das körperfeste Koordinatensystem K_1 wird in die Mitte der Hubsäule gelegt. Seine Höhe entspricht dem Mittelpunkt des Arms. Da seine Rotationsgeschwindigkeit stets 0 beträgt, ergibt sich für die die kinetische Energie

$$\begin{aligned} E_{kin1} &= \frac{1}{2} M_1 \vec{v_1}^2 \\ \vec{v_1} &= \begin{pmatrix} 0 \\ 0 \\ v_{z1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \dot{d}_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ v_1 \end{pmatrix} \\ &\Rightarrow \quad E_{kin1} = \frac{1}{2} M_1 v_1^2 \,. \end{aligned} \tag{3.19}$$

Das Bezugsniveau der potentiellen Energie sei $z_0 = 0$. Für die Lageenergie des Hubkörpers gilt

$$E_{pot1} = M_1 \, g \, z_{S1} = M_1 \, g \, d_1 \,, \tag{3.20}$$

mit

M_1	Masse des Hubkörpers
g	Erdbeschleunigung
z_{S1}	Z-Koordiate des Schwerpunkts S_1
d_1	Auslenkung des Lineargelenks.

Energie des Oberarms, Rotationsgelenk

Das Koordinatensystem K_2 befindet sich körperfest in dir Mitte des Oberarms. Seine Energien berechnen sich mit den Ergebnissen der Kinematik (3.3) aus

$$\begin{aligned} E_{kin2} &= \frac{1}{2} M_2 \vec{v_2}^2 + \frac{1}{2} \underline{I}_2 \vec{\omega}_{k2}^2 \\ &\text{mit } \vec{v_2} = \begin{pmatrix} \dot{x}_{S2} \\ \dot{y}_{S2} \\ \dot{z}_{S2} \end{pmatrix} = \frac{d}{dt} \begin{pmatrix} \frac{a_2}{2} \cos\theta_2 + a_1 \\ \frac{a_2}{2} \sin\theta_2 \\ d_1 \end{pmatrix} \\ &\vec{v_2}^2 = \langle \vec{v_2}, \vec{v_2} \rangle = \frac{1}{4} a_2^2 \, \omega_2^2 + v_1^2 \\ &\text{mit } \omega_2 = \dot{\theta}_2 \\ &\vec{\omega}_{k2} = \begin{pmatrix} 0 \\ 0 \\ \omega_2 \end{pmatrix} \\ &\Rightarrow \frac{1}{2} \underline{I}_2 \, \vec{\omega}_{k2}^2 = \frac{1}{2} I_{z2} \, \omega_2^2 \\ \Rightarrow \quad & E_{kin2} = \frac{1}{2} M_2 \left(\frac{1}{4} a_2^2 \, \omega_2^2 + v_1^2 \right) + \frac{1}{2} I_{z2} \, \omega_2^2 \end{aligned} \tag{3.21}$$

und

$$E_{pot2} = M_2 \, g \, d_1 \, . \tag{3.22}$$

Energie des Unterarms, Rotationsgelenk

Das körperfeste Koordinatensystem K_3 des Unterarms liegt im Schwerpunkt S_3. Für die Energien des Unterarms folgt analog zum Oberarm:

$$\begin{aligned} E_{kin3} &= \frac{1}{2} M_3 \vec{v_3}^2 + \frac{1}{2} \underline{I}_3 \vec{\omega}_{k3}^2 \\ &\text{mit } \vec{v_3} = \begin{pmatrix} \dot{x}_{S3} \\ \dot{y}_{S3} \\ \dot{z}_{S3} \end{pmatrix} = \frac{d}{dt} \begin{pmatrix} \frac{a_3}{2} \cos(\theta_2 + \theta_3) + a_2 \cos\theta_2 + a_1 \\ \frac{a_3}{2} \sin(\theta_2 + \theta_3) + a_2 \sin\theta_2 \\ d_1 \end{pmatrix} \\ \Rightarrow \quad & \vec{v_3}^2 = \\ & \left(a_2 \, a_3 \, (\cos(\theta_2) \cos(\theta_2 + \theta_3) + \sin(\theta_2) \sin(\theta_2 + \theta_3)) + \frac{1}{4} \left(4 \, a_2^2 + a_3^2 \right) \right) \omega_2^2 + \frac{1}{4} a_3^2 \, \omega_3^2 \\ & + \left(a_2 \, a_3 \, (\cos(\theta_2) \cos(\theta_2 + \theta_3) + \sin(\theta_2) \sin(\theta_2 + \theta_3)) + \frac{1}{2} a_3^2 \right) \omega_2 \, \omega_3 + v_1^2 \\ \text{und} \quad & \vec{\omega}_{k3} = \begin{pmatrix} 0 \\ 0 \\ \omega_2 + \omega_3 \end{pmatrix} \\ & \Rightarrow \frac{1}{2} \underline{I}_3 \, \vec{\omega}_{k3}^2 = \frac{1}{2} I_{z3} \, (\omega_2 + \omega_3)^2 \end{aligned} \tag{3.23}$$

$$E_{pot3} = M_3 \, g \, d_1 \tag{3.24}$$

Energie der Hand, Revolvergelenk

Die Modellierung der Hand beinhaltet auch den Lastkörper. Der Schwerpunkt der Hand liege im TCP. Eine mitgeführte Last wird als symmetrisch ausgedehnte Masse mit der Trägheitsmatrix $\underline{I}_L$ dargestellt.

$$
\begin{aligned}
E_{kin4} \quad &= \quad \frac{1}{2}\left(M_4 + M_L\right)\vec{v_4}^{\,2} + \frac{1}{2}\underline{I}_4\vec{\omega}_{k4}^{\,2} \\
&\quad \text{mit } \vec{v_4} = \begin{pmatrix} \dot{x}_{S4} \\ \dot{y}_{S4} \\ \dot{z}_{S4} \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} a_3\,\cos\left(\theta_2+\theta_3\right) + a_2\,\cos\,\theta_2 + a_1 \\ a_3\,\sin\left(\theta_2+\theta_3\right) + a_2\,\sin\,\theta_2 \\ d_1 \end{pmatrix} \\
&\Rightarrow \quad \vec{v_4}^{\,2} = \\
&\quad \left(2\,a_2\,a_3\,\left(\cos\left(\theta_2\right)\,\cos\left(\theta_2+\theta_3\right) + \sin\left(\theta_2\right)\,\sin\left(\theta_2+\theta_3\right)\right) + \left(a_2^2 + a_3^2\right)\right)\,\omega_2^2 + a_3^2\,\omega_3^2 \\
&\quad + \left(2\,a_2\,a_3\,\left(\cos\left(\theta_2\right)\,\cos\left(\theta_2+\theta_3\right) + \sin\left(\theta_2\right)\,\sin\left(\theta_2+\theta_3\right)\right) + 2\,a_3^2\right)\,\omega_2\,\omega_3 + v_1^2 \\
&\text{und} \quad \vec{\omega}_{k4} = \begin{pmatrix} 0 \\ 0 \\ \omega_2+\omega_3+\omega_4 \end{pmatrix} \\
&\quad \Rightarrow\; \frac{1}{2}\underline{I}_4\vec{\omega}_{k4}^{\,2} = \frac{1}{2}\left(I_{z4} + I_{zl}\right)\,\left(\omega_2+\omega_3+\omega_4\right)^2
\end{aligned}
\tag{3.25}
$$

$$
E_{pot4} = \left(M_4 + M_L\right)\,g\,d_1 \tag{3.26}
$$

Weitere Einflüsse: Motorenergie, Reibung, Getriebeelastizität

Die Motormomente werden über Getriebe auf die Gelenkachsen übertragen. Neben den Armsegmenten tragen die Antriebsträgheitsmomente am Getriebeausgang I_{ai} der schnell drehenden Antriebskomponenten (Antriebsmotor I_{mi}, Getriebe I_{gi}, Übersetzung u_i) zum Gesamtträgheitsmoment bei:

$$
I_{ai} = u_i^2\left(I_{mi} + \frac{1}{u_i^2}\left(I_{gi}\right)\right) \tag{3.27}
$$

Die Rotationsenergie der Antriebe kann für Motordrehzahlen, die in der Regel deutlich größer als die Drehzahl der Armsegmente ω_{ai} sind, mit

$$
E_{kin,ai} = \frac{1}{2}I_{ai}\,\omega_{ai}^2 \tag{3.28}
$$

approximiert werden.

Das Reibungsmoment der Gelenkkörper m_{rg} und Antriebe m_{ra} lässt sich nach dem Verfahren von Shiller, Chang und Wong [Wong 96] wie in Abbildung 3.7 mit einem konstanten Teil der Coulombreibung h und einem geschwindigkeitsproportionalen k-Teil durch

$$
\begin{aligned}
m_{rai} &= h_{ai}\,sign\left(\omega_{ai}\right) + k_{rai}\,\omega_{ai} \\
m_{rgi} &= h_{gi}\,sign\left(\omega_{gi}\right) + k_{rgi}\,\omega_i
\end{aligned}
\tag{3.29}
$$

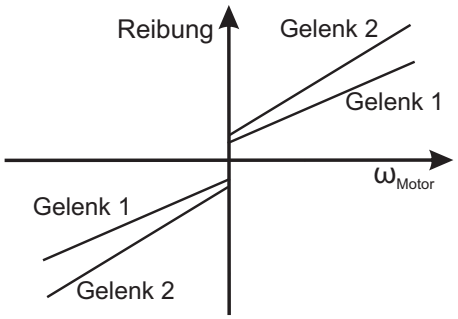


Abbildung 3.7: Approximation der Gelenkreibung mit einem Modell erster Ordnung, angelehnt an [Wong 96] S. 34

annähern. Diese Annäherung ist für kleine Geschwindigkeiten hinreichend genau. Für hohe Geschwindigkeiten sollte ein Modell der zweiten Ordnung verwendet werden.

Hohe Getriebeübersetzungen führen zu einer Getriebeelastizität. Das elastische Getriebe kann als Kombination aus einem idealen starren Getriebe und einer Feder modelliert werden. Das Getriebe verursacht die potentielle Energie $\underline{E}_{potG}$:

$$\begin{aligned} m_{ki} &= k_{fi}\,(\theta_{ai} - \theta_i) \\ \underline{E}_{potG,i} &= \frac{1}{2} k_{fi}\,(\theta_{ai} - \theta_i)^2 \end{aligned} \tag{3.30}$$

mit

m_{ki}	durch die Feder übertragenes Drehmoment
θ_{ai}	Gelenkstellung vor der Feder
θ_i	Gelenkstellung nach der Feder
k_{fi}	Torsionssteifigkeit der Feder

Abbildung 3.8 zeigt mechanische Aufbauten zur Kostenoptimierung. Gegengewichte und pneumatische Gegenkräfte kompensieren Gewichtsdrehmomente, führen zu einer Reduktion des Motormoments und erlauben den Einsatz kleinerer und kostengünstiger Motoren. Ein Gegengewicht kann als träge Masse

$$\begin{aligned} E_{kin1} &= \frac{1}{2}\,(M_1 + M_{gegen})\,\vec{v_1}^2 \\ E_{pot1} &= (M_1 - M_{gegen})\; g\, z_{S1} \end{aligned} \tag{3.31}$$

modelliert werden, eine pneumatische Gegenkraft als zusätzliche potentielle Energie

$$E_{pot,pneu} = \int_0^{\theta_1} D\,(\theta)\,\theta\,d\theta \tag{3.32}$$

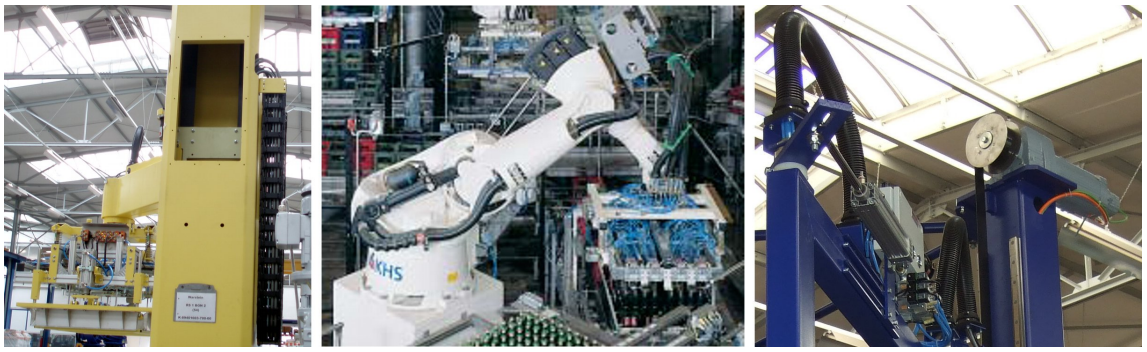


Abbildung 3.8: Modellierung weiterer Eigenschaften der Dynamik: Gegengewicht des Hubteils des RS3 als träge Masse (links), Pneumatikzylinder am Fußgelenk des Kuka-Roboters mit stellungsabhängigem pneumatischen oder hydraulischem Drehmoment (mitte) und nicht-linearer Zusammenhang zwischen Motor- und Gelenkstellung des Einlegers mit Aufwicklung des Hub-Riemens (rechts)

mit der stellungsabhängigen Federhärte $D(\theta)$.

Zwischenlageneinleger mit geringen Anforderungen an Traglast und Positioniergenauigkeit sind mit aufwickelnden Hubgelenken ausgestattet. Die Regler-Stellgrößen Motorstrom und -drehzahl werden in einem erweiterten, 2-stufigen Kinematik- und Dynamikmodell hergeleitet. Die TCP-Lage wird über die Gelenkstellung auf die Motorstellung abgebildet: Lage des TCP $\leftrightarrow$ Stellung der Gelenke $\leftrightarrow$ Stellung der Motoren bzw. TCP-Geschwindigkeit $\leftrightarrow$ Gelenkgeschwindigkeit $\leftrightarrow$ Motordrehzahl. Bei dem RS3-Roboter bildet ein idealisiertes Getriebe Gelenkzustand und Motorzustand linear aufeinander ab. Für den aufwickelnden Zwischenlageneinleger ist die Abbildung spezifisch zu dem mechanischen Aufbau, Durchmesser der Rolle und der Dicke des Riemens.

3.3.3 Komplette Bewegungsgleichung

Aus den kinetischen und potientiellen Energien des vorigen Abschnitts berechnet sich die Lagrangesche Energiefunktion zu

$$L = \sum_{i=1}^{4} \underline{E}_{kin,i} + \underline{E}_{kinA,i} - \underline{E}_{pot,i} - \underline{E}_{potG,i} \,. \tag{3.33}$$

Für die Antriebe (vor der Getriebeelastizität) gilt

$$m_{ai} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_{ai}}\right) - \frac{\partial L}{\partial \theta_{ai}} + m_{rai} \,, \tag{3.34}$$

für die Gelenkkörper (nach der Getriebeelastizität):

$$0 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_{i}}\right) - \frac{\partial L}{\partial \theta_{i}} + m_{rgi} \,. \tag{3.35}$$

Da bei Industrierobotern oft auf aufwändige Antriebssensorik verzichtet wird, zeichnen sie sich durch präzise, unelastische Getriebe aus. Die Vernachlässigung der Getriebeelastizität führt zu der vereinfachten Gleichung

$$m_{ai} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} + m_{rai} + m_{rgi} \tag{3.36}$$

mit $\theta_i = \theta_{ai}$ und $\omega_i = \omega_{ai}$.

Gleichungen (3.21) bis (3.26) und (3.36) liefern die Bewegung für die einzelnen Gelenkachsen[3]. Die Bewegungsgleichungen lassen sich wie in [Olomski 89] zu einer differenzierteren Form als in (3.14) mit

$$m_{ai} = I_i\,\dot{\omega}_i + \sum_j a_{ij}\,\dot{\omega}_j + \sum_j c_{ij}\,\omega_i\,\omega_j + \sum_j z_{ij}\,\omega_j^2 + k_{ri}\,\omega_i + h_i\,\mathrm{sign}\,(\omega_i) + m_{sti} \qquad j = 1..4 \tag{3.37}$$

mit

I_i	Trägheitsmoment des Antriebs und des Gelenkkörpers
a_{ij}	Faktor für Abstützmomente von Körper j auf i
c_{ij}	Faktor für Coriolismomente
z_{ij}	Faktor für Zentrifugalmomente
h_i	Faktor für Coulombreibung
k_{ri}	Faktor für geschwindigkeitsproportionale Reibung
m_{sti}	statische Momente: Gewichtsmoment

zusammenfassen.

[3]Herleitung der Dynamik-Gleichungen im Anhang C

3.4 Fazit

In diesem Kapitel wurde mit der mathematischen Beschreibung des Roboters die Grundlage für die Berechnung der optimalen Geschwindigkeit gelegt. Die Prinzipien der Kinematikmodellierung, der Abbildung der Gelenkstellungen $\vec{\theta}$ auf die Lage der Roboterhand im kartesischen Raum $\underline{X}_{TCP}$, basieren auf dem von Denavit und Hartenberg [Hartenberg 55] vorgestellten Verfahren. Mit diesem Verfahren und den Angaben der mechanischen Konstruktion der KHS wurden drei Kinematikmodelle entwickelt: 4-achsiger SCARA, 3-achsiger RS3 und ein 2-achsiger Einleger.

Ausgehend von dem kinematischen Modell wurde die Dynamik der Roboter hergeleitet. Aus alternativen Methoden wurde die Energiebetrachtung nach Lagrange, die in Kreuzer et al. [Truckenbrodt 94] beschrieben ist, ausgewählt und auf die drei Roboter angewandt. Die Bewegungsgleichung wurde um zusätzliche Randbedingungen wie Reibung und Getriebeelastizität ergänzt. Zudem führte die Betrachtung KHS-spezifischer Robotereigenschaften wie Gegengewichte der Säulenroboter und pneumatische Gegenkräfte zu Erweiterungen der Modellierung.

Die Herleitung der Kinematik- und Dynamikgleichungen der betrachteten Roboter ist im Anhang A bis C dargestellt.

4. Bahnplanung

Die Bahnplanung behandelt eines der Grundprobleme der Robotik: Wie gelangt ein Roboter A in seinem Arbeitsraum $\mathbb{W}$ mit den Hindernissen $\mathbb{B}$ von einer Initiallage ohne Kontakt mit $\mathbb{B}$ in eine Ziellage?
Während die Fragestellung nach einer automatischen Bahnplanung die mobile Robotik dominiert, spielt sie in der Industrierobotik eine untergeordnete Rolle. Die industrielle Umwelt mit allen Hindernissen ist exakt bekannt. Anstelle eines autonomen Verhaltens mit Exploration der unbekannten Umgebung und Berücksichtigung der Dynamik der Umwelt werden bei Industrierobotern die Bahnplanung großteils mit dem Teach-In-Verfahren durch den Bediener durchgeführt. In der KHS wird wegen der Vielfalt der Handhabungsaufgaben die Ziellage des TCP berechnet. Es ergeben sich auch Szenarien, bei denen die Robotersteuerung selbst die nächsten Schritte entscheiden muss. Dies ist nach einem manuellen Eingriff in die Bewegung des Roboters durch den Bediener der Fall. Einfache Lösungen fahren den letzten Punkt der regulären Bahn, bevor der Roboter manuell verfahren wurde, auf direktem Weg an. Eine Alternative ist die Anfahrt eines festen und immer auf direktem Weg erreichbaren Punkts (home). Beide Möglichkeiten sind nicht zeitoptimal und können zu Kollisionen führen. Zudem ermöglicht eine automatische Bahnplanung die Unterstützung des Bedieners und kann die Plausibilität der Eingaben überprüfen. Die automatische Bahnplanung kann alternative Wege finden, die Lage der Stützpunkte optimieren und verkürzt die empirische Optimierung.
Nachdem die Bahnplanung aus Start- und Ziellage eine Menge an Zwischenpunkten ermittelt hat, muss aus diesen Informationen eine kontinuierliche Bahn erstellt werden. Neben der Raumkurve im kartesischen Raum sind die zu jedem Bahnpunkt zugehörigen Gelenkwinkel zu erzeugen. Um ein ruckfreies und schnelles Fahren zu ermöglichen, muss die erstellte Kurve glatt und effizient online berechenbar sein.
Losgelöst von der Bahn wird die Geschwindigkeit unter Einhaltung von Randbedingungen berechnet.

Abschnitt 4.1 beschäftigt sich mit der automatischen Erzeugung von Bahnstützpunkten. Da Bahnplanungsverfahren vor allem aus der mobilen Robotik motiviert sind, werden diese Verfahren hinsichtlich der Anwendbarkeit für den online-Einsatz

mit Industrierobotern beschrieben. Zudem wird auf die Wellenfrontexpansion für eine Planung online näher eingegeangen.

Verschiedene Interpolationsverfahren für geometrische Raumkurven werden in **Abschnitt 4.2** vorgestellt. Er geht detailliert auf eine effiziente Variante von kubischen Splines ein, die mit linearen Abschnitten kombiniert werden können.

Abschnitt 4.3 befasst sich mit der Berechnung des Geschwindigkeitprofils auf der kartesischen Bahn. Es wird dargestellt, wie die Roboterdynamik, Randbedingungen und Benutzervorgaben in die Geschwindigkeit einfließen.

Abschnitt 4.4 geht auf die Gelenkregelung ein und führt einen Mechanismus zur Fahrt über die zuvor berechnete Trajektorie ein.

4.1 Kollisionsvermeidung und Bahnplanung

Nach Latombe [Latombe 96], Hsu, Kindel, Rock [Hsu 00], Chen und Xi [Chen 03] kann zwischen vier Verfahren zur Bahnplanung differenziert werden:

- Roadmap
- Zellendekomposition
- Potentialfeldmethode
- Zufall

Latombe [Latombe 96] führt zunächst den **Konfigurationsraum** ein. Damit lässt sich die Lage der Roboterhand auf einen Punkt im Gelenkwinkelraum abbilden und einfach beschreiben. Auf **Roadmap** basierende Methoden reduzieren die Suche nach einem kollisionsfreien Weg von der Start- zur Ziellage im Konfigurationsraum auf eine einfachere Graphensuche. Die **Zellendekomposition** quantisiert den Konfigurationsraum exakt oder approximativ in Teilstücke und führt die Suche auf Basis der Beziehungen zwischen den Teistücken aus. **Potentialfelder** basieren auf künstlichen Kräften, die von Hindernissen und dem Ziel ausgehen und abstoßend oder anziehend auf den Roboter wirken. **Zufallbasierte** Verfahren benötigen keine aufwendig berechnete Repräsentation des Freiraums sondern erzeugen Bahnen mit zufälligem Verlauf und prüfen im Nachhinein, ob die Bahn kollisionsfrei ist. CAD-Methoden verwenden die am PC erstellte geometrische Form einen Werkstücks und berechnen daraus offline über eines dieser Verfahren eine feste Trajektorie.

Bewegliche Objekte, Multirobotersysteme und mehr-segmentige Roboterarme werden in der erweiterten Problemstellung betrachtet. Um bewegliche Objekte zu modellieren wird der Konfigurationsraum um die Dimension der Zeit zu dem Konfiguration-Zeitraum erweitert. Dadurch sind die bisherigen Verfahren auch auf dynamische Umgebungen anwendbar. Für die Bahnplanung von kooperierenden Robotersystemen wie komplexe KHS-Verpackungsanlagen gibt es ein zentralisiertes und ein entkoppeltes Verfahren. Die zentralisierte Planungsmethode fasst alle Roboter zusammen und arbeitet auf einem zusammengesetzten, hoch-dimensionalen Konfigurationsraum. Das entkoppelte Verfahren plant die Bewegung jedes einzelnen Roboters

getrennt und analysiert anschließend die Bewegungen auf Kollisionen. Die Betrachtung von mehr-segmentigen Roboterarmen wirft zusätzlich die Frage auf, wie um Hindernisse herum gegriffen werden kann.

Bahnplanung ist ein PSPACE-hartes Problem. Die Berechnungskomplexität von Bahnplanungsalgorithmen liegt in $\mathcal{O}(e^m)$ mit der Dimension des Konfigurationsraums m. Also wächst die Zeitkomplexität schnell mit Dimensionen des Konfigurationsraums, daher eine die Reduktion der Komplexität notwendig. Eine Vereinfachung ist die Projektion im Konfigurationsraum, z. B. wird die Roboterhand mit drei gemeinsamen Gelenkschnittpunkten durch ein Lastobjekt ersetzt (Reduktion von drei Dimensionen) oder ein mobiler Roboter wird durch einen Kreis ersetzt (Reduktion um die Orientierung, eine Dimension). Außerdem kann die geometrische Gestalt von Objekten vereinfacht werden: Als Näherung für die konvexe Hülle der Armsegmente können Quader und Kugeln verwendet werden. Falls mit Vereinfachungen keine Bahn gefunden wird, muss der Raum zu Lasten des Rechenaufwands genauer modelliert werden. Ein arbeitsraumspezifisches Näherungsverfahren klassifiziert den Arbeitsraum nach der Anzahl der Hindernisse. Für Gebiete mit wenig Hindernissen kann ein stark vereinfachendes und schnelles Verfahren genommen werden. Andernfalls muss eine genaue und langsame Methode zur Bahnplanung benutzt werden.
In Robotern kann die Bahnplanung eine zentrale Rolle einnehmen, wenn sie mit der Antriebssteuerung für eine Echtzeitberechnung oder der Sensorik mit einem entsprechenden sensorgestützem Verfahren interagieren muss. Diese Einflüsse können bis zum dem Task-Planer reichen.

Für die folgende Betrachtung verschiedener Bahnplanungsverfahren wird ein algebraischer, polygonaler Arbeitsraum, der durch eine Menge von Geraden getrennt ist, vorausgesetzt. Der folgende Satz bietet eine einfache Möglichkeit zur Überprüfung, ob sich zwei Polygone schneiden,

Satz 1 (Überscheidung) *Zwei konvexe Polygone* ***schneiden*** *sich nicht, genau dann wenn ein Polygon komplett in der freien Halbebene einer Kante des anderen liegt.*

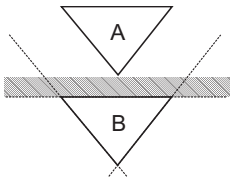


Abbildung 4.1: Polygon A liegt komplett in der freien Halbebene einer Kante von B. Somit schneiden sich A und B nicht.

Satz 1 lässt sich leicht an Abbildung 4.1 nachvollziehen. Konkave Polygone sind stets in konvexe Polygone zu zerlegen.

Im Folgenden werden die grundlegenden Verfahren zur Bahnplanung beschrieben. Neben der Laufzeitkomplexität ist die Vollständigkeit ein wichtiges Kriterium eines Verfahrens:

Definition 2 (Vollständig) *Ein Verfahren zur Bahnplanung ist **vollständig**, wenn es eine Bahn zwischen Start und Ziel liefert, falls eine Bahn existiert. Existiert keine Bahn, dann muss das Verfahren dies rückmelden.*

4.1.1 Roadmap

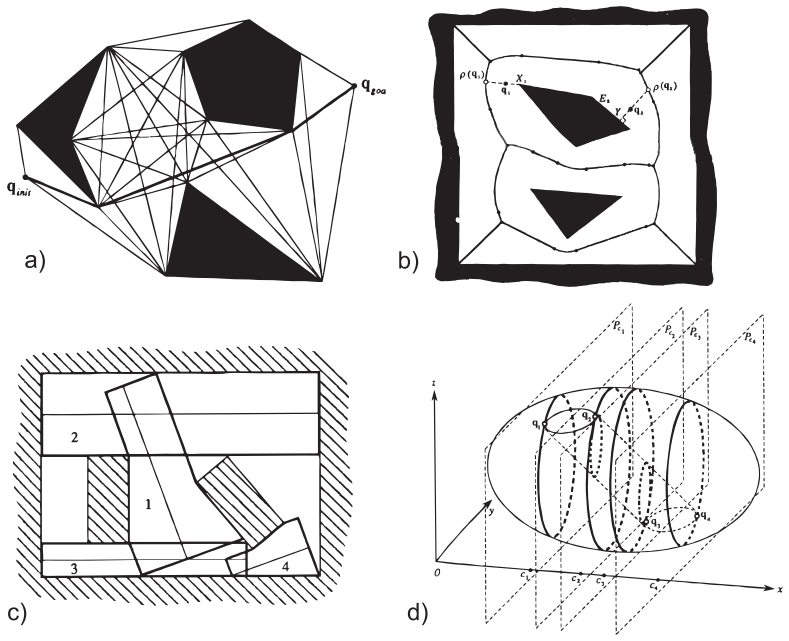


Abbildung 4.2: Bahnplanungsverfahren mit Roadmaps: Die Bahnsuche im Konfigurationsraum wird auf die Suche in einem Graphen reduziert, der mit a) Sichtgraph b) Vornoi-Diagramm c) Freeway d) Silhoutte erstellt wird, aus [Latombe 96] S. 156, 173, 177, 195

Als Roadmap-Verfahren werden in [Latombe 96] folgende Methoden aus Abbildung 4.2 zusammengefasst, die aus der geometrischen Gestalt der Hindernisse einen Verbindungsgraphen zwischen Start und Ziel erzeugen: Sichtgraph, Vornoi-Diagramm, Freeway und Silhouette.

Der Vorteil dieser Verfahren ist, dass Hindernisse exakt und analytisch beschrieben werden und ihre Repräsentation für die Implementierung wenig Speicher in Anspruch nehmen. Die Graphen können jedoch abhängig von der Hinderniskonfiguration wie in Abbildung 4.2a) dargestellt schnell komplex werden. Es ergibt sich die Frage, wie viel Speicher für die Implementierung auf einem System ohne dynamischem Speichermanagement wie einer SPS im Voraus reserviert werden muss. Wird zu viel Speicher reserviert, wird diese knappe Ressource verschwendet. Wird zu wenig Speicher reserviert, kann das Verfahren die Eigenschaft der Vollständigkeit verlieren, weil nicht alle Kanten des Graphs gespeichert werden können. Im folgenden werden die vier Roadmap-Varianten kurz beschrieben.

Für die Konstruktion des **Sichtgraphen** aus Abbildung 4.2a) werden die Polygonecken aller Hindernisse untereinander und mit der Start- und Zielkonfiguration verbunden. Schneidet eine Polygonkante eine Verbindungsstrecke, so wird sie aus

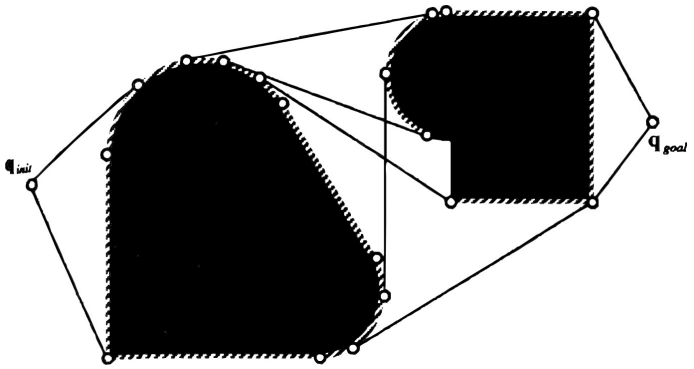


Abbildung 4.3: Sichtgraph mit weniger Kanten zwischen 2 teilweise konvexen, runden Hindernissen für weniger Rechenaufwand, aus [Latombe 96] S. 167

dem Graphen entfernt. Alle Verbindungsstrecken sind somit Sichtlinien. Anschließend wird der kürzeste Weg durch eine Graphensuche z. B. nach Dijkstra durchgeführt.
Die Anzahl der Kanten des Sichtgraphen wächst exponentiell mit der Anzahl der Polygonecken und der Dimension des Konfigurationsraums. Eine Optimierungen ist für komplexe Hindernisse notwendig. Durch die Einführung von Tangentiallinien kann die Anzahl der Linien zwischen 2 benachbarten konvexen Polygonen auf genau 4 verringert werden. Latombe [Latombe 96] stellt außerdem Verfahren für die Behandlung von konkaven und runden Strukturen vor. Die Ergebenisse sind in Abbildung 4.3 dargestellt. Die Sichtgraphenmethode liefert teilweise halbfreie Bahnen, die nahe an Hindernissen vorbeiführen.
Das **Vornoi- oder Retraktionsverfahren** in Abbildung 4.2b) liefert alle Punkte als Graphen, die von mindestens zwei Kanten oder Ecken der Hindernispolygone den gleichen, minimalen Abstand besitzen. Das Verfahren führt zu Bahnen mit dem größtmöglichen Sicherheitsabstand zu den Hindernissen. Das Vornoi-Diagramm enthält in der Regel nicht die Start- oder Zielkonfiguration. Für die Verbindung zwischen Start oder Ziel zur Bahn wird der kürzeste Weg gewählt.
In der **Freeway**-Methode aus Abbildung 4.2c) werden Zylinder zwischen begrenzende Hindernisse gelegt. Die Verbindung der Mittelachsen liefert den Graphen.
Der **Silhouetten**-Algorithmus ist vor allem bei der Bahnplanung im 3D-Raum nützlich. Das 3D-Gebilde in Abbildung 4.2d) wird in Scheiben geschnitten und es werden Extrema der Scheiben d.h. Kanten im 3D-Gebilde gesucht. Der Graph wird durch die Verbindung der Extrema erzeugt.

Neben den Speichergründen der Sichtgraphen-Methode sind die übrigen Verfahren ebenfalls nicht sinnvoll für die SPS-Bahnplanung anwendbar. Die mehrdeutige Bahn im Vernoi-Diagramm ist durch viele Punkte beschrieben, welche einen maximalen Sicherheitsabstand zu Hindernissen haben. Eine zeitoptimale Fahrt impliziert jedoch, dass die Bahn kurz ist und dicht an Hindernissen vorbeiführt. Die Freeway-Methode führt zu ähnlichen Bahnen und der Silhouette-Ansatz ist kompliziert umzusetzen.

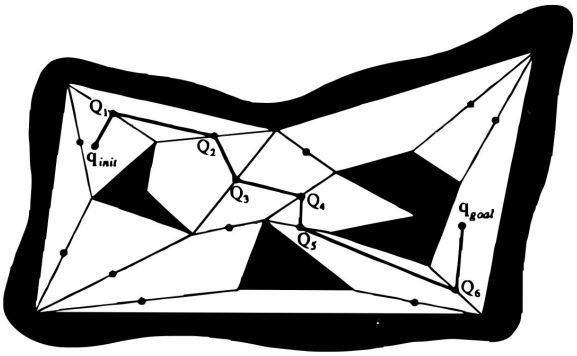


Abbildung 4.4: Bahnplanungsverfahren mit exakter Zellenzerlegung: Der Freiraum wird in konvexe Polygone aufgeteilt, die Nachbarschaftsbeziehung hergeleitet, der Nachbarschaftsgraph erstellt und darin der Weg gesucht, aus [Latombe 96] S. 205.

4.1.2 Exakte Zellenzerlegung

In Zellenzerlegungsverfahren wird der Freiraum in konvexe Zellen zerteilt. Bei exakten Verfahren ergibt die Summe der Zellen den Freiraum, bei approximierenden Verfahren eine Annäherung des Freiraums. Das Ziel der Zerlegung ist die Erzeugung von Zellen mit einer einfachen Geometrie und einer einfach ableitbaren Nachbarschaftsbeziehung. Diese Beziehung kann in einem Nachbarschaftsgraphen dargestellt werden. Eine Graphensuche führt zu der Reihenfolge der Zellen vom Start bis zum Ziel. Die Wegpunkte der Bahn werden jeweils in die Mitte der Zellenkanten gelegt. Abbildung 4.4 zeigt eine so konstruierte Bahn auf Basis von trapezoider Zellenzerlegung.
Die Line-Sweep-Variante dieses Algorithmus' nach Berns [Berns 05] führt zu einer effizienteren Implementierung der Zerlegung. Dabei werden Zellen stets mit vertikalen Kanten zerlegt. Jedoch erhöht sich dadurch die Zahl der Zellen.
Falls die Orientierung des Roboters nicht vernachlässigt werden darf, weil er z.B. in einer bestimmten Orientierung breiter ist und kollidieren kann, können kritische Kanten der Zellen eingeführt werden. Die Suche im Nachbarschaftsgraphen berücksichtigt kritische und unkritische Regionen.

Die exakte Zellenzerlegung besitzt den Vorteil der Roadmap-Methoden, aber auch deren Nachteile. Sie benötigt weniger Speicher zur Repräsentation des Freiraums und der Hindernisse. Jedoch ist die Berechnung der konkreten Gestalt der Zellen aufwändig. Zudem wählt das Verfahren den Pfad, der über die geringste Zahl an Zellen führt. Dieser kann länger und somit schlechter sein als der optimale Pfad.

4.1.3 Approximierende Zellenzerlegung

Das approximierende Verfahren zerteilt den Freiraum in Zellen mit einer einfachen und stets gleichen Gestalt. Der Freiraum kann somit nicht exakt dargestellt werden und das Verfahren ist nicht vollständig. Aber diese Methode führt zu einer simplen Berechnung, die für alle Zellen gleich angewendet wird. Durch die Diskretisierung des Konfigurationsraums wird Rundungsfehlern bei der Implementierung entgegengewirkt, die bei einer exakten Zellenzerlegung auftreten.

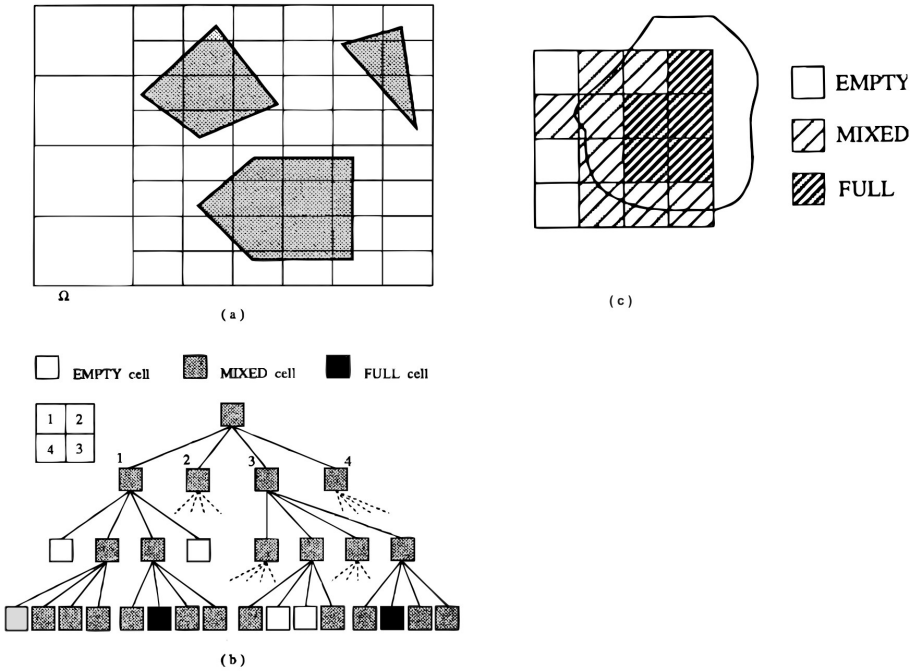


Abbildung 4.5: Bahnplanungsverfahren mit approximierender Zellenzerlegung: Der Freiraum wird in gleichartige Polygone aufgeteilt, die Eigenschaft *frei*, *belegt* oder *gemischt* den Zellen zugeordnet, gemischte werden rekuriv aufgeteilt, der Nachbarschaftsgraph erstellt und darin der Weg gesucht, aus [Latombe 96] S. 257, 270.

Verfahren mit der Diskretisierung des Arbeitsraums führen zu stets gleichem Speicheraufwand und sind geeigneter als die bisher vorgestellten Methoden für die Implementierung auf einer SPS. Der Speicheraufwand lässt sich gut im Voraus abschätzen und durch die Zellengröße beeinflussen. Im folgenden werden zwei Varianten dieser Bahnplanung kurz erläutert.

Der Algorithmus sieht folgende Schritte vor:

1. Den Freiraum in Rechteckzellen dekomponieren, siehe Abbildung 4.5a)
2. Den Zellen die Eigenschaft *frei*, *belegt* oder *gemischt* zuordnen, siehe Abbildung 4.5c)
3. Zellen und Eigenschaften in Baumstruktur ablegen, siehe Abbildung 4.5b)
4. Wenn ein freier Kanal die Start- mit der Zielzelle verbindet, dann fertig
5. Sonst gemischte Zellen weiter dekomponieren

Latombe [Latombe 96] geht auf zwei Varianten ein:

1. Die Divide and Label-Methode legt die Zellen in einer 2^m-Baumstruktur ab und markiert Zellen als frei, blockiert oder gemischt abhängig vom dem Schnitt mit den Begrenzungslinien angrenzender Hindernispolygone. Dieses Vorgehen führt zu vielen gemischten Zellen.

2. Das Approximate And Decompose-Verfahren erzeugt weniger gemischte Zellen, indem zuerst Hindernis-umhüllende oder von Hindernissen umhüllte, große Zellen erzeugt werden, die später in Normalgröße zerteilt werden.
   Um die Suche im Zerlegungsgraphen zu verbessern, kann die hierarchische Graphensuche angewandt werden. Es wird bei der weiteren Zerlegung gemischter Zellen das Ergebnis der Vorarbeit, also die vorherige Suche, miteinbezogen.

Hirzinger und Ralli [Hirzinger 97] verwenden selbstorganisierende Kohonen-Netze für die effiziente, diskrete Modellierung eines hochdimensionalen, hochauflösenden Konfigurationsraums $\mathbb{C}$. Es wird nach der Freiheit von Kollisionen im Raum unterschieden, so dass kollisionsfreier Raum hochauflösend und kollisionsbehafteter Raum niedrig aufgelöst und der gesamte Zustandsraum speichereffizient dargestellt werden kann. Ausgehend von einer äquidistanten Diskretisierung von $\mathbb{C}$ wird das Diskretisierungs-Raster als elastisches Netz lokal so verbogen, dass kollidierende Konfigurationen niedrig aufgelöst werden und die benachbarten kollisionsfreien höher. Durch diese Reorganisation wird die Zahl der kollisionsfreien Konfigurationen ohne zusätzlichen Speicherbedarf erhöht und es können schmale Wege zwischen Hindernissen gefunden werden, die zuvor wegen des zu groben Rasters nicht vorhanden waren. Hirzinger und Ralli [Hirzinger 97] weißen durch Versuche eine bis zu 40-fach höhere Auflösung bis zu $0{,}1^{\circ}$ von $\mathbb{C}$ in der Nähe von Hindernissen mit Kohonen-Netze im Vergleich zu äquidistante Netze mit der gleichen Speichergröße nach. Zudem wird die Bahn glatter.
Nachteilig an diesem Verfahren ist die erhöhte Laufzeitkomplexität und ein im Vergleich zu einer äquidistanten Zerlegung mit gleicher Zellenzahl erhöhter Speicherbedarf, da die Position einer verschobenen Zelle explitzit als n-Tupel gespeichert wird.

Aufgrund der Speicherbeschränkung der SPS ist dieses Verfahren sinnvoll, wobei empirisch ein Kompromiss zwischen Auflösung des Netzes, der notwendigen Speichergröße und der Ausführungseffizienz erzielt werden muss.

4.1.4 Potentialfelder

Die Potentialfeldmethoden unterscheiden sich grundsätzlich von Roadmap- und Dekompositionsverfahren. Im Potentialfeld wird der Roboter als ein Partikel unter dem Einfluss eines künstlichen Kräftefelds aufgefasst. Das Ziel besitzt ein niedriges Potential und übt eine anziehende Kraft aus, Hindernisse ein hohes Potential und abstoßende Kräfte. Die Kraft berechnet sich aus dem negativen Gradienten des Potentialfelds und gibt die Bahn vor.
Potentialfeldmethoden sind für schnelle, sensorbasierte, lokale online Bahnplanung mit unvollständigem Weltbild geeignet. Sie sind auch mit Graphensuchverfahren kombinierbar, so dass sie für globale Bahnplanung offline verwendet werden können.

Die geringe Anforderungen an Rechenleistung macht die Potentialfeldmethoden auch für die Umsetzung auf der Industrierobotersteuerung interessant. Da der Verlauf der kompletten Bahn bei Beginn der Fahrt nicht bekannt sein muss, sind die Speicheranfoderungen ebenfalls gering. Potentialfelder weißen jedoch einige Probleme auf, die im folgenden beschrieben werden.

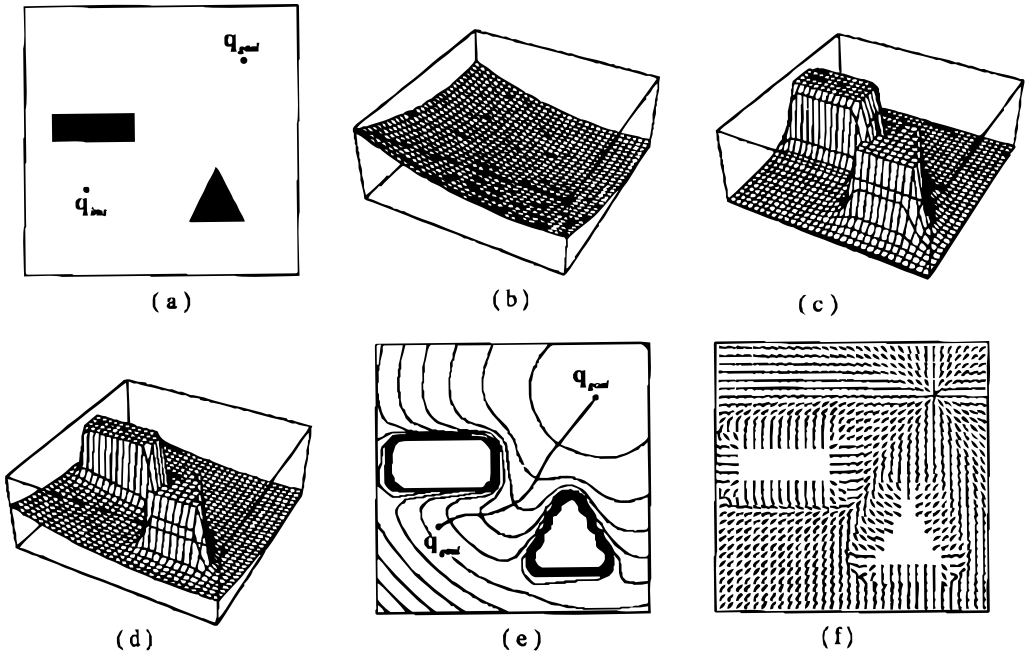


Abbildung 4.6: Bahnplanungsverfahren mit Potentialfeld: (a) Topologische Karte mit Start, Ziel und Hindernissen (b) anziehendes Potentialfeld des Ziels (c) abstoßendes Feld der Hindernisse (d) Superposition beider Felder (e) Äquipotentiallinien mit Bahn über den negativen Gradienten (f) Orientierungen der negativen Gradienten, aus [Latombe 96] S. 302

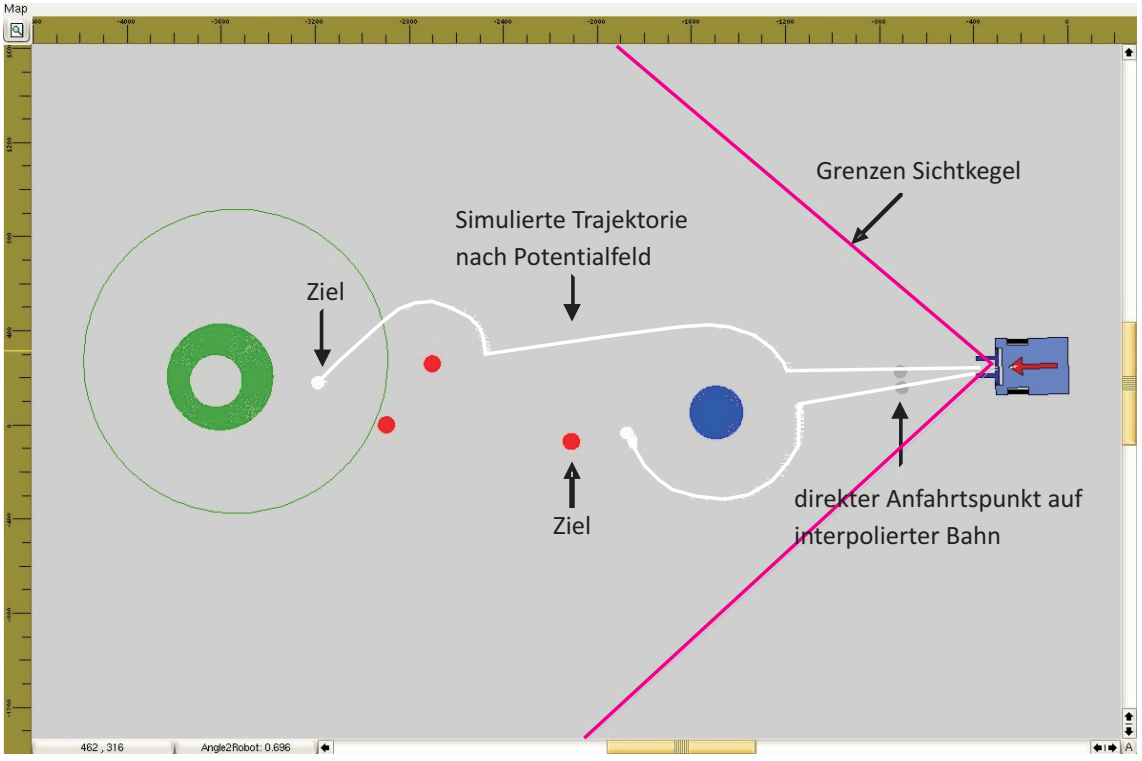


Abbildung 4.7: Bahnplanung eines autonomen Gabelstaplers mit Potentialfeldmethode mit großer Bahnkrümmung vor Hindernissen, aus [Kretschmann 06c] S. 75

Die Bahn weist direkt vor Hindernissen eine starke Krümmung auf, so dass die Geschwindigkeit reduziert werden muss. In Abbildung 4.7 sind solche Bahnen gezeigt. In [Kretschmann 06c] wird dieses Problem umgangen, indem ein kurzer Teil der Bahn simuliert und ein Anfahrtspunkt mit festem Abstand zum Roboter auf der simulierten Bahn berechnet wird. Dieser Punkt wird als direktes Ziel angefahren. Hindernissen kann mit diesem Verfahren zwar weich ausgewichen werden, jedoch können Kollisionen auftreten, wenn sich der Roboter neben dem Hindernis befindet.

Lokale Minima im Feld führen zu Problemen und sind der Hauptnachteil Dieses Verfahrens. Es ist deshalb nicht vollständig. Als Lösung müssen Minima-vermeidende Potentialfunktionen und Minima-Escape-Stategien aufgestellt werden.

Die Kraft auf den Roboter im Potentialfeld berechnet sich wie folgt:

1. Der Betrag der anziehenden Kraft berechnet sich parabolisch oder konstant zum Zielabstand.
2. Die abstoßende Kraft ist eine Hyperbelfunktion zum Hindernisabstand oder 0, falls das Hindernis zu weit entfernt ist.
3. Die vektorielle Addition aller Kräfte an der Roboterposition liefert die Bahnrichtung.

Abbildung 4.6 zeigt die einzelnen Schritte bei der Erstellung der Bahn mit Potentialfeldern. Das hohe Potential der Hindernisse wird mit dem des Starts überlagert. Die Überlagerung aller Potentiale führt zum resultierenden Feld. Der Roboter kann sich nun nun entlang des negativen Potentialgradienten vom Start zum Ziel kollisionsfrei bewegen, so wie Wasser von der Spitze eines Bergs um Hügel herum zum Tal fließt.

Die Kräfte können mit Koeffizienten gewichtet werden. Komplexe Hindernisse werden konvex zerteilt und die abstoßenden Kräfte entsprechend ihrer Größe gewichtet. Liegt das Ziel nahe an Hindernissen, führt das zu Problemen. Als Lösung muss der Kraftkoeffizient für das Ziel lokal erhöht werden. Ebenso kann gefährlicheren Hindernissen erhöhte Koeffizienten zugeordnet werden.

Problem des Potentialfelds: Lokale Minima

Zum Auffinden und zur Bekämpfung der lokalen Minima kann das komplette Potentialfeld im vor der Fahrt mit Funktionalen abgesucht werden. Latombe [Latombe 96] zeigt zwei Alternativen dazu auf:
Im Depth-First-Verfahren wird die komplette Bewegung in einzelne Bahnsegmente diskretisiert und die Verkettung der Segmente liefert die Bahn. Die Methode kann dennoch in lokalen Minima enden. Dies wird erkannt, falls die Bewegung in kleinen Kreisen endet. Ein Entkommen gestaltet sich aber als schwierig.
Der Best-First-Algorithmus legt eine Rasterkarte über das Feld. In einer Open-Close-Liste wird während der Fahrt vermerkt, welche Rasterzellen schon besucht wurden. Es wird immer die Nachbarzelle angefahren, die das niedrigste Potential besitzt. Dies

führt zu einem Ausweg aus lokalen Minima.

Die schwierige Suche nach lokalen Minima und Escape-Strategien wird durch Minimavermeidende Methoden überflüssig. Eine solche Methode ist die numerische Navigation mit Wellenfrontexpansion.
Der Arbeitsraum wird in Zellen diskretisiert. Das Ziel erzeugt eine Welle, die sich fortpflanzt und den Freiraum überschwemmt. Das Potential einer Zelle $P(z_i)$ berechnet sich aus der Dauer, wie lange die Wellenfront von ihrer Erzeugung bis zu z_i unterwegs war. Da die Welle Hindernisse hinterspült und sich somit ein höheres Potential zwischen Roboter und Hindernis ergibt, können sich keine lokalen Minima bilden. Das Potential einer Zelle entspricht der Entfernung zum Ziel um Hindernisse herum. Es entsteht eine Bahn entlang des Grandienten, die nahe an Hindernissen vorbeiführt.
Um den Sicherheitsabstand der Bahn zu Hindernissen zu maximieren, kann dieses Verfahren mit Roadmap-Methoden kombiniert werden: Auf Basis der topologischen Karte wird ein Vornoi-Diagramm erstellt. Dieses liefert mehrere Bahnen mit maximalen Sicherheitsabstand vom Start zum Ziel. Anschließend wird die Wellenfrontexpansion vom Ziel aus durchgeführt. Die Überlagerung des Potentialfelds mit dem Vornoi-Diagramm liefert eine eindeutige Bahn in dem Diagramm. Latombe [Latombe 96] zeigt Varianten der Wellenfront-Methode für höhere Dimensionen auf, um Laufzeit- und Speichereffizienz zu verbessern.
Latombe [Latombe 96] beschreibt zudem Möglichkeiten, wie die Auswirkungen von Minima verringert werden können. Es wird ein elliptisches Potential, dessen räumliche Ausdehnungen parametriert ist, um Hindernisse gelegt. Nach der Berechnung der Minimafunktion wird über die Parameter das elliptische Potential so verändert, dass die Minima verschwinden oder ihre Größe verkleinert wird. Dieser Ansatz ist wegen des Berechnungsaufwands nur für einen Konfigurationsraum geringer Dimensionen geeignet.
Die Alternative ist ein Best-First-Algorithmus, der den Ausweg aus Minima durch die Verkettung zufälliger Bewegungen im Freiraum findet. Der Pfad durch die einzelnen Minima wird in einem Graphen abgelegt und geglättet. Der Nachteil dieses Verfahrens ist die fehlende Reproduzierbarkeit der Bahn, da es auf Zufall basiert. Außerdem terminiert es nicht, falls keine Bahn existiert. Falls nach endlicher Laufzeit keine Bahn gefunden wird, bleibt die Frage offen, ob bei längerer Laufzeit doch eine Bahn gefunden würde.

Die vorgestellte Potentialfeldmethode ist trotz ihrer geringen Anforderungen an Rechenleistung und Speicher wegen des Problems der lokalen Minima, die nur mit komplexen oder zufallsbasierten Verfahren zu umgehen sind, nicht für den SPS-Einsatz geeignet.

4.1.5 Zufallsverfahren

Der Nachteil bisher vorgestellter Verfahren liegt in der aufwendigen Berechnung des Freiraums und des Einfluss' der Hindernisse auf die Bahn, vor allem bei hohen Dimensionen (>3) des Konfigurationsraums. Eine Zufallsplanung erzeugt per Zufallsgenerator neue Zwischenstützpunkte und prüft im Nachhinein, ob der Punkt in

einem Hindernis liegt und ob die Verbindung der vorhandenen Punkte mit dem neuen Zwischenpunkt zu Kollisionen führt. Gibt es eine kollisionsfreie Verbindung, so wird der neue Punkt in das Netz der vorhandenen Punkte eingefügt. Auf Basis dieses Netzes ist eine Graphensuche nach dem besten Weg möglich. Der Planungsaufwand ist unabhängig von der Gestalt des Konfigurationsraums konstant. Der Nachteil ist die fehlende Vollständigkeit des Berechnungsverfahrens und die fehlende Reproduzierbarkeit der Bahn. Die Konvergenzrate von Zufallsverfahren ist Gegenstand der Forschung.

Hsu, Kindel und Latombe [Hsu 00] betrachten eine zufallsbasierte Bahnplanung im Konfiguration-Zeitraum unter Beachtung der Randbedingungen der Kinematik und Dynamik eines nicht-holonomen mobilen Roboters, indem mit der Konfiguration nicht nur die Lage des Roboters sondern auch seine Geschwindigkeit ausgedrückt wird. Es wird der Begriff der wahrscheinlichen Vollständigkeit angewandt. Ein Verfahren ist wahrscheinlich-vollständig, wenn die Wahrscheinlichkeit, dass das Verfahren das Gesuchte liefert, bei steigender Laufzeit gegen 1 konvergiert. Dies sagt jedoch nichts über die Konvergenzrate aus. Das Verfahren diskretisiert den Konfiguration-Zeitraum mit zufallsbasiertem Sampeln anstatt mit einem regelmäßigen Netz wie in der approximativen Zellenzerlegung. Das Sampeln basiert auf der zufälligen Auswahl von Fahrbefehlen und der anschließenden Berechnung der Folgekonfiguration über zeitliche Integration. Liegt die Folgekonfiguration in einem Hindernis, wird sie verworfen und neu gesampelt. Somit entsteht eine Baumstruktur mit gültigen Folgekonfigurationen (Milestones) bis das Ziel erreicht wurde. Die Wahl neuer Sampels geschieht abhängig von der Dichte der Sampels in der unmittelbaren Umgebung, um die Konvergenzrate zu erhöhen. Die Bahn zum Ziel erfüllt automatisch alle kinematischen und dynamischen Bedingungen, da auf Basis von Fahrbefehlen gesampelt wurde und über die direkte Kinematik/ Dynamik die Folgekonfiguration berechnet wurde.
Angewandt auf Industrieroboter bedeutet die Methode aus [Hsu 00] eine zufallsbasierte Drehmomentsteuerung. In dem um Gelenkgeschwindigkeiten und -beschleunigungen erweiterten Konfigurationsraum werden diese Gelenkgrößen über zeitliche Integration der direkten Dynamik ermittelt. Die Herleitung der inversen Dynamik ist nicht notwendig.
Hernando und Gambao [Hernando 02] stellen eine Sichtbarkeitsanalyse auf Basis von Sichtbarkeits-Tetraeder und genetischen Algortihmen (GA) vor. Im Gegensatz zu Verfahren, die einen globalen Planer zur Erstellung und Optimierung von zufälligen Stützpunkten und einen lokalen Planer zur Überprüfung der Erreichbarkeit zweier Punkte verwenden, basiert diese Methoden nur auf einem lokalen Planer. Der GA optimiert abhängig von der Distanz Zwischenpunkte oder -ziele, die mit dem lokalen Planer verbunden werden. Dieser approximiert den von der Hand in der Bewegung überstrichenen Raum mit Tetraeder. Dieses GA-Verfahren besitzt viele Parameter, z. B. zur Ableitung der neuen Generation, die empirisch zu ermitteln sind.

Zufallsverfahren basieren auf einfachen Algorithmen, die schnell zu guten Lösungen finden und leicht nachzuvollziehen sind. Allerdings werden sie der Forderung der Reproduzierbarkeit nicht gerecht. In einer industriellen Umgebung darf die Gestalt der Bahn nicht vom Zufall abhängen sondern sollte deterministisch sein.

4.1.6 Multi-Robotersysteme, Mehrkörpersysteme

In Multi-Robotersystemen wie KHS Palettieranlagen agieren mehrere Roboter in einem gemeinsamen Arbeitsraum. Die Bahnplanung solcher Systeme setzt ein dynamisches Modell voraus und kann in dem Konfiguration-Zeitraum gelöst werden. Latombe [Latombe 96] geht auf die zentralisierte und die entkoppelte Planung ein. Der Nachteil zentralisierter Verfahren ist eine hohe Dimension des Konfiguration-Zeitraums. Entkoppelte Methoden hingegen sind weniger vollständig.
Die Bahn im Konfiguration-Zeitraum muss zeitlich streng monoton sein. Verfahren im Konfiguration-Zeitraum liefern halbfreie Bahnen für bewegliche Hindernisse. Es ist eine exakte Zellendekomposition anwendbar, wobei die erste Koordinate die Zeit darstellt, die übrigen Koordinaten repräsentieren den gewohnten Konfigurationsraum. Die Analyse des Verbindungsgraphen führt zur Bahn. Dieses Verfahren ist ineffizient. Die approximierende Zellenzerlegung ist effizienter, aber gewährleistet nicht die Monotonie der Zeit. Deswegen muss die Dauer der Freizellen berechnet und mit berücksichtigt werden. Eine Berücksichtigung der maximalen Robotergeschwindigkeit geht in eine angepasste Sichtgraphenmethode mit ein. Als Bedingung müssen alle Kanten in Richtung der Zeitachse gerichtet sein. Außerdem dürfen keine Kanten senkrecht zur Zeitachse verlaufen, denn dies würde zu einer unendlich hohen Geschwindigkeit führen.
Kant und Zucker [Zucker 86] verfolgen einen 2-stufigen Ansatz für Umgebungen mit dynamischen Hindernissen. Zuerst werden die dynamischen Objekte vernachlässigt und eine kollisionsfreie Bahn um die statischen Objekte erstellt. Danach wird die Geschwindigkeit auf der gegebenen Bahn so bestimmt, dass keine Kollision mit dynamischen Hindernissen auftritt. Dieses Verfahren ist nicht vollständig. Die Vollständigkeit wird verbessert, wenn im ersten Schritt ein Netz aus mehreren Bahnen erstellt wird und im zweiten Schritt ausgewählt werden kann.

Für mehrere Roboter mit einem zusammengesetzten Konfigurationsraum und zentralisierter Planung kann nach Latombe [Latombe 96] eine Potentialfeldmethode verwendet werden. Ein lokales Minimum liegt dort vor, wo die Roboter verklemmt sind. Die Auswege aus Minima sind simulierte und vorausberechnete Bahnen oder eine zufällige Fahrt. Dieses zentralisierte Verfahren ist zwar vollständig aber zeitaufwändig. Die schnellere, entkoppelte Planung kann prioritätsbasiert mit Zeitscheiben erfolgen, wobei die Planung nur im Konfigurationsraum stattfindet. Eine andere Variante greift Mechanismen aus dem Bereich der Betriebssysteme auf und führt die Bahnkoordinierung mit Hilfe eines Schedulings aus. Die exklusiv nutzbare Ressource des Scheduling-Verfahrens ist hier der gemeinsam verwendete Raum. Für jeden Roboter wird eine Bahn ungeachtet des anderen geplant. Anschließend werden die Bahnen über den Scheduler kombiniert. Der Algorithmus läuft in folgenden Schritten ab:

1. Die Bahn jedes Roboters A_i über seine gefahrene Stecke s_i parametrieren

2. Die (s_1, s_2)-Ebene in Zellen diskretisieren und die Kollisionszellen bestimmen

3. Einen Weg in der (s_1, s_2)-Ebene um die Kollisionszellen herum suchen, um das Ziel $(s_1, s_2) = (0, 0)$ zu erreichen

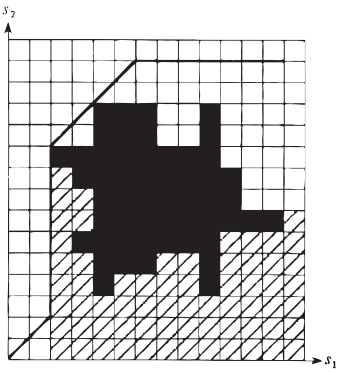


Abbildung 4.8: Die (s_1, s_2)-Ebene mit Kollisionszellen 2er Roboter und den koordinierten Bahnen, aus [Latombe 96] S. 382

Das Ergebnis einer Bahnplanung ist in Abbildung 4.8 gezeigt. Die Bahnkoordinierung ist nicht vollständig und weniger vollständig als die prioritätsbasierte Planung.

Für die detailliertere Robotermodellierung mit mehreren Armsegmenten wird ein größerer Konfigurationsraum verwendet. Hier können die bisher beschriebenen klassischen Methoden angewendet werden.

Im Rahmen dieser Arbeit soll zunächst nur die Bahnplanung eines separierten Roboters durchgeführt werden. Da die in Kapitel 2 vorgestellten Verpackungsanlagen dennoch Multi-Robotersysteme sind, muss die gewählte Bahnplanung ggf. im Rahmen einer entkoppelten Planung des Gesamtsystems eingesetzt werden können. Dazu ist erneut die Laufzeit- und Speichereffizienz der Planung von hoher Bedeutung.

4.1.7 Probleme der vorgestellten Verfahren

Die bisherigen Methoden sind für den Einsatz auf der Steuerung der KHS-Roboter nur beschränkt anwendbar. Exakte Lösungen des Bahnplanungsproblems benötigen viel Rechen- und dynamische Speicherressourcen, die in der SPS knapp sind. Approximierende Bahnplanungen benötigen ebenfalls viel aber konstanten Speicher, um den Konfigurationsraum zu repräsentieren, und sind weniger vollständig. Potentialfeldmethoden arbeiten einfach und schnell, können aber leicht zu Verklemmungen führen. Ein Zufallsverfahren ist nicht deterministisch. Ein Verfahren nach CAD mit festen Bahnen ist zu unflexibel und für eine online-Bahnplanung nicht geeignet.

Die in [Latombe 96], [Hirzinger 97] vorgeschlagenen Verbesserung zur Umgehung lokaler Minima und Reduktion des Speicheraufwands scheinen wegen des zusätzlichen Aufwands für den Einsatz auf der SPS nur bedingt sinnvoll.
Für die offline Planung ist ein Zufallsverfahren anwendbar, da beliebig viel Rechenzeit zur Verfügung steht und es auf beliebig komplexe Hindernisse und Ordnungen von $\mathbb{C}$ anwendbar ist.

Im folgenden Abschnitt wird auf die Wellenfrontexpansion für eine online-Planung näher eingegangen und die Defizite durch Erweiterungen beseitigt.

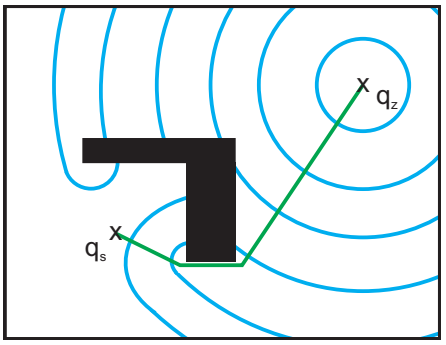


Abbildung 4.9: Bei der Wellenfrontexpansion werden Hindernisse „hinterspült“ und lokale Minima vermieden. Die Bahn liegt auf den Gradienten .

4.1.8 Auswahl und Verfeinerung der Wellenfrontexpansion

Die Wellenfrontexpansion wie von Latombe [Latombe 96] skizziert aus Abbildung 4.9 erfüllt die Anforderung des konstanten Speicherbedarfs in der SPS, der Vollständigkeit und der einfachen Realisierbarkeit. Sie liefert eine Folge von kollisionsfreien Konfigurationen von der Startkonfiguration zum Ziel. Bei einer feinen Diskretisierung von $\mathbb{C}$ werden viele Konfigurationen, die auf geraden Segmenten liegen, als Stützpunkte für die Bahninterpolation erzeugt. Die Lage dieser Konfigurationen beinhalten keine zusätzlichen Informationen. Um die Zahl der Stützpunkte auf eine notwendige Zahl für eine effiziente Speicherung zu reduzieren, wird die zweite Ableitung des Pfads betrachtet. Alle Konfigurationen $\vec{c} \in \mathbb{C}_{pfad}$ mit $\vec{c}'' = 0$ werden verworfen. Somit bleiben nur Eckpunkte als Stützpunkte übrig. Zwischen den Eckpunkten wird die Bahn linear interpoliert.

Algorithmus der Bahnplanung

Der Algorithmus dieses Verfahrens basiert auf den folgenden Makro-Schritten:

1. Beschreibung der Hindernisse in $\mathbb{C}$ transformieren
2. Ziellage des Roboters in die Zielkonfiguration transformieren und über die Wellenfront ausgehend von der Ziel-Konfiguration allen freien Konfigurationen ein Potential zuordnen
3. Startlage des Roboters in die Startkonfiguration transformieren und entlang des negativen Gradienten des Potentials den Leitpfad $\mathbb{C}_{pfad}$ zum Ziel in $\mathbb{C}$ markieren
4. Alle Knicke im Leitpfad mit $\vec{c}'' \neq 0$ in Gelenkwinkelraum abbilden und als Stützpunkte übernehmen

Abbildung 4.10 visualisiert die Ergebnisse dieser Schritte im 2-dimensionalen Raum. An dem Leitpfad ist die Quantisierung von $\mathbb{C}$ erkennbar. Hier wurde der Gelenkwinkelraum in 100×100 Zellen zerteilt.
Dieses Verfahren ist nur dann vollständig, wenn Hindernisse mindestens in der Größenordnung der Zellgröße liegen, z. B. müssen Hindernisse mindestens 3 cm groß sein, wenn der Fahrweg des Hubgelenks des RS3 3 m beträgt.

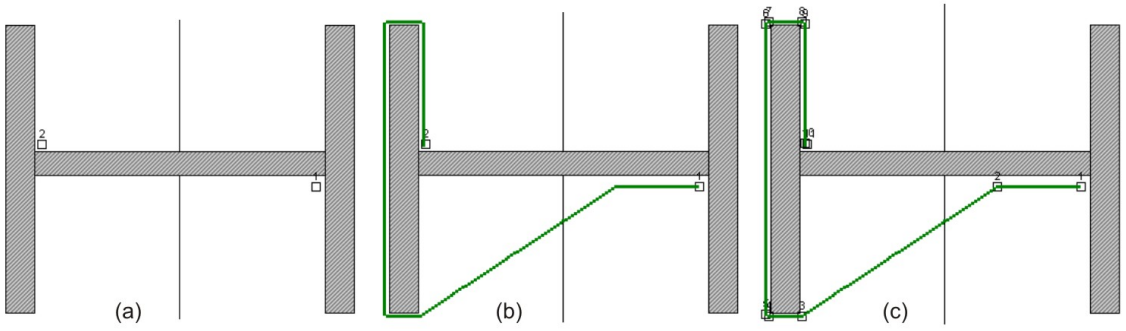


Abbildung 4.10: Schritte der Bahnplanung: a) Repräsentation der Hindernisse, Start und Ziel in $\mathbb{C}$, b) Leitpfad entlang des Potential-Gradienten, c) Stützpunkte in den Ecken des Leitpfads (aus Simulation)

Abschätzung der Laufzeitkomplexität

Zur Approximation der Ausfürungszeit des Algorithmus' werden die Schritte im Detail betrachtet.

1. Sei $n_{Hindernis}$ die Anzahl der Hindernisse, m die Dimension des Gelenkwinkelraums $\mathbb{G}$ und die des Konfigurationsraums $\mathbb{C}$ und c die Anzahl der Konfigurationen pro Dimension. Hindernisse lassen sich durch $2m$ $(m-1)$-dimensionale Grenzelemente im m-dimensionalen Raum beschreiben. Ferner sei $f_{\mathbb{G},\mathbb{C}} : \mathbb{G} \rightarrow \mathbb{C}$ die lineare Abbildung einer Gelenkstellung auf die zugehörige Konfiguration, $f_{Hindernis} : \mathbb{G}^{2m} \rightarrow \mathbb{C}^{2m}$ die Abbildung eines Hindernisses mit seinen $2m$ Begrenzungen in den Konfigurationsraum und $f_{Hindernisse} : \mathbb{G}^{2m\,n_{Hindernis}} \rightarrow \mathbb{C}^{2m\,n_{Hindernis}}$ die Abbildung aller Hindernisse.
   Es gilt: $f_{\mathbb{G},\mathbb{C}} \in \mathcal{O}(1)$, $f_{Hindernis} \in \mathcal{O}(c^m) \Rightarrow f_{Hindernisse} \in \mathcal{O}(n_{Hindernis}\, c^m)$.

2. Über $f_{\mathbb{G},\mathbb{C}}$ wird zunächst das Ziel in $\mathbb{C}$ abgebildet und dieser Konfiguration das Potential $p_{Ziel} = 0$ zugewiesen. Anschließend initialisiert $f_{init} \in \mathcal{O}(c^m)$ alle Konfigurationen mit einem hohen Startpotential. $f_{pot} \in \mathcal{O}(c^{2m})$ iteriert über alle Konfigurationen und weißt ihnen ausgehend von dem Potential der Nachbarkonfiguration p_i das nächst höhere Potential $p_i + 1$ zu. Dies wird wiederholt, bis keine Konfiguration ohne Potentialzuweisung vorhanden sind.
   Die Komplexität dieses Schrittes liegt in $\mathcal{O}(1) + \mathcal{O}(c^m) + \mathcal{O}(c^{2m}) = \mathcal{O}(c^{2m})$. Ein Verfahren mit der minimalen Laufzeit $\mathcal{O}(c^m)$ iteriert nur einmal über alle Konfigurationen und verwaltet abgearbeitete Konfigurationen mit Listen oder ähnlichen dynamischen Programmierkonstrukten. Dies ist in der SPS mit der statischen Speicherverwaltung nicht möglich.

3. $f_{\mathbb{G},\mathbb{C}}$ bildet den Start nach $\mathbb{C}$ ab. $f_{Markierung}$ markiert Konfigurationen, die sich auf dem negativen Potential-Gradienten vom Start zum Ziel befinden. Dazu muss im Worst-Case über alle Konfigurationen iteriert werden und für jede Konfiguration die Nachbarzelle mit dem niedrigsten Potential gesucht werden. Im 1-dimensionalen Raum existieren 2 Nachbarn, in der 2-dimensionalen Ebene 8 Nachbarzellen, im 3-dimensionalen Raum 26 Nachbarn, im m-dimensionalen Raum $k_m = 3\,k_{m-1} + 2$ Nachbarn mit $k_1 = 2$. Die Zahl der Nachbarn wächst

exponentiell. Die Überprüfung der Nachbarn liegt in $\mathcal{O}(x^m)$ mit $x \in \mathbb{R}^+$.
Schritt 3 liegt in $\mathcal{O}(1) + \mathcal{O}(c^m) + \mathcal{O}(x^m) = \mathcal{O}(c^m)$.

4. Über $f^{-1}_{\mathbb{G},\mathbb{C}}$ werden alle Bahn-Konfigurationen in Gelenkstellungen während Schritt 3 rücktransformiert. Nach dieser Rücktransformation $\in \mathcal{O}(1)$ wird die zweite Ableitung der maximal c^m Zwischenstützpunkte numerisch berechnet. Dies geschieht in 2 Iterationen über alle Punkte, wobei während der zweiten Differentiation alle Stützpunkte mit $\vec{q} = 0$ verworfen werden. Zur Erhöhung der Zielgenauigkeit wird die Ziellage aus Schritt 2 der Liste der Zwischenpunkte hinzugefügt.
Der Aufwand von Schritt 4 liegt in $\mathcal{O}(c^m) + \mathcal{O}(2\,c^m) = \mathcal{O}(c^m)$.

Die Laufzeit des gesamten Algorithmus' liegt in $\mathcal{O}(c^{2m})$. Ein optimiertes Verfahren, das eine auf der SPS nicht vorhandene dynamische Speicherverwaltung bedingt, liegt in $\mathcal{O}(c^m)$. Es muss Speicher für c^m Konfigurationen und maximal c^m Zwischenstützpunkte bereitgestellt werden.

4.2 Bahninterpolation

Die Bahnplanung aus dem vorherigen Abschnitt liefert eine Folge von Stützpunkten, um den Roboter kollisionsfrei von der Startkonfiguration in die Zielkonfiguration zu bewegen. Die Aufgabe der Bahninterpolation liegt nun in der Generierung einer kontinuierlichen Bahn, die durch die gegebenen Stützpunkte verläuft, so dass der Antriebssteuerung im Raster des Aktualisierungsintervalls neue Werte für die Gelenkstellungen übergeben werden können. Die Art des Interpolationsverfahrens wirkt sich sowohl auf die Gestalt der Bahn als auch auf den Rechenaufwand aus.

Für ein zeitoptimales Fahren auf einer Trajektorie werden die geometrische Raumkurve und die Zeitprofile getrennt betrachtet. Somit ist eine separierte Optimierung möglich. An [Olomski 89] angelehnt wird die Bahn parametriert. Der Bahnparameter s ist z. B. die Bogenlänge der kartesischen Bahn vom Start bis zum Zeitpunkt t,

$$s(t) = \int_{t_0}^{t} \sqrt{\left(\frac{dx(t)}{dt}\right)^2 + \left(\frac{dy(t)}{dt}\right)^2 + \left(\frac{dz(t)}{dt}\right)^2}\, dt\,. \tag{4.1}$$

Somit berechnen sich die Bahngeschwindigkeit und -beschleunigung zu

$$\begin{aligned} \vec{X} &= \vec{X}(s) \\ \vec{v} &= \dot{\vec{X}} = \vec{X}' \cdot \dot{s} \\ \vec{a} &= \ddot{\vec{X}} = \vec{X}'' \cdot \dot{s}^2 + \vec{X}' \cdot \ddot{s} \\ &\quad \text{mit } (\,)' = \frac{\partial}{\partial s}\,. \end{aligned} \tag{4.2}$$

Bei elastizitätsbehafteten Robotern ist eine stetige Beschleunigung zur Vermeidung von Schwingungen nötig. Die stetige Beschleunigung führt zu einer Bewegung mit einem beschränktem Ruck. Die Ruckbegrenzung muss sich auch in der geometrischen

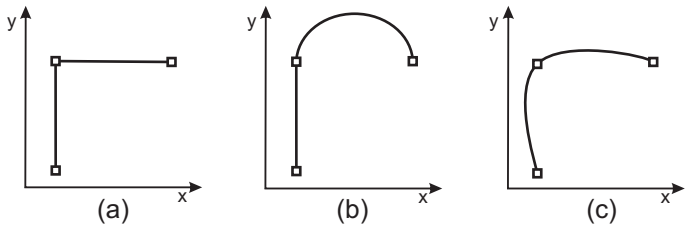


Abbildung 4.11: Interpolationstypen: (a) linear, (b) zirkular, (c) polynomial

Raumkurve wiederfinden: Es dürfen keine Knicke oder unstetige Krümmungsänderungen in der Bahn vorkommen. Die Bahn sollte einen glatten Verlauf besitzen.

Zunächst werden die grundlegenden Interpolationstypen zur Generierung der Bahn vorgestellt. Danach wird auf kubische Splines näher eingegangen.

4.2.1 Linearinterpolation, Bézier- und B-Splines

Ein sinnvoller Interpolationsalgorithmus muss eine Sollbahn liefern, die stetig ist. Doch dieses Kriterium ist für ein zeitoptimales Fahren nicht ausreichend. Die Bahn sollte zusätzlich glatt sein, also 2-mal stetig differenzierbar nach dem Ort (C_2-stetig). Der einfachste Algorithmus, die Linearinterpolation zwischen zwei aufeinander folgenden Stützpunkten aus Abbildung 4.11a), weist die geforderte Stetigkeit zwischen den Stützpunkten vor. Allerdings tritt eine Unstetigkeit bereits bei der ersten Ableitung an den Stützstellen bei den Übergängen auf.
Die zirkulare Interpolation in Abbildung 4.11(b) erfüllt die Forderung der stetigen Differenzierbarkeit nur höchstens bis zur ersten Ableitung in den Übergängen.
Höhergradige Polynome wie in Abbildung 4.11(c) könen stetige Bahnen mit beliebig vielen Stützstellen erzeugen. Ein Polynom mit Grad $n-1$ erlaubt die Interpolation zwischen n Stützstellen und ist stetig bis zur $(n-2)$-ten Ableitung. Allerdings zeigt ein Polynom Oszillationen abhängig von Zahl und der Lage der Stützstellen wie in Abbildung 4.12 dargestellt. Schäfer [Schäfer 06] stellt polynomiale Interpolation nach einem Standard-Verfahren mit Matrizen, nach Newton, Lagrange und Bézier gegenüber. Während das Standardverfahren wegen der Matrixinversion mit Aufwand von $\mathcal{O}(n^3)$ die schlechteste Alternative darstellt, müssen bei dem effizienteren Newton-Verfahren alle Basisfunktionen neu berechnet werden, wenn ein neuer Stützpunkt hinzukommt. Dies ist bei Lagrange nicht der Fall. Es zeigt sich jedoch bei den drei Verfahren eine Oszillation der Kurve bei steigender Zahl an Stützpunkten. Bézier-Kurven weisen keine Oszillationen vor, jedoch beeinflusst die Lage eines Stützpunktes ebenfalls den Verlauf der kompletten Kurve.
Spline-Funktionen stellen einen Kompromiss zwischen der Einfachheit der Funktionen und der Forderung nach der stetigen Differenzierbarkeit dar. Eine Spline-Interpolation über n Stützstellen besteht aus der Menge von $n-1$ Interpolationspolynomen, die jeweils in einem Stützpunktintervall gültig sind und die Polynom-Koeffizienten sind so gewählt, dass an den Übergängen die 2-fach stetige Differenzierbarkeit gewährleistet ist. Bei Splines wird das Problem der Oszillation umgangen, da ein Interpolationspolynom auf jeweils zwei Stützstellen basiert.
Schäfer [Schäfer 06] behandelt Bahnplanung und speziell die Interpolation. Nach der

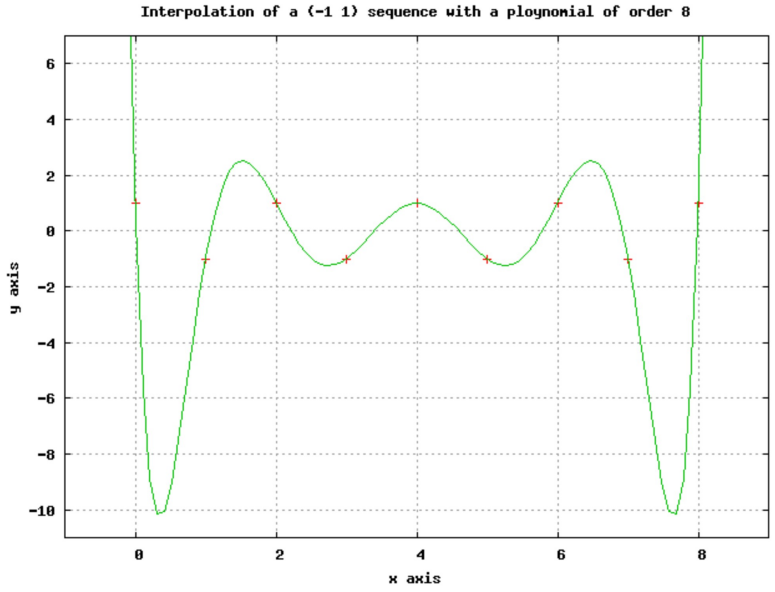


Abbildung 4.12: Oszillationen bei einem Polynom 8. Grades, aus [Schäfer 06] S. 23

Abwägung der Bahnplanungsalternativen Rapidly-Exploring Random Trees, Selbstorganisation und Neuronale Netze, Elastische Bänder, Roadmap und Potentialfelder wird auf Basis-Splines näher eingegangen, da für Basis-Splines eine konvexe Hülle in Form eines Kontrollpolynoms erstellt werden kann. Ist die topologische Karte des Freiraums bekannt, so kann ein an Hindernisse angepasstes Kontrollpolynom erstellt werden. Die anschließende Interpolation mit B-Splines liegt innerhalb der Hülle des Kontrollpolynoms und impliziert somit eine Kollisionsvermeidung. Die Verschiebung eines Stützpunkts wirkt sich bei Basis-Splines nur lokal, also nicht auf die komplette Kurve aus. Diese Eigenschaft ist von Vorteil, wenn ein Stützpunkt einer kollisionsfreien Bahn verändert wird. Somit muss nicht die komplette Bahn auf Freiheit von Kollisionen überprüft werden.
Kubische Splines basierend auf kubischen Polynomen und erfüllen die 2-fach stetige Differenzierbarkeit. Sollen kubische Splines mit beliebigen Bahnsegmenten glatt verbunden werden können, so muss die Ordnung der Splines an den Randintervallen auf vier erhöht werden.
Bézier-Splines, die auf Bernstein-Polynomen basieren, werden im Computergrafik-Bereich verwendet. Sie haben jedoch den Nachteil, dass die Kurven an den Stützpunkten vorbei und nur durch die Endpunkte verlaufen. Dies kann durch die Einführung von Hilfsstützpunkten behoben werden.
Schäfer [Schäfer 06] führt eine Parametrierung auf der erstellten Raumkurve durch, um Schleifen der interpolierten Raumkurve entgegenzuwirken. Es wird auf die äquidistante, chordale, zentripetale und auf die Foley-Parametrierung eingegangen, wie in Abbildung 4.13 gezeigt. Die äquidistante Methode erzeugt Schleifen, falls die Distanz der Stützpunkte stark verschieden ist. Chordal scheint als die beste Lösung, da sie effizient zu implementieren ist. Anschließend wird eine Reparametrisierung der Kurve durchgeführt, um die Geschwindigkeitsaspekte der Kurve zu behandeln. Ist die interpolierte Raumkurve durch $f : [a, b] \rightarrow \mathbb{C}^n$ gegeben, ergibt sich mit der Re-Parametrierungsfunktion $\xi : [r, s] \rightarrow [a, b]$ die re-parametrisierte Raumkurve

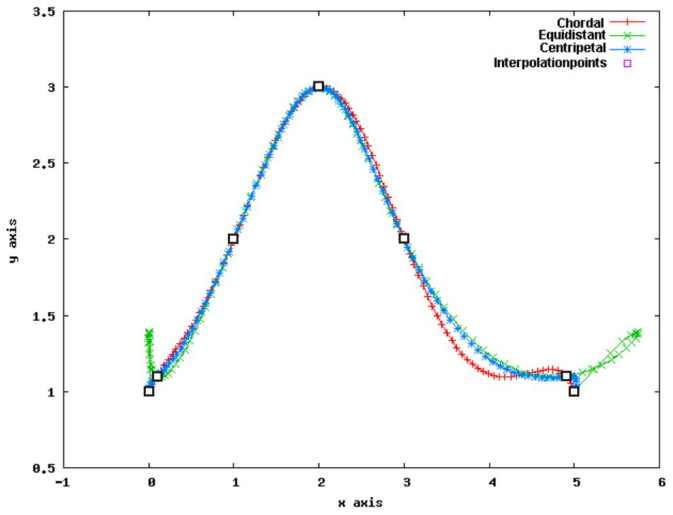


Abbildung 4.13: Schleifen in der Bahn bei äquidistanter, chordaler und zentripetaler Parametrisierung einer B-Spline Kurve, aus [Schäfer 06] S. 36

$\tilde{f}(x) = f(\xi(x))$. Durch diesen Mechanismus können mathematische Eigenschaften der Raumkurve angepasst werden. Zum Beispiel kann die Krümmung der Raumkurve in die Re-Parametrierungsfunktion einfließen, um in Bereichen starker Krümmung bei gleich bleibender Parameteränderung Δx den Roboter in diesem Bereich langsamer mit einer kleineren Änderung der Position Δf fahren zu lassen.
In [Schäfer 06] ist die Wahl der Geschwindigkeit fest mit dem Interpolations- und Re-Parametrisierungsverfahren gekoppelt. Auf Kriterien bei der Wahl der Geschwindigkeit wird außer der Bahnkrümmung, der resultierenden Zentrifugalkraft und Benutzervorgaben von Bahngeschwindigkeit und -beschleunigung nicht eingegangen. Da die Beschränkungen der Roboterkinematik und -dynamik vernachlässigt werden, ist ein zeitoptimales und ruckfreies Fahren mit diesem Ansatz nicht möglich. Eine automatische Überprüfung der Realisierbarkeit der vorgegebenen Stützstellen mit ihren Geschwindigkeitswerten ist nur teilweise realisiert.
Berglund, Jonsson und Soderkvist [Soderkvist] behandeln die Optimierung von B-Splines und die Kollisionsvermeidung. Die Kollisionsvermeidung wird mit Hilfe von den Spline-umhüllenden Polygonzügen, die sich nicht mit Hindernissen schneiden, realisiert. Die gesuchte B-Spline-Bahn soll innerhalb der Begrenzung liegen und eine minimale gesamte Bahnkrümmung besitzen.

Im Folgenden wird auf die mathematische Beschreibung der linearen, kubischen Spline- und Bézier-Interpolation näher eingegangen und die Vor- und Nachteile der Verfahren untersucht.

Linearinterpolation

Die lineare Interpolation überführt die Lage der Roboterhand im kartesischen Inertialsystem vom Startpunkt $\vec{X}_0$ in den Zielpunkt $\vec{X}_z$ linear über den Parameter s mit

$$\vec{X}(s) = \vec{X}_0 + \frac{s}{s_z}\left(\vec{X}_z - \vec{X}_0\right). \tag{4.3}$$

Die gefahrene Gesamtstrecke berechnet sich zu

$$s_z = \sqrt{(x_z - x_0)^2 + (y_z - y_0)^2 + (z_z - z_0)^2} \tag{4.4}$$

und für die Ableitungen am Bahnanfang und -ende gilt

$$\begin{aligned} \vec{X}'(s_0) &= \vec{X}'(s_z) = \frac{1}{s_z}(\vec{X}_z - \vec{X}_0) \\ \vec{X}''(s_0) &= \vec{X}''(s_z) = 0\,. \end{aligned} \tag{4.5}$$

Obwohl eine Bahn mit linear interpolierten Segmenten nicht die Anforderung der 2-fach stetigen Differenzierbarkeit erfüllt, wird sie in kommerziellen Systemen häufig verwendet. Sie wird vor allem dann eingesetzt, wenn gerade geometrische Konturen von Werkstücken abgefahren werden. Lineare Segmente können glatt über Splines verbunden werden.

Diese Interpolationsart ist einfach, effizient und von praktischem Nutzen für KHS-Roboter: Der KHS-Einleger muss Zwischenlagen aus einem Korb holen und sich dabei exakt senkrecht bewegen. Außerdem liefern die vorgestellten Bahnplanungsverfahren oft einzelne Stützpunkte, wobei nur die lineare Verbindung dieser Punkte zu einer kollisionsfreien Bahn führt. Bei einer folgenden Optimierung können die linearen Segmenten in eine günstigere Interpolationsart überführt werden. Kapitel 5 geht auf diese Optimierung ein.

Kubische Splines

Splines eignen sich für die Erzeugung beliebig komplexer Konturen, wobei die Stetigkeit bis hin zur zweiten Ableitung nach dem Bahnparameter ein ruckbegrenztes Fahren ermöglicht.
Folgende Spline-Modelle basieren auf den Verfahren aus [Olomski 89].

Eine kubische Splinefunktion über n Stützstellen (s_i, f_i) ist die partiell definierte Funktion

$$F(s) := \begin{cases} P_1(s)\,, & \text{für } s \in [s_1, s_2] \\ P_2(s)\,, & \text{für } s \in [s_2, s_3] \\ .. & \\ P_{n-1}(s)\,, & \text{für } s \in [s_{n-1}, s_n] \end{cases} \tag{4.6}$$

aus $n-1$ kubischen Polynomen P_i

$$P_i : [s_i, s_{i+1}] \to \mathbb{R} \text{ mit } P_i(s) := a_i + b_i\,(s - s_i) + c_i\,(s - s_i)^2 + d_i\,(s - s_i)^3\,. \tag{4.7}$$

Um das Gleichungssystem mit den Koeffizienten a_i, b_i, c_i und d_i eindeutig zu lösen, werden $4(n-1)$ Bedingungen benötigt. Für jedes der $n-1$ Intervalle entstehen die Interpolationsbedingungen

$$\begin{aligned} P_i(s_i) &= f_i, i = 1...n-1 \\ P_i(s_{i+1}) &= f_{i+1}, i = 1...n-1\,. \end{aligned} \tag{4.8}$$

Weitere 2 $(n-2)$ Bedingungen ergeben sich aus der Forderung nach der 2-fach stetigen Differenzierbarkeit für die inneren Stützstellen

$$\begin{aligned} P_i'(s_{i+1}) &= P_{i+1}'(s_{i+1}), i = 1...n-2 \\ P_i''(s_{i+1}) &= P_{i+1}''(s_{i+1}), i = 1...n-2. \end{aligned} \tag{4.9}$$

Somit bleiben 2 Randbedingungen offen, die für einen

- natürlichen Spline mit $P_1''(s_1) = P_{n-1}''(s_n) = 0$, also die Kurve geht am Anfang und am Ende in eine Gerade über,
- eingespannten Rand mit $P_1'(s_1) = f_1'$ und $P_{n-1}'(s_n) = f_{n-1}'$ mit vorgegebenen f_1' und f_{n-1}' einer angrenzenden Funktion
- periodischen Rand mit $P_1'(s_1) = P_{n-1}'(s_n)$ und $P_1''(s_1) = P_{n-1}''(s_n)$

verwendet werden können.

Für die Ableitungen der Polynome gilt

$$\begin{aligned} P_i'(s) &= b_i + 2\,c_i\,(s-s_i) + 3\,d_i\,(s-s_i)^2 \\ P_i''(s) &= 2\,c_i + 6\,d_i\,(s-d_i)\,. \end{aligned} \tag{4.10}$$

Für die Interpolation der kartesischen Lage der Roboterhand $\vec{X}(s)$ wird je eine Splinefunktion pro Koordinatenrichtung und Winkel benötigt.

Die Teilstrecken für die Parametrisierung mit s zwischen den Stützpunkten berechnet sich zu

$$\begin{aligned} \Delta s_i &= s_{i+1} - s_i = \int_{s_i}^{s_{i+1}} \sqrt{x'(s)^2 + y'(s)^2 + z'(s)^2}\,ds \\ &\approx \sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2 + (z_{i+1}-z_i)^2}\,, \end{aligned} \tag{4.11}$$

wobei die Approximation durch den Polygonzug vor allem bei starken Bahnkrümmungen zu großen Abweichungen führt, die in einer erneuten Berechnung der Bahnlänge über numerische Integration berücksichtigt werden müssen.

Kubische Splines basieren auf kubischen Polynomen, welche die einfachsten Funktionen sind um die geforderte C_2-Stetigkeit zu erfüllen. Wegen dieser Eigenschaft ist dieser Interpolationstyp ein geeigneter Kandidat zur Implementierung der Online-Bahninterpolation für die KHS-Roboter. Bevor der Aufwand dieser Interpolation abgeschätzt und verbessert wird, werden zunächst weiter Verfahren auf ihre Anwendbarkeit untersucht.

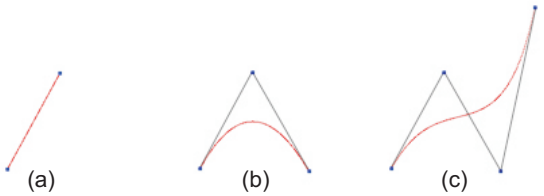


Abbildung 4.14: Bézierkurve mit Kontrollpolygon des Grade 1, 2 und 3, aus [Wikipedia 07]

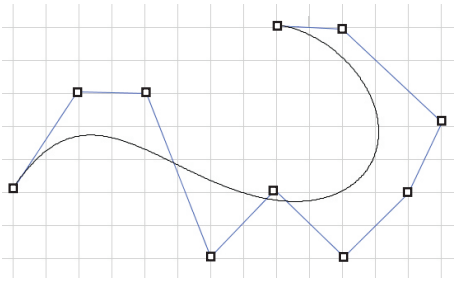


Abbildung 4.15: Die Bézierkurve verläuft an den Stützpunkten vorbei.

Bernsteinpolynome, Bézier- und B-Splines

Eine Bézierkurve $C(t)$ mit Grad n und $n+1$ Stützpunkten P_i (Kontrollpolygon) mit dem Parameter $t \in [0,1]$ basiert auf Bernsteinpolynomen $B_{i,n}$ mit

$$C(t) := \sum_{i=0}^{n} B_{i,n}(t)\ P_i \tag{4.12}$$

und

$$B_{i,n}(t) := \binom{n}{i} t^i\ (1-t)^{n-i}\ . \tag{4.13}$$

Abbildung 4.14 zeigt Bézierkurven des Grades $n = 1$, $n = 2$ und $n = 3$. Die Kurven verlaufen durch die Endpunkte P_0 und P_n, aber nicht durch die Zwischenpunkte. Die Zwischenpunkte „ziehen“ die Kurve an sich wie in Abbildung 4.15 dargestellt. Eine positive Eigenschaft der Bézierkurve ist, dass die konvexe Hülle des Kontrollpolygons die Kurve komplett umschließt. Liegt die Hülle im Freiraum, so ist die Bahn kollisionsfrei.

Abbildung 4.16 zeigt die Konstruktion einer kubischen Bézierkurve mit $n = 3$ nach

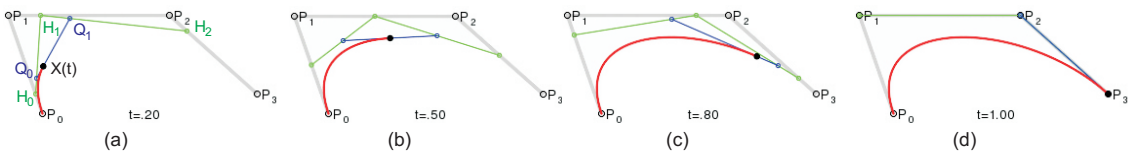


Abbildung 4.16: Kontruktion einer kubischen Bézierkurve nach [de Casteljau 59], aus [Wikipedia 07]

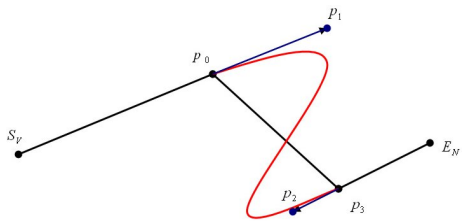


Abbildung 4.17: Einfluss der Position der Stützpunkte der benachbarten Spline-Segmente S_v und E_n auf das Kontrollpolygon mit P_1 und P_2 die Konstruktion eines Bézier-Splines, aus [Sohn 04] S. 76

dem De Casteljau-Algorithmus [de Casteljau 59]. Das Kontrollpolygon der Bézierkurve wird rekursiv geteilt. Die Kurven der Teilpolygone führen zu einer Annäherung der originalen Kurve.
Die Punkte H_i laufen linear mit t von P_i nach P_{i+1} entlang der Strecke $\overline{P_i\, P_{i+1}}$. Die Punkte Q_i laufen linear mit t von H_i nach H_{i+1} entlang der Strecke $\overline{H_i\, H_{i+1}}$. Der Punkt $X\,(t)$ läuft entlang der Strecke $\overline{Q_0\, Q_1}$.
Da eine Verschiebung eines Kontrollpunktes die Gestalt der Kurve global ändert, werden in [Sohn 04] kubische Bézier-Splines verwendet. Die Lage der zwei inneren Kontrollpunkte des Splines werden aus der Position der Kontrollpunkte der benachbarten Splines wie in Abbildung 4.17 abgeleitet, z. B. mit

$$p_1 = p_0 + \tau\ (p_0 - S_v)\ . \tag{4.14}$$

Der Parameter τ hat einen Einfluss auf die Ausprägung der Kurven. Mit $\tau = 0$ entsteht eine lineare Interpolation, ein sinnvoller Wert für τ liegt nach [Sohn 04] in $[1,2; 1,7]$. An den Enden des Splines existieren keine Nachbarpunkte. Dort wird die Ableitung auf 0 gesetzt.

B-Splines haben ähnliche Eigenschaften wie Bézier-Splines. Anstelle des Bernsteinpolynoms verwenden sie eine andere Basisfunktion, die durch die Rekursionsformel von de Boor/Cox/Mansfield definiert ist. Bei Bézierkurven hängt die Anzahl der Kontrollpunkte direkt mit dem Grad der Kurve zusammen. B-Splines benötigen weniger Kontrollpunkte zur Darstellung einer ähnlichen Kurve. B-Splines verlaufen nicht durch die Endpunkte, sind rechenintensiver aber flexibler. Bézierkurven und B-Splines lassen sich ineinander überführen.

Die Verfahren dieses Abschnitts liefern im Vergleich zu kubischen Splines einen weicheren Kurvenverlauf, der höhere Bahngeschwindigkeiten erlauben kann. Der Rechenaufwand ist jedoch höher und die Bahnen haben den Nachteil, dass sie i. d. R. nicht durch die Kontrollpunkte verlaufen, sondern durch sie lediglich abgelenkt werden. Dies resultiert in einem aufwändigeren Kollisionstest: Obwohl Kontrollpunkte im Hindernisraum liegen können, kann die Bahn kollisionsfrei sein. Eine Bahn mit kubischen und linearen Segmenten kollidiert immer, wenn ein Stützpunkt in einem Hindernis liegt. Diese Eigenschaft erlaubt einen schnellen Vorab-Kollisionstest und der Bediener der Bahnplanung kann sicher sein, dass die Bahn durch dem von ihm definierten Stützpunkt exakt verläuft und sich nicht nur in seiner Nähe befindet.

Aus diesen Gründen wird die kubische Splineinterpolation ausgewählt und der Rechenaufwand im folgenden Abschnitt für den Online-Einsatz verbessert.

4.2.2 Effiziente Berechnung kubischer Splines

Für die Berechnung der kubischen Spline-Koeffizienten müssen $4\,(n-1)$ Gleichungen gelöst werden. Das Gleichungssystem kann nach Standardverfahren der Numerik z. B. Gauß-Jordan mit $\mathcal{O}\,(n^3)$ gelöst werden. Dies ist rechenaufwändig.
Bézierkurven sind ebenfalls rechenaufwändig und die Bahn verläuft nicht durch die Zwischenstützpunkte.
B-Splines bieten durch das Kontrollpolygon Freiheitsgrade bei der Kurvengestaltung. Die Lage der Kontrollpunkte ist nicht eindeutig festgelegt und wäre Gegenstand einer eigenständigen Optimierung.

In [Press 92] wird eine effiziente Variante der Berechnung von kubischen Spline-Koeffizienten vorgestellt, die für diese Arbeit verwendet wird. Sie wurden ursprünglich in [Boor 77] beschrieben.
Die Linearinterpolation einer Raumkoordinate X aus Gleichung (4.3) kann als Lagrangeinterpolation

$$X = A\,X_i + B\,X_{i+1} \tag{4.15}$$

mit

$$\begin{aligned} A &:= \frac{s_{i+1} - s}{s_{i+1} - s_i} \\ B &:= 1 - A = \frac{s - s_i}{s_{i+1} - s_i} \end{aligned} \tag{4.16}$$

dargestellt werden.
Angenommen, die 2. Ableitung X_i'' an den Stützstellen s_i sei bereits bekannt. Dann kann zur geforderten Erfüllung der Stetigkeit der 2. Ableitung der Interpolation zur rechten Seite der Gleichung (4.15) ein kubisches Polynom, dessen 2. Ableitung sich linear von X_i'' an der Stelle s_i nach X_{i+}'' an s_{i+1} ändert, addiert werden. Dieses Polynom muss außerdem solch eine Gestalt haben, dass es an den Stellen s_i und s_{i+1} den Wert 0 annimmt.
Nach der Addition dieses Polynoms zur rechten Seite von Gleichung (4.15) ergibt sich

$$X = A\,X_i + B\,X_{i+1} + C\,X_i'' + D\,X_{i+1}'' \tag{4.17}$$

mit den eindeutigen

$$\begin{aligned} C &:= \frac{1}{6}\left(A^3 - A\right)\,(s_{i+1} - s_i)^2 \\ D &:= \frac{1}{6}\left(B^3 - B\right)\,(s_{i+1} - s_i)^2\ . \end{aligned} \tag{4.18}$$

Die 2. Ableitungen X_i'' sind nicht bekannt. Sie können mit der 1. Ableitung von Gleichung (4.17)

$$\frac{\partial X}{\partial s} = \frac{X_{i+1} - X_i}{s_{i+1} - s_i} - \frac{3\,A^2 - 1}{6}\,(s_{i+1} - s_i)\,X_i'' + \frac{3\,B^2 - 1}{6}\,(s_{i+1} - s_i)\,X_{i+1}'' \tag{4.19}$$

Abbildung 4.18: Linear- und Spline-Interpolation gemischt (aus Simulation)

berechnet werden, indem in Gleichung (4.19) s mit s_i aus $[s_{i-1}, s_i]$ substituiert und mit Gleichung (4.19) mit $s = s_i$ aus $[s_i, s_{i+1}]$ gleichgesetzt wird. Nach einer Umformung entstehen $n-2$ Gleichungen

$$\frac{s_i - s_{i-1}}{6} X''_{i-1} + \frac{s_{i+1} - s_{i-1}}{3} X''_i + \frac{s_{i+1} - s_i}{6} X''_{i+1} = \frac{X_{i+1} - X_i}{s_{i+1} - s_i} - \frac{X_i - X_{i-1}}{s_i - s_{i-1}} \quad (4.20)$$

für $i = 2...n-1$.

Wie oben in der ersten Spline-Variante bleiben zwei Bedingungen offen, die für die Erstellung eines natürlichen Splines mit $X''_1 = X''_{n-1} = 0$ oder für eingespannte Ränder verwendet werden können. Bei eingespannten Rändern mit vorgegebenen X'_1 und X'_{n-1} berechnet sich die X''_i durch einsetzen in Gleichung (4.19).

Das zu lösende Gleichungssystem basiert auf der Gleichung (4.20) und hängt linear von der gesuchten Größe X''_i ab. Jedes X''_i ist dabei nur von seinen Nachbarn $i \pm 1$ abhängig. Somit kann das System mit einem tridiagonalen Algorithmus mit einer Bandmatrix, bei der nur die Hauptdiagonale und je eine Nebendiagonale besetzt ist, in $\mathcal{O}(n)$ gelöst werden.

Nicht nur die initiale Berechnung der Spline-Koeffizienten, sondern auch der Aufwand der ständigen Auswertung der Spline-Funktion im Takt der Antriebssteuerung sind maßgeblich für die Effizienz in Hinblick auf einen online-Einsatz im Roboter. [Press 92] verwendet für die Berechnung der Interpolationswerte Gleichung (4.17) und ist somit nicht effizienter als der klassische Ansatz. Außerdem ist eine beidseitige Wahl der X'_i- und X''_i-Werte an den Rändern eines Splines zur Verbindung mit einem anderen Spline, einer Gerade oder eines beliebigen Bahnsegments wie in Abbildung 4.18 nicht behandelt. Der für die Interpolationsfunktion notwendige, genaue Wert des Bahnparameters s ist nicht bekannt, da er von der Länge der anschließend erzeugten interpolierten Raumkurve abhängt.

Die Defizite werden mit folgendem Ansatz behoben. Die Funktion soll anhand eines gewählten Abschnitts i und eines lokalen Bahnparameters t den Interpolationswert liefern. Eine Umformung von Gleichung (4.17) mit der Bedingung $|s_{i+1} - s_i|$ ergibt für jede kartesische Raumkoordinate oder Winkel j

$$X_j(i,t) := t\, X_{j,i+1} - \frac{1}{6}\left((t-1)\, t\, \left((t-2)\, X''_{j,i} - (t+1)\, X''_{j,i+1}\right)\right) + X_{j,i} - t\, X_{j,i} \quad (4.21)$$

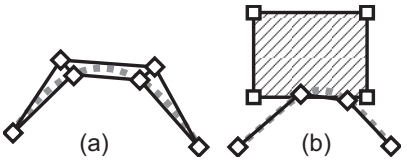


Abbildung 4.19: Diskretisierung der Bahn mit a) konvexen Hüllpolygonen b) Linien

mit

$$\begin{aligned} i &= 1..n-1 \\ t &\in [0,1] \ . \end{aligned} \tag{4.22}$$

Die Berechnung der Interpolationskoeffizienten für Splines und lineare Stücke wird miteinander verknüpft, so dass glatte Übergänge mit angepassten Krümmungen bei Bahnen mit gemischten Interpolationsarten realisiert werden können. Der Quellcode der Initialisierung ist im Anhang D dargestellt.

4.2.3 Kollisionsüberprüfung

Nachdem über die Stützstellen der Bahnplanung aus Abschnitt 4.1 mit einer Interpolation eine glatte Bahn gelegt wurde, kann diese je nach der Ausprägung der hinzugefügten Kurven mit Hindernissen kollidieren.

Sohn [Sohn 04] verwendet den Parameter τ, um Kurven der Bézier-Splines zu Lasten der Bahngeschwindigkeit enger zu gestalten. Auf einen expliziten Kollisionstest wird nicht eingegangen. Im Folgenden wird ein Verfahren für einen einfachen Kollisionstest in der Ebene entwickelt.

Die interpolierte Bahn wird diskretisiert und ein Test nach Satz (1) durchgeführt. Für die Quantisierung kann eine Hülle aus konvexen Polygonen erstellt werden wie in Abbildung 4.19a) oder eine Annäherung aus Strecken wie in Abbildung 4.19b). Die Hüllen-Variante ist sicherer, verschenkt aber mehr Platz. Der Kollisionstest mit einem Polygonzug ist effizienter zu berechnen.
Für jede Teilstrecke des Polygonzugs $\overline{S_i\, S_{i+1}}$ muss überprüft werden, ob das Hindernis komplett in einer Halbebene liegt

$$\text{sign}\left((\vec{S}_{i+1} - \vec{S}_i) \times (\vec{P}_j - \vec{S}_i)\right) = \text{ const } \forall \text{ Eckpunkte } P_j \text{ eines Hindernispolygons .} \tag{4.23}$$

Zudem muss für jede Hinderniskante $\overline{P_j\, P_{j+1}}$ überprüft werden, ob die Teilstrecke des Polygonzugs $\overline{S_i\, S_{j+i}}$ komplett in einer Halbebene liegt

$$\begin{aligned} &\text{sign}\left((\vec{P}_{j+1} - \vec{P}_j) \times (\vec{S}_i - \vec{P}_j)\right) \\ &= \text{sign}\left((\vec{P}_{j+1} - \vec{P}_j) \times (\vec{S}_{i+1} - \vec{P}_j)\right) \forall \text{ Kanten } j \text{ des Hindernisses .} \end{aligned} \tag{4.24}$$

Ist für eine Teilstrecke Gleichung (4.23) oder (4.24) erfüllt, dann ist sie kollisionsfrei.

Falls die Beschreibung der Hindernisse im diskreten Konfigurationsraum wie in Abschnitt 4.1.8 vorliegt, kann die Bahn numerisch nach $\mathbb{C}$ transformiert werden und dort auf Kollisionen mit Hinderniskonfigurationen getestet werden. Liegt eine interpolierte Bahn numerisch vor, also mit m durch die Interpolation eingefügten Zwischenpunkten zwischen je zwei der n Stützpunkten, muss jeder der $m\ (n-1)+1$ Punkte in den Konfigurationsraum $\mathbb{C}$ transformiert werden. Falls für eine Konfiguration $\vec{c}_i \in \mathbb{C}_{Hindernis}$ gilt, kollidiert die Bahn. Der Aufwand des Kollisionstests liegt in $\mathcal{O}(m \cdot n)$.

Falls eine Kollision auftritt, kann an dieser Stelle ein neuer Stützpunkt eingefügt und anschließend außreichend weit in den Freiraum verschoben werden. Nach der Interpolation wird erneut auf Kollisionen getestet.

4.3 Geschwindigkeitsprofil mit Randbedingungen

In den vorherigen Abschnitten wurde die Erstellung der Bahn erläutert. Gelenkstellungen können in einem beliebig feinen Abtastraster über den Bahnparameter s ausgelesen und an die Robotersteuerung übergeben werden. Die Bahn als glatte geometrische Raumkurve beinhaltet aber nur den statischen Teil der Trajektorie. In den folgenden Abschnitten wird die Erstellung des Geschwindigkeitsprofils beschrieben. Damit der Roboter ökonomisch arbeitet, soll das Geschwindigkeitsprofil einerseits eine möglichst schnelle Fahrt ermöglichen, andererseits müssen Randbedingungen wie maximales Drehmoment der Antriebe berücksichtigt werden, so dass die Fahrt über die Solltrajektorie keine Grenzen der Roboter-Dynamik verletzt. Da ein zeitoptimales Fahren nicht immer erwünscht ist, z. B. wenn für die Synchronisation mit anderen Aktuatoren eine bestimmte Bahngeschwindigkeit eingehalten werden soll, müssen Benutzervorgaben ebenfalls berücksichtigt werden.

Johanni [Johanni 88] befasst sich mit der Wahl des optimalen Geschwindigkeitsverlaufs bei gegebener Bahnkontur. Auf Basis der dynamischen Beschränkungen werden Bahngeschwindigkeiten zu markanten Bahnstellen berechnet. Aus diesen Punkten wird mit Interpolation einen Verlauf erzeugt. Suchverfahren und iterative Optimierungen führen zu dem gesuchten Geschwindigkeitsverlauf. Dieses Verfahren ist durch die Interpolation zu ungenau, da es Geschwindigkeiten zwischen den Punkten vernachlässigt. Exakte Verfahren führen zu einem realistischen aber nicht-linearen Geschwindigkeitsverlauf, der nur numerisch abspeicherbar ist.

Für diese Arbeit wird für die Wahl der Geschwindigkeit ein suboptimales Näherungsverfahren verwendet, das die Roboterdynamik miteinbezieht. Als Geschwindigkeit wird die Bahngeschwindigkeit im kartesischen Raum gewählt, da diese für den Benutzer von größtem Interesse ist. Das Ergebnis der Geschwindigkeitsberechnung ist eine kleine Menge von (s_i, v_i)-Punkten, die unabhängig von den Stützstellen der Bahn sind. Mit Hilfe dieser Stützpunkte kann die Robotersteuerung online die Geschwindigkeit abschnittsweise berechnen. Das Verfahren zur Ermittlung dieser

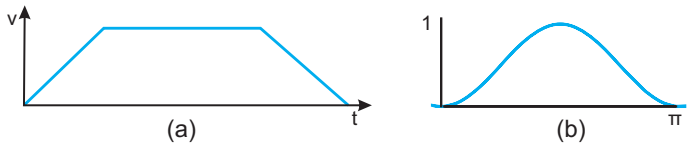


Abbildung 4.20: a) einfaches, trapezförmiges Geschwindigkeitsprofil, b) $\sin^2$ als Grundfunktion für ruckbegrenzte Beschleunigungen

Geschwindigkeit-Stützpunkte soll ebenfalls online und auch während der Fahrt ausgeführt werden können.

Abschnitt 4.3.1 greift zunächst Zeitprofile $a\,(t)$ mit definiertem Ruck für die Durchführung von Geschwindigkeitsänderungen Δv auf den Bahnabschnitten Δs ein voraus. Die Geschwindigkeiten und Strecken werden aus dem (s_i, v_i)-Geschwindigkeitsprofil gewonnen.

Abschnitt 4.3.2 behandelt die Mehr-Zieloptimierung des kompletten Geschwindigkeitsprofils. Die Ergebnisse werden mit (s_i, v_i)-Punkten beschrieben.

4.3.1 Ruckfreie Beschleunigungsprofile

Die Aufgabe des Generators für Trajektorien ist die Überführung der Gelenkstellungen des Roboters von der Startkonfiguration $\vec{q}(t_0)$ in die Zielkonfiguration $\vec{q}(t_z)$. Es gibt viele Möglichkeiten für diese Überführung. Einfache Generatoren nutzen ausschließlich Benutzervorgaben des Profils im Gelenkwinkelraum und erstellen ein Geschwindigkeitsprofil mit einer linearen Abhängigkeit der Bahngeschwindigkeit $v = \dot{s}$ von der Zeit. Diese Profile haben eine trapezförmige Gestalt wie in Abbildung 4.20a). Die Knicke im Geschwindigkeitsverlauf wirken sich jedoch nachteilig aus, da der Roboter aufgrund seiner Trägheit den Unstetigkeiten der Beschleunigung nicht folgen kann und somit einen Bahnfehler erzeugt und sich aufschwingen kann. Eine Bahn soll frei von solchen Rucken sein. Dies kann durch eine Beschleunigung auf Basis der Sinusfunktion aus Abbildung 4.20b) gewährleistet werden.

Craig [Craig 05] geht auf die Erstellung eines Profils ein und beschreibt Maßnahmen zur Glättung der Geschwindigkeit. Für die Stellungen eines Gelenks im Zeit-Gelenkstellungprofil gelten die Bedingungen

$$\begin{aligned} q\,(t_0) &= q_0 \\ q\,(t_z) &= q_z \end{aligned} \tag{4.25}$$

und für die Gelenkgeschwindigkeiten im Start- und Zielpunkt wird

$$\begin{aligned} \dot{q}\,(t_0) &= 0 \\ \dot{q}\,(t_z) &= 0 \end{aligned} \tag{4.26}$$

vorausgesetzt. In einer ersten Variante werden kubische Polynome verwendet und ihre Koeffizienten anhand der vier Bedingungen eindeutig bestimmt. Dies führt zu parabolischen Geschwindigkeitsverläufen und linearen Beschleunigungsverläufen.

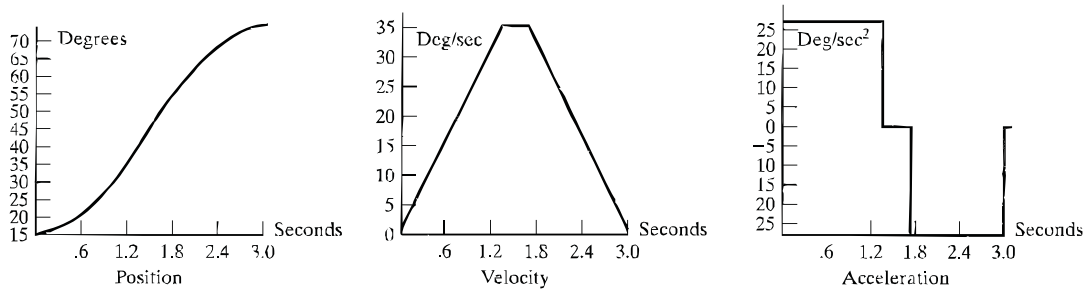


Abbildung 4.21: Verschliffenes, lineares Geschwindigkeitsprofil als Resultat der Überlagerung einer linearen und parabolischen Funktion, angelehnt an [Craig 05] S. 213

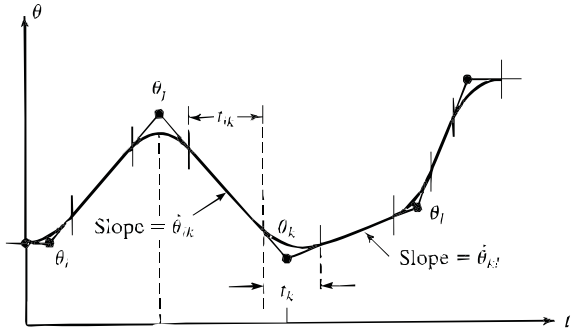


Abbildung 4.22: Verkettung einer Bahn aus mehreren verschliffenen Segmenten, aus [Craig 05] S. 214

Diese Variante kann um beliebige Zwischenpunkte im Profil erweitert werden mit $\dot{q}(t_i) = q_i$. Die Steigung zwischen den Stützpunkten $\frac{q_{i+1}-q_i}{t_{i+1}-t_i}$ gibt jeweils die Geschwindigkeit in einem Stützpunkt (t_i, q_i) vor. Um neben der Geschwindigkeit auch die Beschleunigung an den Punkten vorzugeben, wird die Ordnung der Polynome erhöht.
Die zweite Variante verwendet eine Geschwindigkeitsfunktion, die aus der Überlagerung einer linearen und einer parabolischen Funktion besteht. Die Parabel wird zum Verschleifen der Ecken verwendet und führt zu einem weichen Verlauf, siehe Abbildung 4.21. Dieses Verfahren kann ebenfalls für über Zwischenpunkte spezifizierte Profile angewendet werden und führt zu dem Ergebnis aus Abbildung 4.22. Nur der Start- und Zielpunkt werden bei dieser Methode exakt durchfahren. Ist ein exaktes Durchfahren eines gegebenen Zwischenpunkts notwendig, muss dieser durch zwei Pseudo-Punkte ersetzt werden.
Bei den Verfahren aus [Craig 05] sind Fahrzeit, Gelenkposition und -geschwindigkeit direkt miteinander verknüpft und macht eine getrennte Planung von Bahn und Geschwindigkeit unmöglich. Außerdem werden keine Randbedingungen wie Ruck oder Maximalgeschwindigkeit miteinbezogen und die Frage nach den optimalen Werten für das Verschleifen bleibt offen.

In [Olomski 89] wird das Geschwindigkeitsprofil im kartesischen Raum losgelöst von der konkreten Bahn behandelt, indem ausschließlich auf den Bahnparameter s zurückgegriffen wird. Das Geschwindigkeitsprofil wird mit wenigen Punkten (s_i, v_i) beschrieben.

Um den Bahnruck

$$r := \dot{a} = \ddot{v} = \dddot{s} \tag{4.27}$$

und somit die Stetigkeit der Beschleunigung in die Modellierung zu integrieren, wird ein Modell dritter Ordnung gewählt. Ruckfrei bedeutet hier lediglich eine Begrenzung des Rucks. Basierend auf diesem Modell wird die Erstellung der Zeitprofile in diesem und nächsten Abschnitt hergeleitet. Zur Abgrenzung von der Roboterdynamik wird das Modell zur Generierung der Zeitprofile als Geschwindigkeitsmodul bezeichnet.

Das Ziel des Geschwindigkeitsmoduls ist die Überführung der Startwerte

$$\begin{aligned} s\left(t_0\right) &= s_0 \\ v\left(t_0\right) &= v_0 \\ a\left(t_0\right) &= 0 \end{aligned} \tag{4.28}$$

unter den Randbedingungen

$$\begin{aligned} \left|v\left(t\right)\right| &\leq v_{max} \\ \left|a\left(t\right)\right| &\leq a_{max} \\ \left|r\left(t\right)\right| &\leq r_{max} \end{aligned} \tag{4.29}$$

in die Endwerte

$$\begin{aligned} s\left(t_z\right) &= s_z \\ v\left(t_z\right) &= v_z \\ a\left(t_z\right) &= 0, \end{aligned} \tag{4.30}$$

eines Streckenabschnitts. Die Verknüpfung von Streckensegmenten ergeben einen kompletten Verlauf der Profile wie in Abbildung 4.23 gezeigt.

Aus der Beschreibung des Geschwindigkeitprofils wie in Abbildung 4.24 mit den Punkten (s_i, v_i) wird für jeden Streckenabschnitt Δs und der gewünschten Änderung der Geschwindigkeit Δv ein entsprechendes Beschleunigungsprofil gesucht, so dass an den Rändern des Abschnitts $a = 0$ gilt. Der gesamte Geschwindigkeitsverlauf kann somit in einzelne Zeitsegmente unterteilt werden. Die Zeitsegmente müssen nicht mit den geometrischen Abschnitten der Bahn, die durch die Bahnstützpunkte begrenzt sind, übereinstimmen.
Der Geschwindigkeitsverlauf der kompletten Stecke ergibt sich aus der Abfolge von drei möglichen Beschleunigungs- und Bremsverläufen:

- $a = 0$: Geschwindigkeit halten
- Dreieck-Profil für a
- durch a_{max} begrenztes Profil: Trapez-Profil,

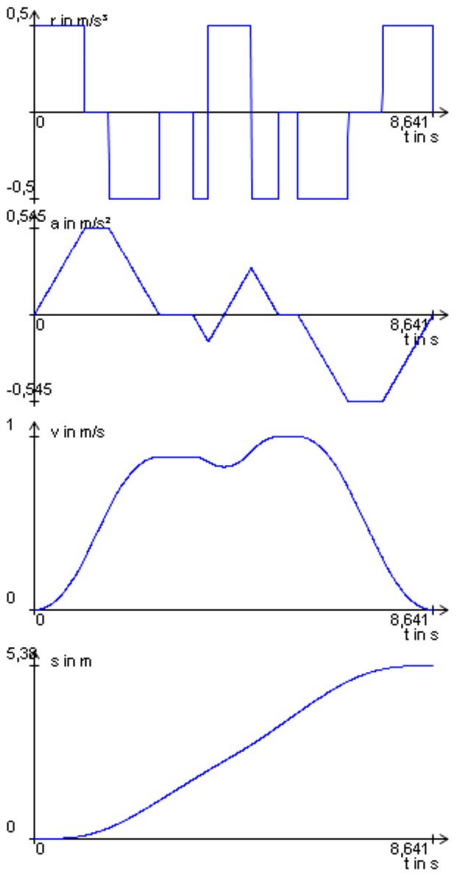


Abbildung 4.23: Zeitprofile mit Ruck, Beschleunigung, Geschwindigkeit und Bahnstrecke (aus Simulation)

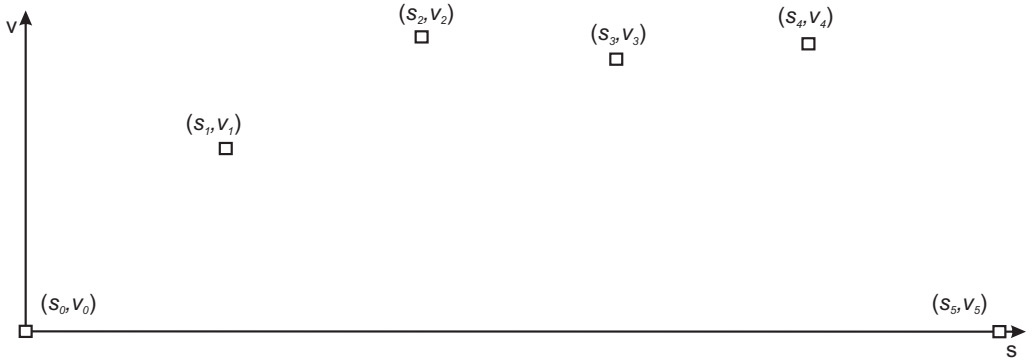


Abbildung 4.24: Beschreibung eines Geschwindigkeitsverlaufs mit diskreten (s_i, v_i)-Punkten (aus Simulation)

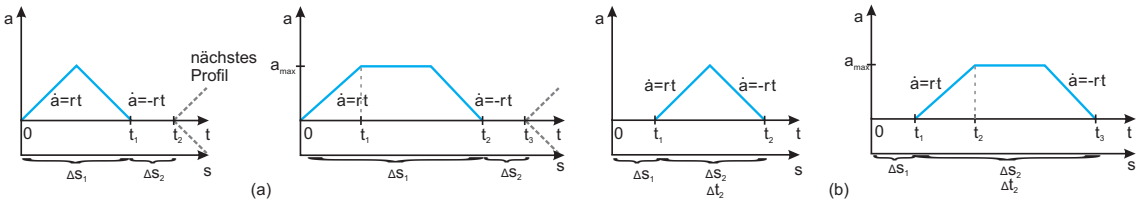


Abbildung 4.25: Dreieck- und trapezförmige Beschleunigungsprofile mit (a) sofortiger und (b) verzögerter Beschleunigung

wobei für eine möglichst hohe Beschleunigung und schnelle Fahrt stets der Ruck $|\dot{a}| = |r| = r_{max}$, wie in Abbildung 4.25a) dargestellt, betragen soll. Zum Zeitpunkt t_1 ist das Dreieckprofil beendet und die Geschwindigkeitsänderung Δv wurde erreicht. Analog wurde zum Zeitpunkt t_2 das Trapezprofil durchfahren und die Bahngeschwindigkeit wird gehalten. Zum Zeitpunkt t_2 wird das Ende des Streckenabschnitts Δs beim Dreieckprofil erreicht, beim Trapezprofil bei t_3. Abbildung 4.25b) zeigt Beschleunigungsprofile mit vorausgehender konstanter Geschwindigkeit. Sie finden weiter unten ihre Anwendung, falls eine bestimmte Geschwindkeitsdifferenz und Strecke überwunden werden sollen, aber wegen weiteren Randbedingungen nicht sofort beschleunigt werden darf.

Es muss sichergestellt sein, dass die Fahrzeit, Bahnabschnittsstrecke, Ruck und maximale Beschleunigung so gewählt sind, dass die einzelnen Zeitabschnitte ohne Verletzung der Randbedingungnen abfahrbar sind. Es wird zunächst ermittelt, welche Bahnstrecke und Fahrzeit für eine gewünschte Geschwindigkeitsänderung für beide Profilarten (Dreieck und Trapez) benötigt wird.

Um den Aufwand und die Rundungsfehler einer numerischen Berechnung von $v\,(t)$ und $s\,(t)$ aus dem Beschleunigungsprofil $a\,(t)$ zu verringern, werden die unterschiedlichen Varianten der Profile zunächst analytisch integriert. Das Ziel ist es, die Beschleunigungsprofile so darzustellen, dass

1. der Bahnparameter s für die Interpolation durch die Ausführung einer einfachen Funktion $s\,(t)$ mit dem Parameter t schnell während der Fahrt des Roboters gewonnen werden kann,

2. die Dauer des Profils Δt berechnet werden kann, um die gesamte Fahrdauer der Bahn zu berechnen.

Die Zeitprofile für den eigenen Lösungsansatz werden in den folgenden Abschnitten hergeleitet. Für die Beschleunigungsprofile wird vorausgesetzt, dass der Bahnabschnitt Δs ausreichend lang ist, um die gewünschte Geschwindigkeitsdifferenz Δv mit begrenztem Ruck r zu erreichen. Wie dieses Kriterium erfüllt wird, ist weiter unten beschrieben.

Dreickprofil für sofortige Beschleunigung

Das Dreicksprofil wird durch

$$a(t) := \begin{cases} r\,t, & \text{für } t \in [0,\ \frac{1}{2}\,t_1] \\ -r\,t + r\,t_1, & \text{für } t \in]\frac{1}{2}\,t_1,\ t_1] \\ 0, & \text{für } t > t_1 \end{cases} \tag{4.31}$$

definiert. Es soll innerhalb der Bahnabschnitts Δs die Geschwindigkeitsdifferenz Δv mit dem festen Ruck r herbeiführen. Ist der Bahnabschnitt Δs größer als für die Geschwindigkeitsänderung nötig, so wird die Geschwindigkeit bis zum Ende beibehalten.

Das Geschwindigkeitsprofil mit der Startgeschwindigkeit $v_{start} = 0$ berechnet sich zu

$$v(t) := \int_0^t a(\tau)\ d\tau = \begin{cases} \frac{1}{2} r\,t^2, & \text{für } t \in [0,\ \frac{1}{2}\,t_1] \\ -\frac{1}{4} r\,(2\,t^2 - 4\,t\,t_1 + t_1^2), & \text{für } t \in]\frac{1}{2}\,t_1,\ t_1] \\ \frac{1}{4} r\,t_1^2, & \text{für } t > t_1 \end{cases} \tag{4.32}$$

und für die Bahnstrecke mit Startstrecke $s_{start} = 0$ gilt

$$\begin{aligned} s(t) &:= \int_0^t v(\tau)\ d\tau \\ &= \begin{cases} \frac{1}{6} r\,t^3, & \text{für } t \in [0,\ \frac{1}{2}\,t_1] \\ \frac{1}{24} r\,t_1^3 - \frac{1}{6} r\,t^3 + \frac{1}{2} r\,t_1\,t^2 - \frac{1}{4} r\,t_1^2\,t, & \text{für } t \in]\frac{1}{2}\,t_1,\ t_1] \\ -\frac{1}{8} r\,t_1^3 + \frac{1}{4} r\,t_1^2\,t, & \text{für } t > t_1\,. \end{cases} \end{aligned} \tag{4.33}$$

Ist mit einer Dreiecksbeschleunigung eine Geschwindigkeitsdifferenz Δv zu überwinden, berechnen sich die benötigte Fahrdauer und -strecke auf dem Dreieckprofil zu

$$\begin{aligned} t_1 &= \sqrt{\frac{4\,|\Delta v|}{r}} \\ \Delta s_1 &= \frac{1}{8} r\,t_1^3 \\ \Rightarrow \Delta s_1 &= \sqrt{\left|\frac{\Delta v}{r}\right|}\,|\Delta v|\ . \end{aligned} \tag{4.34}$$

Mit einer Startgeschwindigkeit v_{start} ergibt sich die benötigte Zeit t_2 zur Überwindung von Δs mit der Geschwindigkeitsänderung Δv

$$\begin{aligned} \Delta s &= s(t_2) = t_2\ (v_{start} + \Delta v) - \Delta v \sqrt{\frac{\Delta v}{r}} \\ \Rightarrow\ t_2 &= \frac{\Delta v \sqrt{\frac{\Delta v}{r}} + \Delta s}{v_{start} + \Delta v} \end{aligned} \tag{4.35}$$

Dreicksprofil für verzögerte Beschleunigung

Das Dreicksprofil wird durch

$$a\left(t\right) := \begin{cases} 0, & \text{für } t \in [0,\ t_1] \\ r\ \left(t - t_1\right), & \text{für } t \in]t_1,\ t_1 + \frac{\Delta t_2}{2}] \\ -r\,t + r\ \left(t_1 + \Delta t_2\right), & \text{für } t \in]t_1 + \frac{\Delta t_2}{2},\ t_2] \end{cases} \tag{4.36}$$

Mit diesem Profil wird ebenfalls innerhalb des Bahnabschnitts Δs die Geschwindigkeitsdifferenz Δv mit dem festen Ruck r überwunden. Es wird verwendet, wenn nicht direkt beschleunigt werden darf, da sonst die dynamischen Randbedingungen verletzt werden. Weiter unten in Abschnitt 4.3.2 werden solche Profile wieder aufgegriffen.

Auf die ausführliche Beschreibung der analog herzuleitenden v- und s-Funktionen wird hier verzichtet.

Ist mit einer Dreiecksbeschleunigung eine Geschwindigkeitsdifferenz Δv zu überwinden, berechnen sich die benötigte Fahrdauer und -strecke auf dem Dreieckprofil zu

$$\begin{aligned} t_1 &= \frac{\Delta s - \left(2\,v_{start} + \Delta v\right)\ \sqrt{\frac{\Delta v}{r}}}{v_{start}} \\ \Delta t_2 &= \sqrt{\frac{4\,\Delta v}{r}}\,. \end{aligned} \tag{4.37}$$

Dreicksprofil für verzögertes Abbremsen

Soll eine Verzögerung stattfinden, wird der Beschleunigungsverlauf des Dreicksprofil für sofortiges Beschleunigen auf der t-Achse bei $\frac{t_2}{2}$ punktgespiegelt und v_0 wird die Zielgeschwindigkeit zugeordnet. Dies hat zur Konsequenz, dass im Dreieckprofil für sofortiges Abbremsen die höhere Anfangsgeschwindigkeit möglichst lange gehalten und die Fahrzeit dadurch verringert wird. Das Dreicksprofil für sofortiges Abbremsen berechnet sich analog zu den vorgestellten Profilen. Folgend wird nur das Dreicksprofil für verzögertes Abbremsen vorgestellt.

Analog zum Profil für verzögertes Abbremsen wird dieses Zeitprofil durch

$$a\left(t\right) := \begin{cases} 0, & \text{für } t \in [0,\ t_1] \\ -r\ \left(t - t_1\right), & \text{für } t \in]t_1,\ t_1 + \frac{\Delta t_2}{2}] \\ r\,t - r\ \left(t_1 + \Delta t_2\right), & \text{für } t \in]t_1 + \frac{\Delta t_2}{2},\ t_2] \end{cases} \tag{4.38}$$

definiert. Das Geschwindigkeitsprofil mit der Startgeschwindigkeit v_{start} berechnet sich zu

$$v(t) := \int_0^t a(\tau)\, d\tau = \begin{cases} v_{start}, & \text{für } t \in [0,\ t_1] \\ -\frac{1}{\left(2\, v_{start}^2\right)} \left(2\, r\ \left(2\, v_{start} - \Delta v\right)\ \left(t\, v_{start} - \Delta s\right)\ \sqrt{\frac{\Delta v}{r}}\right. & \\ \left. +r\ \left(t^2\, v_{start}^2 - 2\, t\, v_{start}\, \Delta s + \Delta s^2\right)\right) & \\ -2\, v_{start}^3 + \Delta v\ \left(4\, v_{start}^2 - 4\, v_{start}\, \Delta v + \Delta v^2\right)), & \text{für } t\ \in]t_1,\ t_1 + \frac{\Delta t_2}{2}] \\ -\frac{1}{\left(2\, v_{start}^2\right)} \left(2\, r\, \Delta v\ \left(t\, v_{start} - \Delta s\right)\ \sqrt{\frac{\Delta v}{r}}\right. & \\ \left. -r\ \left(t^2\, v_{start}^2 - 2\, t\, v_{start}\, \Delta s + \Delta s^2\right) - 2\, v_{start}^3\right. & \\ \left. +\Delta v\ \left(2\, v_{start}^2 - \Delta v^2\right)\right), & \text{für } t\ \in]t_1 + \frac{\Delta t_2}{2},\ t_2] \end{cases} \tag{4.39}$$

und die Bahnstrecke ist hier nicht weiter ausgeführt.

Ist mit einer Dreiecksbeschleunigung eine Geschwindigkeitsdifferenz Δv zu überwinden, berechnen sich die benötigte Fahrdauer und -strecke auf dem Dreieckprofil zu

$$\begin{aligned} t_1 &= \frac{\Delta s - \left(2\, v_{start} - \Delta v\right)\ \sqrt{\frac{\Delta v}{r}}}{v_{start}} \\ \Delta t_2 &= \sqrt{\frac{4\, \Delta v}{r}}\,. \end{aligned} \tag{4.40}$$

Trapezprofil für sofortige Beschleunigung

Je nach Länge des Bahnabschnitts Δs und der zu überwindenden Geschwindigkeitsdifferenz Δv kann die Maximalbeschleunigung des Dreicksprofils einen vorgegeben Grenzwert a_{max} überschreiten. In diesem Fall ist das Trapezprofil zu verwenden. Analog zum Dreieckprofil wird es durch

$$a(t) = \begin{cases} r\, t, & \text{für } t\ \in [0,\ t_1] \\ a_{max}, & \text{für } t\ \in]t_1,\ t_2 - t_1] \\ -r\, t + r\, t_2, & \text{für} t\ \in]t_2 - t_1,\ t_2] \\ 0, & \text{für } t > t_2 \end{cases} \tag{4.41}$$

$$\Rightarrow a_{max} = a(t_1) \Leftrightarrow t_1 = \frac{1}{r}\, a_{max}$$

definiert. Nach der anfänglichen Beschleunigung mit dem Maximalwert a_{max} wird die Geschwindigkeit konstant gehalten, nachdem Δv überwunden wurde.

Das Geschwindigkeitsprofil für die Startgeschwindigkeit v_{start} berechnet sich zu

$$v(t) = \begin{cases} \frac{1}{2} r\, t^2 + v_{start}, & \text{für } t \in [0,\ t_1] \\ a_{max}\, t - \frac{a_{max}^2 - 2\, r\, v_{start}}{2\, r}, & \text{für} t \in]t_1,\ t_2 - t_1] \\ -\frac{r\, t^2}{2} + \frac{t\left(a_{max}^2 + r\, \Delta v\right)}{a_{max}} - \frac{a_{max}^4 - 2\, a_{max}^2\, r\, v_{start} + r^2\, \Delta v^2}{2\, a_{max}^2\, r}, & \text{für } t \in]t_2 - t_1,\ t_2] \\ v_{start} + \Delta v, & \text{für } t > t_2 \end{cases}$$

$$\Rightarrow\ \Delta v = v(t_2)\ \Leftrightarrow\ t_2 = \frac{|\Delta v|}{a_{max}} + \frac{a_{max}}{r} \tag{4.42}$$

Für die Bahn mit Startstrecke $s_{start} = 0$ gilt

$$s(t) = \begin{cases} \frac{1}{6} r\, t^3 + t\, v_{start}, & \text{für } t \in [0,\ t_1[\\ \frac{1}{2} a_{max}\, t^2 + \frac{1}{2\, r} t\ (2\, r\, v_{start} - a_{max}^2) + \frac{1}{6\, r^2} a_{max}^3, & \text{für } t \in]t_1,\ t_2 - t_1[\\ -\frac{r\, t^3}{6} + \frac{t^2\left(a_{max}^2 + r\, \Delta v\right)}{2\, a_{max}} - \frac{t\left(a_{max}^4 - 2\, a_{max}^2\, r\, v_{start} + r^2\, \Delta v^2\right)}{2\, a_{max}^2\, r} & \\ + \frac{a_{max}^6 + r^3\, \Delta v^3}{6\, a_{max}^3\, r^2}, & \text{für } t \in]t_2 - t_1,\ t_2[\\ t\ (v_{start} + \Delta v) - \frac{\Delta v\left(a_{max}^2 + r\, \Delta v\right)}{2\, a_{max}\, r}, & \text{für } t > t_2 \end{cases}$$

$$\Rightarrow\ \text{für } v_{start} = 0:\ \Delta s_1 = s(t_2) = \frac{1}{2} a_{max}\, t_2^2 - \frac{1}{2\, r}\, a_{max}^2\, t_2 \tag{4.43}$$

Ist mit einer Trapezbeschleunigung eine Geschwindigkeitsdifferenz Δv zu überwinden, berechnen sich die benötigte Fahrtdauer und -strecke auf dem Trapezprofil mit $v_{start} = 0$ aus (4.42) und (4.42) zu:

$$\begin{aligned} t_2 &= \frac{|\Delta v|}{a_{max}} + \frac{a_{max}}{r} \\ \Delta s_1 &= \frac{|\Delta v|^2}{2\, a_{max}} + \frac{a_{max}\ |\Delta v|}{2\, r} \end{aligned} \tag{4.44}$$

Mit einer Startgeschwindigkeit v_{start} ergibt sich die benötigte Zeit t_3 zur Überwindung von Δs mit der Geschwindigkeitsänderung Δv und der Startgeschwindigkeit v_{start}

$$\begin{aligned} \Delta s &= s(t_3) = -\frac{a_{max}\, \Delta v}{2\, r} - \frac{\Delta v^2}{2\, a_{max}} + t_3\, v_{start} + t_3\, \Delta v \\ \Rightarrow\ t_3 &= \frac{a_{max}^2\, \Delta v + 2\, a_{max}\, r\, \Delta s + r\, \Delta v^2}{2\, a_{max}\, r\ (v_{start} + \Delta v)} \end{aligned} \tag{4.45}$$

Analog zu zu diesem Profil werden die drei übrigen Trapezprofile mit Verzögerung und Abbremsen hergeleitet. Es ergeben sich insgesamt acht Beschleunigungsprofile anhand folgender Kriterien

- abbremsen oder beschleunigen,
- sofortige oder verzögerte Beschleunigung,
- mit bzw. ohne Erreichen der maximalen Beschleunigung a_{max} (Trapez bzw. Dreieck)

Der Zusammenhang zwischen benötigter Bahnstrecke für eine gewünschte Geschwindigkeitsänderung Δv wie in den Gleichungen (4.46) und (4.45) wird im nächsten Abschnitt zur Wahl des Beschleunigungsprofils für ein Zeitsegment wieder aufgegriffen.

Einfluss der Startgeschwindigkeit

Nach dem Durchlauf des Dreieck-Beschleunigungsprofils am Zeitpunkt t_1 bzw. beim Trapezprofil an t_2 ist die Zielgeschwindigkeit bereits erreicht. Diese wird gehalten bis der aktuelle Steckenabschitt Δs komplett durchfahren wurde und t_2 bzw. t_3 erreicht wurde.
Falls zu Beginn des Beschleunigungsprofils bereits eine Geschwindigkeit $v_{Start} \neq 0$ herrscht, muss diese während Δs_2 berücksichtigt werden und verkürzt somit die Fahrzeit t_2 bzw. t_3. Dies wird prinzipiell durch folgenden Zusammenhang ausgedrückt:

$$\begin{aligned} \Delta s_2 &= \Delta s - \Delta s_1 - v_0\, t_1 \\ t_2 &= \frac{\Delta s_2}{|\Delta v| + v_0} + t_1 \\ \text{mit } v_0 &= \min\{v_i, v_{i+1}\} \end{aligned}$$

analog für Trapezprofil:

$$\begin{aligned} \Delta s_2 &= \Delta s - \Delta s_1 - v_0\, t_2 \\ t_3 &= \frac{\Delta s_2}{|\Delta v| + v_0} + t_2 \end{aligned} \tag{4.46}$$

4.3.2 Zeitoptimales Bahnprofil mit Randbedingungen

Im vorherigen Abschnitt wurde für den durch (s_i, v_i)-Punkte gegebene Geschwindigkeitsbeschreibung der Bahn im kartesischen Raum ein ruckfreies Beschleunigungsprofil erstellt. Auf die Erstellung dieses Geschwindigkeitsverlaufs wird in diesem Abschnitt eingegangen.

Der Geschwindigkeitsverlauf kann entweder manuell durch den Bediener angegeben oder aus Vorgaben von Randbedingungen automatisch berechnet werden. In beiden Fällen muss überprüft werden, ob die gewünschte Bewegung auf der gegebenen geometrischen Bahn durch die Roboterdynamik realisierbar ist. Wird z. B. die Grenze des Antriebdrehmoments oder der maximalen Gelenkgeschwindigkeit an einer Bahnstelle überschritten, so muss die zugehörige Bahngeschwindigkeit reduziert werden. Um ein realisierbares Geschwindigkeitsprofil zu erhalten, wird zunächst ein Grenzgeschwindigkeitsprofil erstellt. Es drückt die maximal stationäre Bahngeschwindigkeit unter Einhaltung aller Randbedingungen der Roboterdynamik aus. Da dieses Grenzprofil sprunghaft ist, wird es anschließend analysiert und der konkrete Geschwindigkeitsverlauf mit (s_i, v_i)-Punkten abgeleitet.

Grenzgeschwindigkeitsprofil

Das Grenzgeschwindigkeitsprofil ordnet jedem Bahnpunkt eine bahn- und dynamikspezifische maximale Geschwindigkeit zu. Die Geschwindigkeit $v = \dot{s}$ ist so zu wählen, dass folgende Bedingungen eingehalten werden:

$$\begin{aligned} \text{Drehmoment :} \quad & m_{ai} \leq |m_{ai,max}| \\ \text{Gelenkgeschwindigkeit :} \quad & \omega_i \leq |\omega_{i,max}| \\ \text{Gelenkbeschleunigung :} \quad & \dot{\omega}_i \leq |\dot{\omega}_{i,max}| \ . \end{aligned} \tag{4.47}$$

Ausgehend von den Bewegungsgleichungen aus Gleichung (3.37)

$$m_{ai} = I_i \, \dot{\omega}_i + \sum_j a_{ij} \, \dot{\omega}_j + \sum_j c_{ij} \, \omega_i \, \omega_j + \sum_j z_{ij} \, \omega_j^2 + k_{ri} \, \omega_i + h_i \, sign \, (\omega_i) + m_{sti}$$

muss die Bahngeschwindigkeit $\dot{s}$ unter Berücksichtigung von (4.47) berechnet werden.

Bei bekanntem maximalen Drehmoment der Gelenke $m_{ai,max}$ ist ein direktes Auflösen der Bewegungsgleichung (3.37) nach einer Gelenkbeschleunigung $\dot{\omega}_i$ bei bekannter Bahnposition $\vec{\theta}\,(s)$ nicht eindeutig und liefert nur einen Zusammenhang der Beschleunigung und Geschwindigkeiten aller Gelenke. Es ergibt sich eine Gleichung mit $2\,n$ Freiheitsgraden: Die Gelenkpositionen θ_i hängen eindeutig von dem Bahnparameter s ab, wobei die Gelenkgeschwindigkeiten ω_i und -beschleunigungen $\dot{\omega}_i$ wählbar sind. Constantinescu und Croft [Croft 00] betrachten zusätzlich die Begrenzung des Gelenkrucks $\dot{m}_{ai}$, indem die Bewegungsgleichung nach der Zeit diffenrenziert wird. Dadurch entsteht ein System mit $3\,n$ Freiheitsgraden. Da in dieser Arbeit der Bahnruck beschränkt wird, kann der Gelenkruck vernachlässigt werden.

Da als weitere Randbedingung die Bewegung auf der vorgegebenen Bahn $\vec{X} = \vec{X}\,(s)$ stattfinden muss, reduziert diese Parametrierung über den Bahnparameter s die Ordnung des Systems auf zwei. Für eine eindeutige Lösung lassen sich die Gelenkgrößen als Funktion des Bahnparameters darstellen:

$$\begin{aligned} \theta_i \, (t) &= \theta_i \, (s) \\ \omega_i \, (t) &= \dot{\theta}_i \, (t) = \theta_i' \, \dot{s} \\ \dot{\omega}_i \, (t) &= \ddot{\theta}_i \, (t) = \theta_i'' \, \dot{s}^2 + \theta_i' \, \ddot{s} \, , \end{aligned} \tag{4.48}$$

wobei $(.)'$ die Ableitung der Gelenkstellungen nach dem Ort im Gelenkwinkelraum beschreibt. Die Ableitungen lassen sich aus dem Verlauf der geometrischen Bahn berechnen.

Die parametrierte Bewegungsgleichung berechnet sich zu

$$m_{ai} = A_i \, \ddot{s} + B_i \, \dot{s}^2 + C_i \, \dot{s} + D_i \tag{4.49}$$

mit

$$\begin{aligned} A_i &= (I_{ai} + I_{gi})\ \theta'_i + \sum_j a_{ij}\,\theta'_j \\ B_i &= (I_{ai} + I_{gi})\ \theta''_i + \sum_j \left(a_{ij}\,\theta''_j + c_{ij}\,\theta'_i\,\theta'_j + z_{ij}\,{\theta'_j}^2\right) \\ C_i &= (k_{rgi} + k_{rai})\ \theta'_i \\ D_i &= m_{sti} + h_{ai} \end{aligned} \tag{4.50}$$

Mit Gleichung (4.49) lässt sich (4.47) als

$$\begin{aligned} m_{ai} &\leq |m_{ai,max}| \\ \theta'_i\,\dot{s} &\leq |\omega_{i,max}| \\ \theta''_i\,\dot{s}^2 + \theta'_i\,\ddot{s} &\leq |\dot{\omega}_{i,max}| \end{aligned} \tag{4.51}$$

darstellen und die Bahngeschwindigkeit und -beschleunigung berechnen sich angenähert aus den Bedingungen

$$\begin{aligned} \dot{s}_{1,i} &= \begin{cases} \frac{1}{|\theta'_i|}\omega_{i,max}, & \text{für } \theta'_i \neq 0 \\ \infty, & \text{sonst} \end{cases} \\ \dot{s}_{2,i} &= \begin{cases} \sqrt{\frac{\dot{\omega}_{i,max}}{\theta''_i}}, & \text{für } \theta''_i \neq 0 \\ \infty, & \text{sonst} \end{cases} \\ \ddot{s}_{1,i} &= \frac{\dot{\omega}_{i,max}}{|\theta'_i|} \end{aligned} \tag{4.52}$$

zu

$$\begin{aligned} \dot{s}_i &= \min\{\dot{s}_{1,1}, \dot{s}_{1,2}, .., \dot{s}_{2,1}, \dot{s}_{2,2}, ..\} \\ \ddot{s}_i &= \ddot{s}_{1,i}\,. \end{aligned} \tag{4.53}$$

$\dot{\omega}_{i,max}$ berechnet sich aus der Bewegungsgleichung für die jeweilige Gelenkstellung $\vec{\theta}(s)$, indem für m_{ai} das maximale konstante Drehmoment $m_{ai,max}$ der Antriebe angenommen wird. Die maximale Gelenkgeschwindigkeit $\omega_{i,max}$ ist unabhängig von der Gelenkstellung $\vec{\theta}(s)$ konstant.

Gleichung (4.51) begrenzt den Zustandsraum $s - \dot{s} - \ddot{s}$, in dem sich der Roboter befinden kann. Zu jedem Bahnparameter s und jeder Geschwindigkeit $\dot{s}$ wird die Grenze der Beschleunigung $\ddot{s}$ ermittelt. Abbildung 4.26 zeigt eine Grenze dieses Zustandsraums.
Die zeitoptimale Geschwindigkeit liegt unterhalb dieser Grenzfläche. Der beste, exakte Geschwindkeitsverlauf verwendet nicht die im vorherigen Abschnitt vorgestellten, einfachen Beschleunigungsprofile sondern kann nur numerisch abgespeichert werden. Das Ziel ist jedoch eine einfache Beschreibung des Geschwindigkeitsprofils mit geringem Speicherbedarf und wenigen s_i, v_i-Stützpunkten.

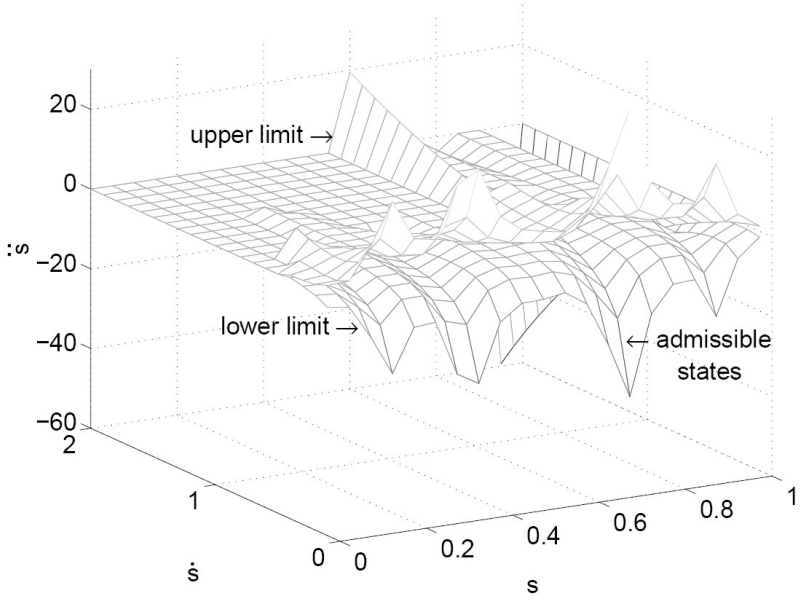


Abbildung 4.26: Grenzgeschwindigkeitsfläche im $s - \dot{s} - \ddot{s}$-Raum, aus [Croft 00] S. 3

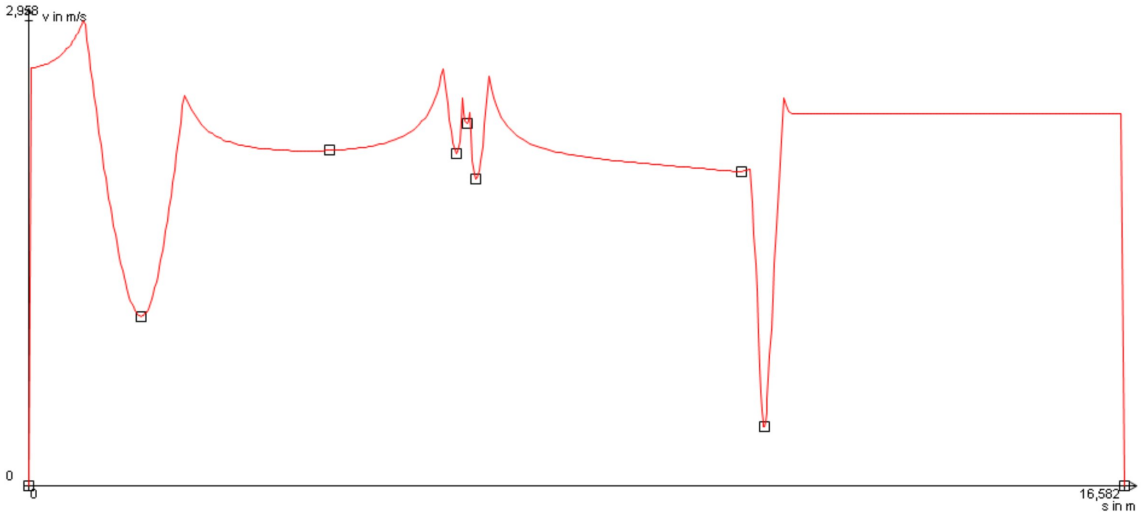


Abbildung 4.27: Grenzgeschwindigkeitsprofil $\dot{s}_{max}$ für $\ddot{s} = 0$ (aus Simulation)

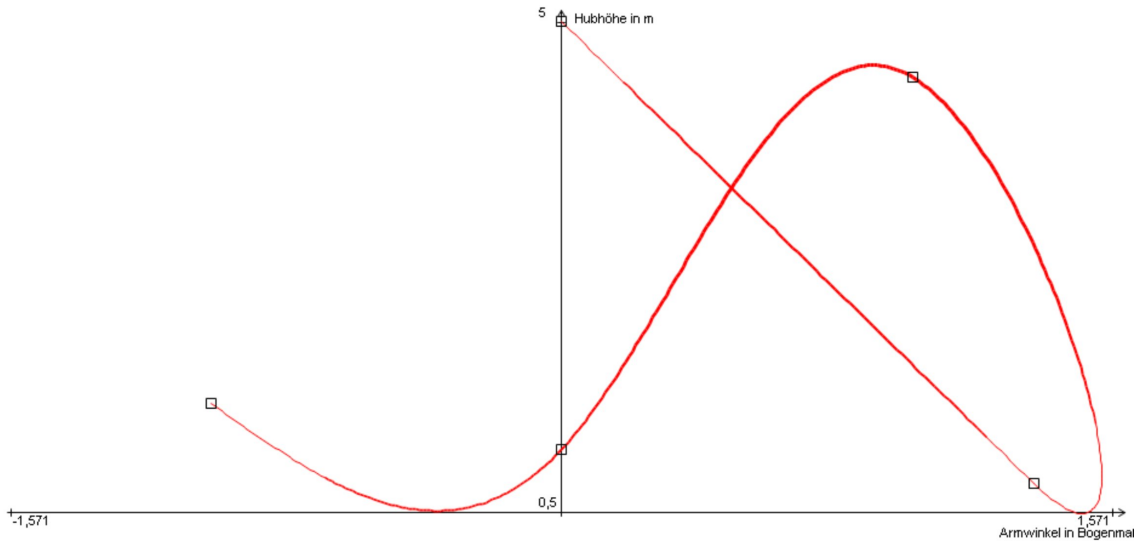


Abbildung 4.28: Geometrische Bahn im Gelenkwinkelraum als Ausgangspunkt für das Grenzgeschwindigkeitsprofil (aus Simulation)

Shiller, Chang und Wong [Wong 96] kommen durch eine andere Darstellung der Bewegungsglechung zu der Randbedingung

$$\ddot{s}_{min}\left(s,\dot{s}\right) \leq \ddot{s} \leq \ddot{s}_{max}\left(s,\dot{s}\right) , \tag{4.54}$$

wobei ab einer gewissen Bahngeschwindigkeit $\dot{s} \geq \dot{s}_{max}$ keine zulässige und sinnvolle Beschleunigung $\ddot{s}$ existiert. Der Wert $\dot{s}_{max}$ definiert die Grenzgeschwindigkeitskurve mit der Vernachlässigung der Bahnbeschleunigung $\ddot{s} = 0$.
Im Gegensatz zu [Wong 96], [Olomski 89] wird in dieser Arbeit die Beschleunigung $\ddot{s}$ nicht komplett vernachlässigt, da dies zu nicht akzeptablen Ungenauigkeiten und Verletzungen der Drehmomentgrenzen führen kann. Die Gleichung (4.53) liefert die angenäherten Bedingungen zur Erstellung der Grenzkurve. Die maximale Beschleunigung $\ddot{s}$ fließt nicht in die Grenzgeschwindigkeit, sondern in die Gestaltung der Beschleunigungsprofilen aus Abschnitt 4.3.1 mit ein: Überschreitet die Maximalbeschleunigung eines Dreiecksprofils a_{max} die Grenzbeschleunigung $\ddot{s}$, so muss ein Trapezprofil gewählt werden. Nach der Auswahl des Beschleunigungsprofils kann $\ddot{s}$ für die weitere Berechnung vernachlässigt werden: $\ddot{s} = 0$. Das Grenzgeschwindigkeitsprofil ist als $s - \dot{s}_{max}$-Diagramm mit $\ddot{s} = 0$ in Abbildung 4.27 dargestellt.
Die Maxima des Grenzgeschwindigkeitsprofils sind an Bahnstellen s zu finden, an denen alle Antriebe gleichzeitig fahren können und sich die Geschwindigkeiten addieren. Minima treten dort auf, wo eine Änderung der Bahnrichtung zu einer Gelenkbeschleunigung führt. Die Gelenkbeschleunigung wird durch das maximale Drehmoment des zugehörigen Antriebs begrenzt. Abbildung 4.28 zeigt die geometrische Bahn, die dem Grenzprofil zugrunde liegt.

Interpoliertes Geschwindigkeitsprofil

Das Grenzgeschwindigkeitsprofil zeigt die maximal mögliche Bahngeschwindigkeit. Es ist jedoch wegen den Geschwindigkeitssprüngen und Verletzungen der dynamischen Randbedingungen nicht abfahrbar. Gesucht ist ein Geschwindigkeitsverlauf,

der stets unterhalb der Grenzgeschwindigkeit verläuft und die Randbedingungen einhält.
Die triviale Lösung $v = \min\{\dot{s}_{max}\}$ ist nicht ausreichend, da das Profil ein zeitoptimales Fahren ermöglichen soll. Stattdessen muss abschnittsweise auf höhere Bahngeschwindigkeiten beschleunigt werden, soweit dies möglich ist. Dabei muss die Randbedingung der Ruckbegrenzung beachtet werden,

$$r = \dddot{s} \leq r_{max} \,. \tag{4.55}$$

Diese Randbedingung wurde bereits bei der Gestaltung der Beschleunigungsprofile verwendet und wird hier wieder aufgegriffen.

Die Bahnabschnitte und Geschwindigkeiten werden wie folgt ermittelt:

1. alle Minima (s_i, v_i) im Grenzgeschwindigkeitsprofil $\dot{s}_{min}$ suchen
2. überprüfen, ob alle gefundenen v_i mit vorgegebenem Ruck erreichbar sind, nicht erreichbare verwerfen
3. Zwischenpunkte $s_{neu,i}$ zwischen den restlichen s_i einfügen
4. Geschwindigkeit der Zwischenpunkte unter Randbedingungen erhöhen
5. Punkte, deren Geschwindigkeit zwischen ihren zwei benachbarten Punkten liegen und die nicht auf einem Grenzgeschwindigkeitskurven-Minimum (s_i, v_i) liegen, entfernen

Die verbleibenden Minima- und Zwischenpunkte begrenzen je einen Beschleunigungsabschnitt und liefern die Start- und Zielgeschwindigkeit für diesen Abschnitt. Die einzelnen Schritte im Detail:

1. Die Suche nach den Minima in Schritt 1 wird numerisch durchgeführt.
   Sei n_{grenz} die Anzahl der Punkte des numerischen Grenzgeschwindigkeitsprofils. Die Iteration über alle Geschwindigkeitspunkte liegt in $\mathcal{O}\,(n_{grenz})$.
2. Die Gestalt des Beschleunigungsprofils hat einen direkten Einfluss auf die Position der Geschwindigkeitsstützpunkte. Hier muss zunächst überprüft werden, ob die maximale Beschleunigung im Dreicksprofil $a_{\Delta,max}$ unterhalb der maximal erlaubten Beschleunigung $\ddot{s}$ aus der Gleichung (4.53) liegt,

   $$\ddot{s} \geq a_{\Delta,max} = \sqrt{r\,\Delta v} \,. \tag{4.56}$$

   Ist dies der Fall, dann geht die Ruckbedingung mit dem Kriterium aus Gleichung (4.34) in Schritt 2 ein, wobei eine Startgeschwindigkeit $v_i \geq 0$ für die Beschleunigungsprofile berücksichtigt wird,

   $$\hat{s}_{i+1} - \hat{s}_i \geq s_{min} = \sqrt{\frac{|\hat{v}_{i+1} - \hat{v}_i|}{r}}\;(\hat{v}_{i+1} + \hat{v}_i) \;. \tag{4.57}$$

Ist die maximale Beschleunigung des Dreiecksprofils zu hoch, so wird das Trapezprofil und das aus Gleichung (4.44) abgeleitete Kriterium angewendet,

$$\hat{s}_{i+1} - \hat{s}_i \geq s_{min} = \frac{\left|\hat{v}_{i+1}^2 - \hat{v}_i^2\right|}{2\,a_{max}} + \frac{a_{max}}{2\,r}\,\left|\hat{v}_{i+1} + \hat{v}_i\right| . \tag{4.58}$$

Es werden alle Minima i mit einer zu niedrigen oder zu hohen Geschwindigkeit, so dass mit der zur Verfügung stehenden Bahnstrecke Δs zum Nachbarpunkt und der ruckbegrenzte Beschleunigung der Nachbarpunkt (s_{i+1}, v_{i+1}) nicht erreicht werden kann, für die folgenden Schritte verworfen.
Der Aufwand dieses Schritts liegt mit den $n_{minima} < n_{grenz}$ Punkten in $\mathcal{O}\left(n_{minima}\right)$.

3. Um zwischen den übrigen Grenzprofilminima die Geschwindigkeit zu erhöhen und das Grenzprofil besser ausnutzen zu können, werden Zwischenpunkte eingefügt. Die Zwischenpunkte für den Schritt 3 liegen jeweils in der Mitte zwischen 2 benachbarten Punkten,

$$\left(\hat{s}_{neu}, \hat{v}_{neu}\right) = \left(\frac{1}{2}\,\left(\hat{s}_i + \hat{s}_{i+1}\right), 0\right) . \tag{4.59}$$

Diese Lösung ist effizient zu implementieren. Olomski [Olomski 89] skizziert eine Heuristik mit einer iterativen Optimierung für die Plazierung der Zwischenpunkte, so dass die Grenzgeschwindigkeitskurve besser ausgenutzt werden kann. Dies bringt praktisch nur eine geringe Verbesserung.
Schritt 3 liegt in $\mathcal{O}\left(n_{minima}\right)$.

4. Im Schritt 4 sind die Zwischenpunkte nach oben zu verschieben. Hier gelten ähnliche Bedingungen wie in Schritt 2 für das Dreieckprofil,

$$\Delta s := s_{i+1} - s_{neu} = s_{neu} - s_i$$
$$\Delta s \geq \max\left\{\sqrt{\frac{\left|v_{i+1} - v_{neu}\right|}{r}}\,\left(v_{i+1} + v_{neu}\right), \sqrt{\frac{\left|v_{neu} - v_i\right|}{r}}\,\left(v_{neu} + v_i\right)\right\} \tag{4.60}$$

und für das Trapezprofil

$$\begin{aligned}\Delta s \geq \max\Big\{&\frac{\left|v_{i+1}^2 - v_{neu}^2\right|}{2\,a_{max}} + \frac{a_{max}}{2\,r}\,\left|v_{i+1} + v_{neu}\right|, \\ &\frac{\left|v_{neu}^2 - v_{i-1}^2\right|}{2\,a_{max}} + \frac{a_{max}}{2\,r}\,\left|v_{neu} + v_{i-1}\right|\Big\} .\end{aligned} \tag{4.61}$$

Die Zwischenpunkte werden iterativ mit maximal n_{it} Schritten in Richtung höherer Geschwindigkeit verschoben, bis diese Ruckbedingung nicht mehr erfüllt ist. Bei jeder Iteration wird erneut anhand Gleichung (4.56) geprüft, ob für die Beschleunigung das Dreieckprofil verwendet werden kann oder das Trapezprofil verwendet werden muss.
Schritt 4 liegt in $\mathcal{O}\left(n_{minima}\,n_{it}\right)$.

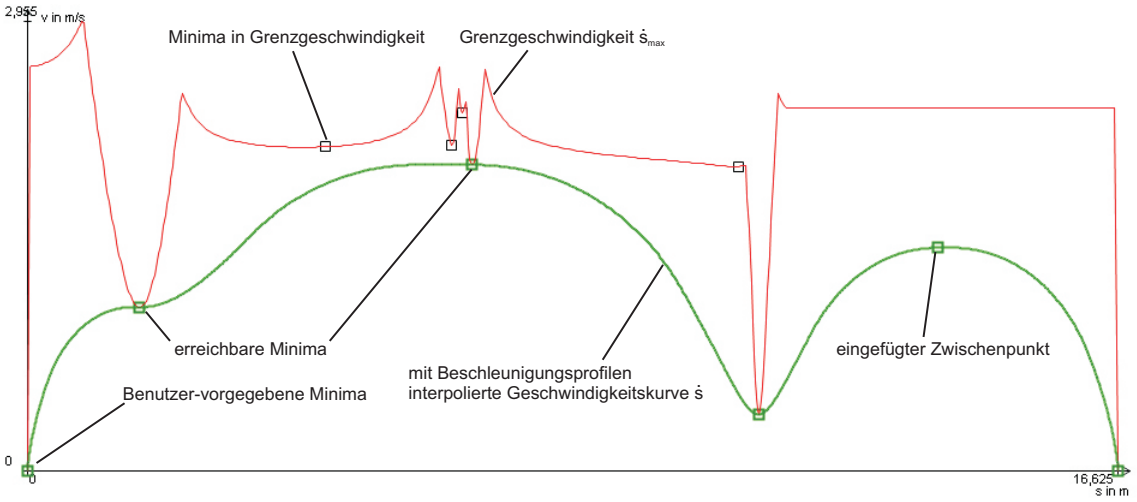


Abbildung 4.29: Interpoliertes Geschwindigkeitsprofil durch die Minima des Grenzgeschwindigkeitsprofils (aus Simulation)

5. Durch die Schritte 1 bis 4 können Zwischenpunkte entstehen, deren Geschwindigkeit zwischen ihren beiden Nachbarpunkten liegt. Da in allen Punkten $a = 0$ gilt, verschlechtern diese Punkte die Laufzeit und werden in Schritt 5 mit einem Aufwand in $\mathcal{O}\left(n_{minima}\right)$ entfernt.

Der Gesamtaufwand zur Erzeugung der (s, v)-Punkte liegt in $\mathcal{O}\left(n_{minima}\, n_{it} + n_{grenz}\right)$.

Abbildung 4.29 zeigt das Resultat der fünf Schritte und den interpolierten Geschwindigkeitsverlauf. Die Interpolation über die Geschwindigkeitsextrema erfordert die Beschleunigungsprofile aus Abschnitt 4.3.1. Die Lage eines Extremums und die seines Nachbarn ergeben die zu überwindende Geschwindigkeitsdifferenz und die zur Verfügung stehende Strecke. Diese Kriterien führen zur Auswahl des passenden Beschleunigungsprofils. Wird das Beschleunigungsprofil mit je n_a Punkten zwischen zwei benachbarten Geschwindigkeitsextrema numerisch berechnet, ergibt sich eine Laufzeitkomplexität von $\mathcal{O}\left(n_a\, n_{grenz}\right)$.

Diese Berechnung liefert zunächst nur eine Abbildung der Fahrzeit auf die Bahnbeschleunigung. Bahngeschwindigkeit und -strecke werden durch eine numerische Integration des Zeit-Beschleunigungsverlaufs in $\mathcal{O}\left(n_a\, n_{grenz}\right)$ berechnet. Nun kann der Bahnparameter der interpolierten Bahn mit der integrierten Bahnlänge der Beschleunigungsprofile und somit auch mit der Fahrzeit in Beziehung gebracht werden. Die Numerische Zuordnung der Bahnzeit zu der Bahnstrecke iteriert über alle interpolierten Bahnpunkte und führt eine Suche nach der Bahnstrecke in den Beschleunigungsprofilen durch.
Bei einer binären Suche ergibt sich der Aufwand $\mathcal{O}\left(m \cdot n \cdot \log\left(n_a\, n_{grenz}\right)\right)$.

Probleme der Approximation

Die Betrachtung der Grenzgeschwindigkeitskurve $\dot{s}_{max}$ mit der Vereinfachung $\ddot{s} = 0$ kann zu Verletzungen der Drehmomentgrenzen in der Nähe von Minima von $\dot{s}_{max}$

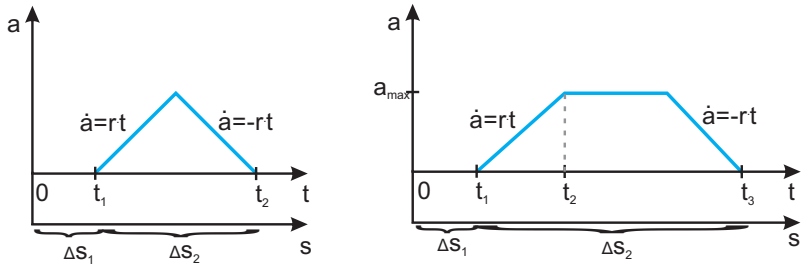


Abbildung 4.30: Neue Beschleunigungsprofile mit verzögerter Beschleunigung gegen die Verletzung von Drehmomentgrenzen bei Minima der Grenzkurve

führen. Denn nach den Minima wird beschleunigt, um die höhere Geschwindigkeit des folgenden (s_i, v_i)-Stützpunkts zu erreichen.
Olomski [Olomski 89] fügt je einen weiteren Stützpunkt $(s_i + \epsilon, v_i)$ neben allen Minima i ein, um eine die Beschleunigung zu hinauszuögern. Auf die genaue Lage dieser Punkte wird nicht eingegangen.
In dieser Arbeit wird ein neuer Ansatz verwendet, der auf die Einführung weiterer Punkte verzichtet. Die Beschleunigungsprofile aus Abschnitt 4.3.1 werden durch die Profile aus Abbildung 4.30 ergänzt. Ist (s_i, v_i) ein Minimum mit $v_i > 0$, dann wird das neue Beschleunigungsprofil verwendet. Dadurch erhöht sich die Fahrzeit.
Drehmomentgrenzen können weiterhin verletzt werden, jedoch zeigte dieser Ansatz für alle untersuchten Bahnen gute Ergebnisse. Falls die Randbedingung des Drehmoments nicht eingehalten wird, wird die Geschwindigkeit v_{i+1} des folgenden Zwischenpunkts (s_{i+1}, v_{i+1}) verringert und somit die Fahrt mit konstanter Geschwindigkeit verlängert.
Die in Abschnitt 3.3 dargestellte maximale, effektive Dauerbelastung der Motoren $m_{eff,ai}$ ist eine wichtige Größe, die einzuhalten ist, um eine Überhitzung der Motoren im Dauerbetrieb zu verhindern. Das Effektivdrehmoment

$$m_{eff,ai} = \sqrt{\frac{1}{T} \int_0^T m_{ai}^2 (t)\ dt} \tag{4.62}$$

kann numerisch aus dem Geschwindkeitsprofil berechnet werden. Anschließend werden die Vorgaben der Maximalmomente angepasst und eine erneute Geschwindigkeitsberechnung durchgeführt. Somit ist ein zeitoptimales Fahren im Dauerbetrieb möglich.

Eigenschaften der Geschwindigkeitsberechnung

Der 2-stufige Mechanismus der Geschwindigkeitsberechnung mit Grenzprofil und anschließender Interpolation durch die hergeleiteten Beschleunigungsprofilen erlaubt einem Anwender, manuell einzelne Geschwindigkeiten zu verändern. Soll die Geschwindigkeit an einer bestimmten Bahnstelle $v(s_0) = v_0$ betragen, genügt es diesen Punkt in dem Grenzprofil zu ändern. Dieser Punkt stellt ein neues Minimum dar und die interpolierte Geschwindigkeit wird unter Einhaltung aller Randbedingungen automatisch berechnet. Ist die geänderte Geschwindigkeit zu hoch und würde Randbedingungen verletzten, so wird sie bei der Interpolation nicht beachtet.
Das Geschwindigkeitsprofil wird speicherschonend mit wenigen (s_i, v_i)-Punkten beschrieben und kann effizient abschittsweise online interpoliert werden. Auch eine

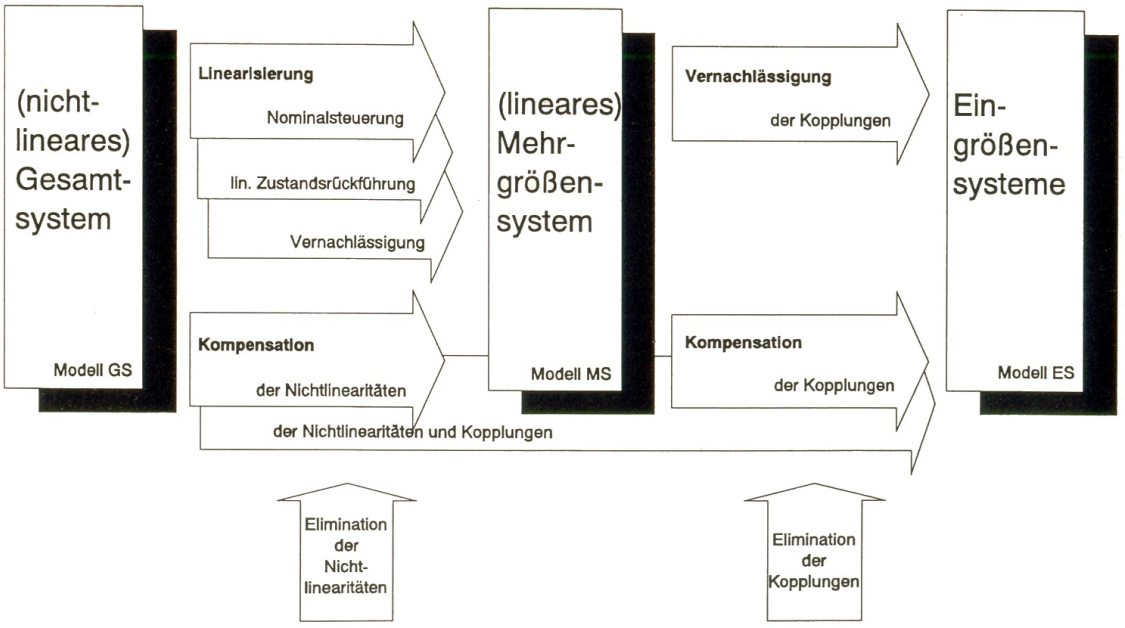


Abbildung 4.31: Reduktion der Komplexität in der Roboterregelung, aus [Truckenbrodt 94] S. 142

Änderung des Verlaufs während der Fahrt und die online-Neuberechung der Punkte ist mit gewisser Voraussicht möglich. Das Profil kann ebenso offline am PC optimiert und auf die Robotersteuerung übertragen werden.

4.4 Bahnregelung

Störungen, Verschleiß und Modellierungsungenauigkeiten führen zu Bahnfehlern bei der Fahrt über die vorgegebene Trajektorie. Während bei Aufgaben aus dem CAM-Bereich, Punktschweißen, Lackieren, usw. die exakte Fahrt über die Bahn notwendig ist, liegt das Hauptziel hier in einem überschwingungsfreien Erreichen der Endpunkte. Geringe Konturabweichungen der Bahn sind tolerierbar, wobei in Hinsicht auf zeitoptimales Fahren und Koordination des Roboters mit übrigen Aktuatoren Fehler in der zeitlichen Zuordnung gravierender sind.

4.4.1 Klassische Regelverfahren

Ein geringer Konturfehler ist oft das Hauptziel einer Regelung. Die Achsen sollen den Sollwerten sowohl in der Stellung als auch in Geschwindigkeit folgen. Roboterkinematiken mit den Wechselwirkungen der Gelenke führen zu einem komplexen und nicht-linearen Optimierungsproblem. Dieses Problem ist mit einer stufenweisen Vereinfachung lösbar. Wie in Abbildung 4.31 führen Linearisierung und Kompensation zu einem linearen, gekoppelten Modell. Durch Vernachlässigungen wird das gekoppelte System auf Einzelachsensysteme abgebildet, die einfacher zu modellieren sind. Diese Vereinfachungen sind zulässig, solange sie als Störungen in der Regelung aufgefasst werden können.

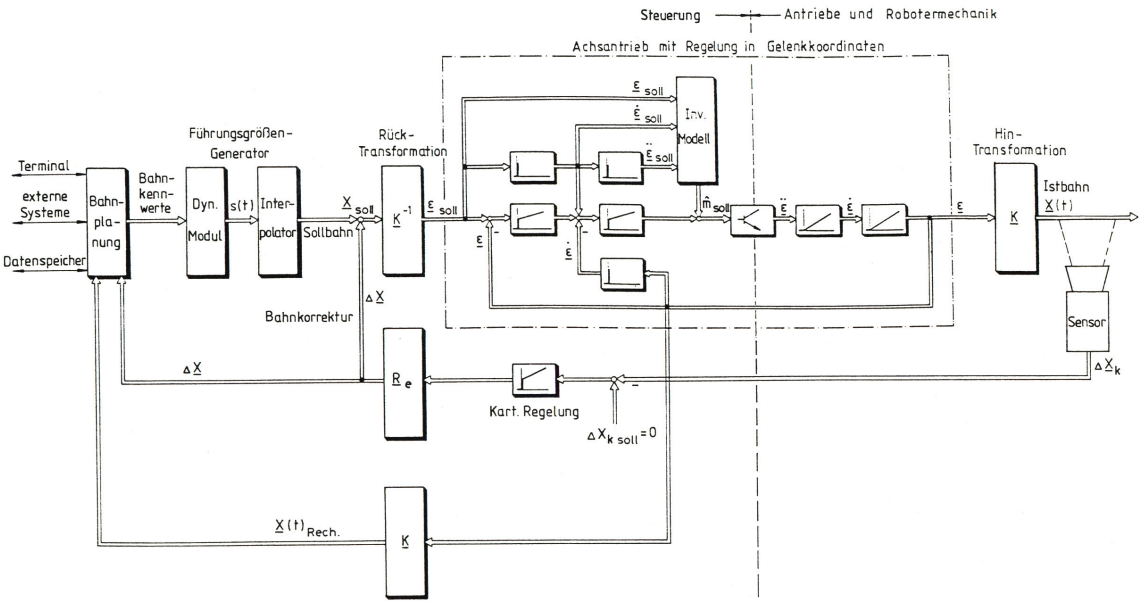


Abbildung 4.32: Regelungssystem eines Roboters mit inverser Dynamik zur Vorsteuerung und einem optischen Sensor zur Erfassung des kartesischen Bahnfehlers, aus [Olomski 89] S. 6

Das vereinfachte Modell kann nun mit einer Kaskadenregelung der entkoppelten, starren Achsen mit der Vernachlässigung der Elastizität durchgeführt werden und wirkt gegen dynamische Lagefehler. Die übliche Reaktionsverzögerung mechanischer Systemen muss beachtet werden und kann durch die Vorsteuerung und die Generierung der Führungsgrößen behoben werden. Eine verzögerungsfreie Folgeregelung ist mit einem direktem Ansprechen der inneren Regler (Gelenkbeschleunigung, Geschwindigkeit) möglich. In der Kaskadenregelung werden Störungen der inneren Regler kompensiert, bevor übergeordnete die Störung erkennen.

Da sich die Bahn und der Konturfehler aus der Überlagerung der Bewegung aller Gelenke ergibt, besteht die Forderung nach dem gleichen dynamischen Verhalten aller Regler, z. B. kann die am stärksten beanspruchte Achse die Regelparameter für die übrigen Regler vorgeben, um die Arbeit aller Regler in einem linearen Bereich zu fördern. Gleiche Reglerdynamik ist jedoch nicht hinreichend für einen geringen Konturfehler. Denn dieser ist auch bahn- und kinematikspezifisch.
Ein Lösungsansatz stellt eine Einzelachsregelung dar, welche die Gelenke verzögerungsfrei den Sollvorgaben folgen lässt. Es entsteht kein Schlepp- und Konturfehler auf der Bahnebene.

Ein nicht-lineares Regelmodell mit Kopplung muss variable Trägheitsmomente, Reibung und Gewicht beachten. Als Lösung kann ein Widerstandsdrehmoment auf das Antriebsdrehmoment mit der inversen Kinematik aufgeschaltet werden.

Eine neue Implementierung der Kaskaden-Achsenregelung wie in Abbildung 4.32 ist für den KHS-Roboter nicht nötig. Die Achsregelung wird als Standardaufgabe

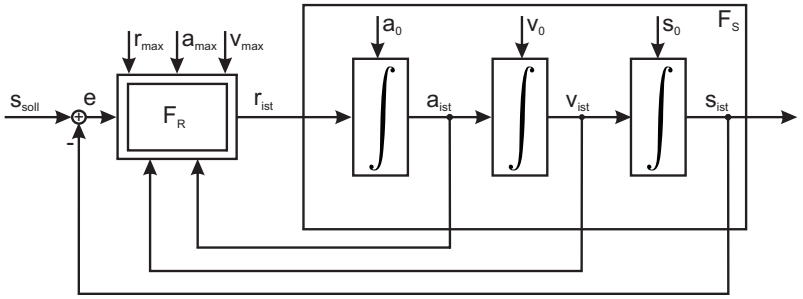


Abbildung 4.33: Blockschaltbild des Bahnreglers und der Regelstrecke

bereits durch die Steuerungshardware und die Systemsoftware erledigt. In den KHS-Systemen mit der darunter liegenden Einzelachssteuerung werden Achsen mit Fahrbefehlen (Zielstellung) und Parametern (Beschleunigungsrampe, maximale Drehzahl, etc.) beauftragt und regeln Störungen aus. Starke Störungen wie Überschreitungen des maximal zulässigen Motorstroms oder das Positionieren von Gelenken über Softwareendschalter hinaus werden dem Anwendungsprogamm mitgeteilt und erfordern ggf. den Eingriff des Bedieners.

4.4.2 Modellierung des zustandsbasierten Bahnreglers

Unter der Voraussetzung einer idealen Gelenkregelung und dem exakten Folgen der Solltrajektorie wird ein Mechanismus benötigt, mit dem eine berechnete Trajektorie abgefahren werden kann. Soll die Trajektorie nur teilweise oder mit einer spezifischen Geschwindigkeit abgefahren werden, muss eine zeitintensive Neuberechnung der Trajektorie stattfinden, obwohl die Bahnkontur gleich bleibt. Dies kann durch einen Bahnregler vermieden werden, der die Anfahrt einer beliebigen Zielposition auf der Bahn gestattet.

Die bisher vorgestellte Literatur geht auf die Implementierung dieses praktischen Aspekts nicht weiter ein. Olomski [Olomski 89] beschreibt ein Dynamikmodul, das eine ähnlichen Funktionalität wie die hier geforderte anbietet. Das Geschwindigkeitsprofil kann zwar beliebig gestaltet sein, berechnet aber alle Umschaltzeiten im Voraus und lässt keine Änderung der Geschwindigkeit während der Fahrt zu bzw. impliziert eine komplette Neuberechnung bei der Online-Anpassung der Sollgrößen. Deshalb wird im Folgenden ein Modell eines Bahnreglers aus Abbildung 4.33 hergeleitet, der Ruck, Beschleunigung, Geschwindigkeit und die Zielposition einregelt.

Die Regelung soll schnellstmöglich aber überschwingungsfrei alle Zielgrößen zugleich erreichen. Da im physischen Robotersystem wegen Filtern in der Antriebssteuerung und Sensorik eine erhebliche Totzeit existieren kann, führt dies zu Schwierigkeiten und Überschwingen der Regelung. Deshalb ist die Regelungsstrecke F_S nicht der physische Roboter sondern eine verzögerungsfreie, simulierte Bahnstrecke.

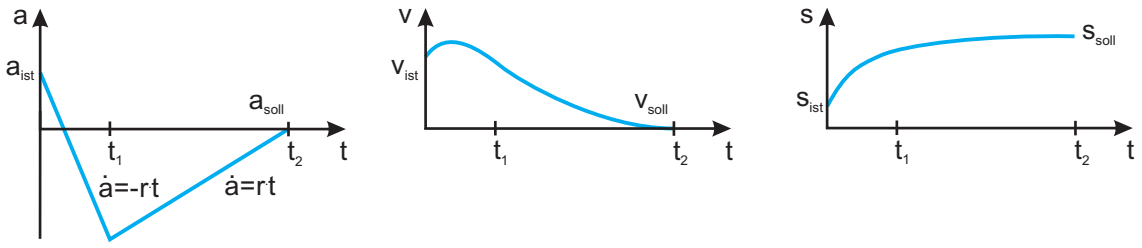


Abbildung 4.34: Dreieckförmiges Beschleunigungsprofil der Bahnregelung. Es soll den Bahnparameter s mit Ruck r_{max} auf die Ziellage s_{soll} überführen

Das Grundprinzip des Bahnreglers F_R basiert auf dem Algorithmus (4.1):

Listing 4.1: Bahnregler

```
wiederhole
  wenn sollGroessenGeaendert dann
    wiederhole
      berechneBremsweg
      wenn reststrecke > bremsweg dann
        regleGeschwindigkeit(v_max)
      sonst
        regleGeschwindigkeit(v_ziel)
      wenn_ende
      testeZielErreicht
    bis zielErreicht wiederhole_ende
  wenn_ende
wiederhole_ende
```

Der Regler soll abhängig von der restlichen Bahnstrecke auf die Zielposition der Bahn oder auf die maximale Geschwindigkeit einregeln.

Herleitung des Bremswegs und Wahl der Sollgeschwindigkeit

Der Bahnregler soll wie in Abbildung 4.34 ausgehend von einem Zustand mit beliebiger Bahnbeschleunigung a_{ist}, Bahngeschwindigkeit v_{ist} und Bahnstelle s_{ist} den Zielzustand unter Einhaltung des maximalen Bahnrucks r_{max} erreichen.

Aus Abbildung 4.34 ergibt sich analog zu den Beschleunigungsprofilen aus Abschnitt 4.3.1 der Verlauf der Regler-Zeitprofile

$$\begin{aligned}
a_1(t) &:= a_{ist} - r_{max}\, t \\
a_2(t) &:= a_{ist} - r_{max}\, t_1 + r_{max}\,(t - t_1) \\
v_1(t) &:= v_{ist} + \int_0^t a_1(\tau)\, d\tau \\
v_2(t) &:= v_1(t_1) + \int_{t1}^t a_2(\tau)\, d\tau \\
s_1(t) &:= s_{ist} + \int_0^t v_1(\tau)\, d\tau \\
s_2(t) &:= s_1(t_1) + \int_{t1}^t v_2(\tau)\, d\tau
\end{aligned} \tag{4.63}$$

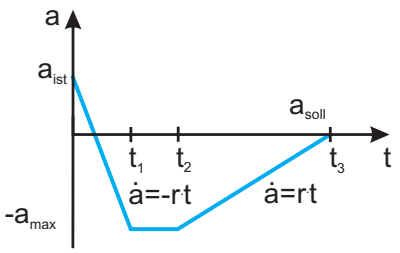


Abbildung 4.35: Trapezförmiges Beschleunigungsprofil der Bahnregelung bei Überschreitung der maximalen Beschleunigung

Um zu entscheiden, ob bei einem Beschleunigungs- oder Abbremsvorgang die vorgegebene, maximale Bahnbeschleunigung a_{max} überschritten wird, werden zunächst die Bahnzeiten t_1 und t_2 berechnet. Sie ergeben sich aus dem Gleichungssystem

$$\begin{aligned} a_2\left(t_2\right) &= a_{soll} \\ v_2\left(t_2\right) &= v_{soll} \end{aligned} \tag{4.64}$$

mit der Lösung

$$\begin{aligned} t_1 &= \frac{\sqrt{2}\left(\sqrt{a_{ist}^2 + a_{soll}^2 + 2\,r_{max}\,\left(v_{ist} - v_{soll}\right)} + \sqrt{2}\,a_{ist}\right)}{2\,r_{max}} \\ t_2 &= \frac{\sqrt{2}\,\sqrt{a_{ist}^2 + a_{soll}^2 + 2\,r_{max}\,\left(v_{ist} - v_{soll}\right)} + a_{ist} + a_{soll}}{r_{max}}\,. \end{aligned} \tag{4.65}$$

Die notwendige maximale Bahnbeschleunigung $a_{max,nötig}$ berechnet sich zu

$$\begin{aligned} \widehat{a} = a_1\left(t_1\right) &= -\frac{\sqrt{2}\,\sqrt{a_{ist}^2 + a_{soll}^2 + 2\,r_{max}\,\left(v_{ist} - v_{soll}\right)}}{2} \\ \Rightarrow\; a_{max,nötig} = \left|\widehat{a}\right| &= \frac{\sqrt{2}\,\sqrt{a_{ist}^2 + a_{soll}^2 + 2\,r_{max}\,\left|v_{ist} - v_{soll}\right|}}{2}\,. \end{aligned} \tag{4.66}$$

Falls die nötige Bahnbeschleunigung $a_{max,nötig}$ größer als die gewünschte Beschleunigung a_{max} ist, muss ein trapezartiges Beschleunigungsprofil aus Abbildung 4.35 verwendet werden.

Abhängig vom dem Typ des Beschleunigungsprofils und dem aktuellen Zustand des Reglers $(a_{ist}, v_{ist}, s_{ist})$ berechnet sich der Bremsweg s_b bzw. der nötwendige Weg zum ruckbegrenzten Erreichen des Zielzustands $(a_{soll}, v_{soll}, s_{soll})$.

Für den Bremsweg s_b des dreieckförmigen Profils wird die Bahnstrecke $s_2\left(t_2\right)$ unter der Bedingung $s_{ist} = 0$ betrachtet

$$\begin{aligned} &s_2\left(t_2\right) = \\ &\frac{\sqrt{2}}{12\,r_{max}^2}\left(3\,\sqrt{a_{ist}^2 + a_{soll}^2 + 2\,r_{max}\,\left(v_{ist} - v_{soll}\right)}\,\left(a_{ist}^2 - a_{soll}^2 + 2\,r_{max}\,\left(v_{ist} + v_{soll}\right)\right)\right. \\ &\left. +2\,\sqrt{2}\,a_{ist}^3 + 6\,\sqrt{2}\,a_{ist}\,r_{max}\,v_{ist} - 2\,\sqrt{2}\,a_{soll}\,\left(a_{soll}^2 - 3\,r_{max}\,v_{soll}\right)\right)\,. \end{aligned} \tag{4.67}$$

Der stets positive Bremsweg berechnet aus $s_2(t_2)$ zu

$$s_b = \frac{\sqrt{2}}{12\,r_{max}^2}\left(3\sqrt{a_{ist}^2+a_{soll}^2+2\,r_{max}\,(|v_{ist}|-|v_{soll}|)}\,\left(a_{ist}^2-a_{soll}^2+2\,r_{max}\,(|v_{ist}|+|v_{soll}|)\right)\right.$$
$$\left.+2\sqrt{2}\,\left|a_{ist}^3\right|+6\sqrt{2}\,|a_{ist}|\,r_{max}\,|v_{ist}|-2\sqrt{2}\,|a_{soll}|\,\left(a_{soll}^2-3\,r_{max}\,|v_{soll}|\right)\right). \tag{4.68}$$

Nach der analogen Betrachtung des trapezförmigen Profils mit

$$\begin{aligned} a_1(t) &:= a_{ist}-r_{max}\,t \\ a_2(t) &:= a_{ist}-r_{max}\,t_1 \\ a_3(t) &:= a_{ist}-r_{max}\,t_1+r_{max}\,(t-t_2) \end{aligned} \tag{4.69}$$

und den Randbedingungen

$$\begin{aligned} a_2(t_2) &= a_{max} \\ a_3(t_3) &= a_{soll} \\ v_3(t_3) &= v_{soll} \end{aligned} \tag{4.70}$$

berechnen sich die Bahnzeiten zu

$$\begin{aligned} t_1 &= \frac{a_{ist}+a_{max}}{r_{max}} \\ t_2 &= \frac{a_{ist}^2+2\,a_{ist}\,a_{max}+a_{soll}^2+2\,r_{max}\,(v_{ist}-v_{soll})}{2\,a_{max}\,r_{max}} \\ t_3 &= \frac{a_{ist}^2+2\,a_{ist}\,a_{max}+2\,a_{max}^2+2\,a_{max}\,a_{soll}+a_{soll}^2+2\,r_{max}\,(v_{ist}-v_{soll})}{2\,a_{max}\,r_{max}}. \end{aligned} \tag{4.71}$$

Für den Bremsweg des trapezförmigen Profils ergibt sich nach analoger Herleitung

$$s_b = \frac{1}{(24\,a_{max}\,r_{max}^2)}\left(3\,a_{ist}^4+8\,\left|a_{ist}^3\right|\,a_{max}+6\,a_{ist}^2\,\left(a_{max}^2+2\,r_{max}\,|v_{ist}|\right)\right.$$
$$+24\,|a_{ist}|\,a_{max}\,r_{max}\,|v_{ist}|+6\,a_{max}^2\,\left(2\,r_{max}\,(|v_{ist}|+|v_{soll}|)-a_{soll}^2\right)$$
$$+8\,|a_{max}|\,a_{soll}\,\left(3\,r_{max}\,|v_{soll}|-a_{soll}^2\right)+3\,\left(a_{soll}^2+2\,r_{max}\,(|v_{ist}|-|v_{soll}|)\right)$$
$$\left.\cdot\left(2\,r_{max}\,(|v_{ist}|+|v_{soll}|)-a_{soll}^2\right)\right). \tag{4.72}$$

Der profilspezifische Bremsweg ist die Grundlage der weiteren Geschwindigkeitsregelung. Falls der Bremsweg s_b größer ist als die zur Verfügung stehende restliche Bahnstrecke $|s_{soll}-s_{ist}|$, muss auf den Zielzustand $(a_{soll}, v_{soll}, s_{soll})$ eingeregelt werden. Sonst findet eine Geschwindigkeitsregelung auf die maximale Geschwindigkeit v_{max} statt.

Geschwindigkeitsregelung

Der Geschwindigkeitsregler liefert nur den wert-diskreten Bahnruck als Stellgröße. Der Wert des Bahnrucks ist abhängig vom aktuellen Zeitintervall i. Das Umschalten

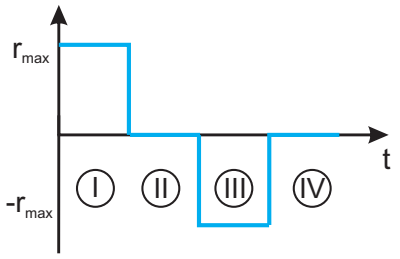


Abbildung 4.36: Werte für den Ruck des Bahnreglers in den Zeitintervallen i

von i wird weiter unten behandelt.

Die Fahrtrichtung $ri \in \{-1, 1\}$ ist positiv, falls $s_{soll} > s_{ist}$ gilt, sonst negativ. Abhängig vom Intervall i und ri ergibt sich der Bahnruck

$$r_{ist} = \begin{cases} ri\, r_{max}, & \text{für } i = 1 \\ 0, & \text{für } i \in \{2, 4\} \\ -ri\, r_{max}, & \text{für } i = 3 \end{cases} . \tag{4.73}$$

Die Ist-Werte der Regelung a_{ist}, v_{ist} und s_{ist} berechnen sich im k-ten Regelzyklus mit der Zyklusdauer Δt aus der numerischen Summation

$$\begin{aligned} a_{ist,k} &= a_{ist,k-1} + r_{ist}\, \Delta t \\ v_{ist,k} &= v_{ist,k-1} + a_{ist,k}\, \Delta t \\ s_{ist,k} &= s_{ist,k-1} + v_{ist,k}\, \Delta t\ . \end{aligned} \tag{4.74}$$

Der Regler starte im Zeitintervall $i = 1$ mit der Zielgeschwindigkeit $v_{soll} = v_{max}$. Ein Intervallwechsel tritt ein, wenn

- a_{max} erreicht ist: $a_{ist} \geq a_{max} \rightarrow i = 2$
- eine sofortige Verringerung der aktuellen Beschleunigung dennoch zur Überschreitung von v_{soll} führt: $v_{ist} + \frac{a_{ist}^2}{2\, r_{max}} \geq v_{soll} \rightarrow i = 3$.

Im Intervall $i = 2$ wird die Beschleunigung $a_{ist} = a_{max}$ gehalten. Ein Wechsel tritt analog zum vorherigen Schritt ein, wenn

- eine sofortige Verringerung der aktuellen Beschleunigung dennoch zur Überschreitung von v_{soll} führt: $v_{ist} + \frac{a_{ist}^2}{2\, r_{max}} \geq v_{soll} \rightarrow i = 3$.

In Intervall $i = 3$ wird die Beschleunigung so lange verringert bis $a_{ist} = 0 \rightarrow i = 4$.

In Intervall $i = 4$ wird die aktuelle Geschwindigkeit gehalten.

Die Bedingung des Bremswegs $|s_{soll} - s_{ist}| > s_b$ muss in jedem Regelzyklus überwacht werden. Falls sie nicht mehr gilt, findet ein sofortiges Umschalten mit $v_{soll} = v_{soll,Ziel}$ und $i = 1$ statt.

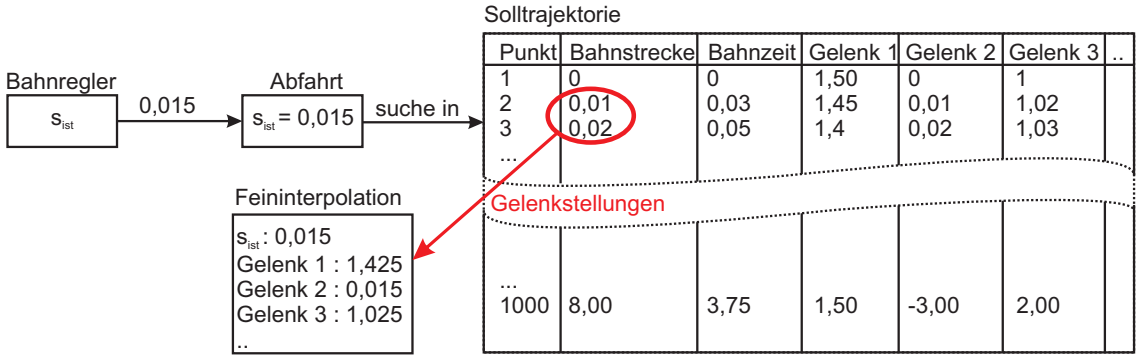


Abbildung 4.37: s_{ist} der Bahnregelung mit dem Beispielwert 0,015 führt über die Suche und die Feininterpolation in der Solltrajektorie zu den Gelenkstellungen für die Antriebssteuerung .

Exaktes Erreichen der Ziellage

In der Ziellage sollen alle Zustandsparameter die Werte a_{soll}, v_{soll} und s_{soll} zugleich annehmen. Aus der numerischen, mehrfachen Summation des Rucks r_{ist} mit der Zyklusdauer Δt ergibt sich auf dem Rechner wegen Rundungsfehlern eine geringe Ungenauigkeit des Bahnparameters an der Ziellage $s_{ist} \neq s_{soll}$. Denn nach dem Umschalten der Regelung auf die Zielgeschwindigkeit v_{soll}, nachdem die Bedingung $|s_{soll} - s_{ist}| \leq s_b$ galt, gilt an der Ziellage aufgrund des aufsummierten Fehlers nicht $|s_{soll} - s_{ist}| = s_b$. Um diesen Fehler zu kompensieren wird der Bahnruck während des Zeitintervalls $i = 3$ um den Faktor r_{faktor} und die Bahnbeschleunigung bei dem Eintritt in Zeitintervall $i = 3$ korrigiert:

$$\begin{aligned} r_{faktor} &:= 2\,(|v_{ist}| - |v_{soll}|)\,\frac{v_{ist} + 2\,v_{soll}}{3\,(s_{soll} - s_{ist})}\,\frac{v_{ist} + 2\,v_{soll}}{3\,(s_{soll} - s_{ist})}\,\frac{1}{r_{max}} \\ a_{ist} &= 3\,ri\,r_{faktor}\,r_{max}\,\frac{s_{soll} - s_{ist}}{v_{ist} + 2\,v_{soll}} \end{aligned} \tag{4.75}$$

Bahnstrecke und -geschwindigkeit verlaufen während der Korrektur stetig und die Bahnbeschleunigung erfährt einen geringen Sprung.

Anwendungen des Bahnreglers: s- und t-Regelung

Ist die Solltrajektorie gegeben, kann eine Regelung auf der Bahnstrecke s oder der Bahnzeit t aufsetzen. Beide Größen sind streng monoton wachsend und eindeutig auf Gelenkstellungen abgebildet.

Abbildung 4.37 zeigt die Verwendung der Regelungsausgabe s_{ist} im Kontext der Bahnstecke s. Hiermit wird die Bahn mit einer einfachen ruckbegrenzten Rampe abgefahren. Eine s_{ist}-spezifische Änderung von v_{max} durch ein Anwenderprogramm ermöglicht auch die Definition komplexer Geschwindigkeitsverläufe. Die zeitintensive, zeitoptimale Geschwindigkeitsberechnung und die aufwendige Modellierung der Roboterdynamik werden für die s-Regelung nicht benötigt. Wird a_{max} oder v_{max} zu hoch gewählt, können die dynamischen Grenzen des Roboters an Bahnstellen, die hohe Drehmomente den Gelenke abverlangen, verletzt werden, während der Roboter an anderen Bahnstellen durchaus schneller fahren könnte.

Bei der t-Regelung wird die Regelungsausgabe s_{ist} auf die Bahnzeit t abgebildet und ermöglicht die Abfahrt der Bahn auf Basis des zeitoptimalen Geschwindigkeitsprofils mit beliebig skalierter Fahrzeit.

Anwendungsfälle:

1. Ein neuer Roboter wird in Betrieb genommen. Ohne viel Aufwand soll eine glatte Bahn mit geringer Geschwindigkeit abgefahren werden.

2. Die Bahn eines Roboters soll schnellstmöglich abgefahren werden. Die Randbedingungen der Last erlauben jedoch nur eine geringere, unbekannte Geschwindigkeit.

3. Die Bahn eines Roboters soll schnellstmöglich abgefahren werden. Er soll mitten in der Bahn anhalten.

Vorgehensweisen:

1. Zur Erstellung der Solltrajektorie müssen die Bahninterpolation und der s-Bahnregler in der Robotersteuerung implementiert werden. Die Regelparameter $(r_{max}, a_{max}, v_{max})$ sollten klein gewählt werden. Über die Vorgabe von s_{soll} wird der Roboter ruckbegrenzt über die Bahn bewegt. Der Maximalwert für $s_{soll} = \widehat{s}$ ist die komplette Bahnlänge.

2. Zusätzlich zu Fall 1 müssen die Robotermodelle zur Berechnung der zeitoptimalem Trajektorie erstellt werden. Soll die Bahn 1:1 mit der zeitoptimalen Geschwindigkeit und der berechneten Fahrdauer gefahren werden, wird der t-Regler mit $r_{max} = \infty, a_{max} = \infty$ und $v_{max} = 1$ parametriert. Die Ruck- und Beschleunigungsbegrenzung des Bahnreglers wird nicht verwendet, da bei der zeitoptimalen Trajektorie die Größen bereits berücksichtigt wurden. Für den Bahnregler muss $s_{soll} \in \left[0, \widehat{t}\right]$ gesetzt werden, wobei $\widehat{t}$ die Fahrdauer der gesamten Bahn darstellt. Zur Inbetriebnahme mit einer Last wird mit einem kleinen Wert für v_{max} begonnen und langsam auf 1 erhöht. Werte über 1 überfordern die Roboterantriebe, können aber aufgrund von Modellidealisierungen praktisch verwendbar sein.

3. Im Gegensatz zu Fall 2 soll der Roboter durch den Regler mitten auf der Bahn anhalten. Der Halt ist im zeitoptimalen Geschwindigkeitsprofil nicht eingeplant. Das Profil besitzt an dieser Stelle ein $v >> 0$. Deshalb muss der t-Regler ruckbegrenzt abbremsen. r_{max} und a_{max} sollten kleine Werte besitzen. Als Seiteneffekt verlängert sich die Beschleunigungsphase zu Beginn der Trajektorie, da sowohl der Bahnregler als auch die zeitoptimale Geschwindigkeitsberechnung zu einer eigenen ruckbegrenzten Beschleunigung führen. Diese Überlagerung tritt auch am Ende der Trajektorie auf.

4.5 Fazit

In diesem Kapitel wurden zunächst die Prinzipien verschiedener Bahnplanungsverfahren nach Latombe [Latombe 96] vorgestellt und auf Anwendbarkeit für den online-Einsatz bei Industrierobotern untersucht. Für die weitere Verwendung wurde das Wellenfrontverfahren ausgewählt und seine algorithmische Beschreibung mit Aufwandsabschätzung erstellt.

Die Interpolation erzeugt aus der Bahnbeschreibung der Bahnplanung eine verwendbare Trajektorie. Die Analyse unterschiedlicher Interpolationsverfahren führte zu der Auswahl der Kombination aus Linearinterpolation mit effizient implementierbaren kubischen Splines wie von Press [Press 92] vorgestellt.

Auf Basis der geometrischen Kontur der Trajektorie wurden die Ergebnisse der Robotermodellierung nach den Prinzipien von Olomski [Olomski 89] für die Berechnung der optimalen Bahngeschwindigkeit und der Beschreibung des Zeitprofils in Form von (s, v)-Stützpunkten verwendet. Olomski [Olomski 89] verzichtet jedoch auf die Beschleunigungsprofile, sondern verwendet die (s, v)-Punkte direkt als Vorgaben während der Fahrt für den Bahnregler und kann somit eine zeitoptimale Trajektorie nicht mit z. B. exakt 50% der optimalen Geschwindigkeit fahren. Sein Ansatz zeigt zudem Lücken durch die Vernachlässigung der Bahnbeschleunigung und kann zu Verletzungen der dynamischen Grenzen des Roboters führen. Durch eine eigene Entwicklung der Interpolation im Grenzgeschwindigkeitsprofil mit unterschiedlichen ruckbegrenzten Beschleunigungsprofilen unter Beachtung der Grenzbeschleunigung $\ddot{s}$ wurde diese Lücke geschlossen.

Das Ergebnis der Berechnung wird numerisch in einer Trajektorien-Tabelle abgelegt. Sie verknüpft den Bahnparameter s mit der Bahnzeit t und den Gelenkstellungen $\vec{\theta}$.

Zur Abfahrt einer Trajektorie wurde ein zustandsbasierter Bahnregler entwickelt. Er lässt eine Änderung der Reglervorgaben wie maximale Bahngeschwindigkeit während der Fahrt zu und ermöglicht in s-Regelung die Abfahrt einer Bahn auch ohne die aufwendige zeitoptimale Geschwindigkeitsberechnung.

5. Optimierung und Simulation

Robotik ist ein interdisziplinäres Gebiet. Bevor Anwendungsprogramme mit einem Robotersystem genutzt werden können, müssen Regelung, Sensorik, Antriebselektronik, Hardwareanbindung, Sicherheitsaspekte, etc. spezifiziert und implementiert sein. Dieses Hindernis muss zuerst überwunden werden, obwohl Forscher sich auf Funktionen der Anwendungsebene konzentrieren möchten.
Die SPS-Programmierumgebung bietet mit Debuggen nur eine sehr beschränkte Möglichkeit zum Test und zur Simulation an. Eine Simulation ist nicht nur ein Werkzeug zum Test der vorgestellten Modelle, sondern sie kann neue Erkenntnisse über das Verhalten des Bahnplanungssystems liefern. Die hier beschriebenen nachteiligen Eigenschaften des bisherigen Systems wurden zum Teil erst durch eine Simulation ersichtlich.

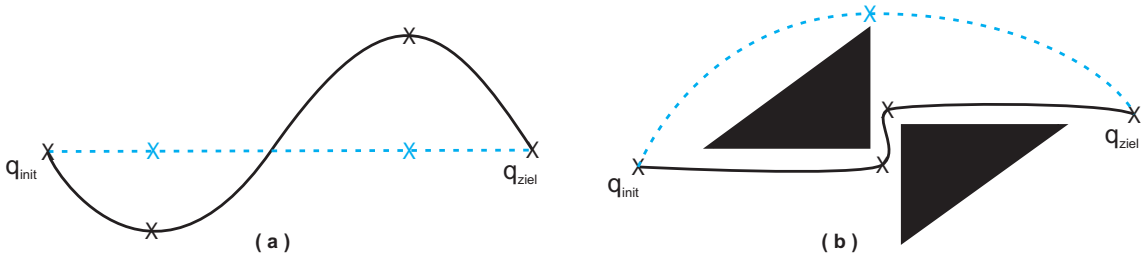


Abbildung 5.1: Ziele der Trajektorienoptimierung: Zu den ursprünglichen Stützstellen und der Bahn (durchgezogen) soll eine zeitoptimierte Bahn (gestrichelt) gefunden werden. Die Optimierung kann zu einer Verkürzung (a) oder einer Verlängerung (b) der Bahn führen

Die Optimierung soll zu einer besseren Bahn führen. Dies kann wie in Abbildung 5.1a) eine einfache Begradigung bedeuten oder wie in Abbildung 5.1b) die komplette Änderung des Bahnverlaufs implizieren.

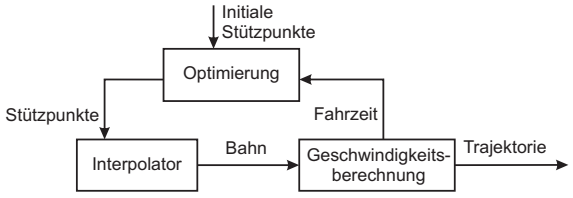


Abbildung 5.2: Rückkopplung zwischen Planung des Zeitprofils und der Bahngenerierung für die Optimierung der Fahrzeit

Die zeitoptimale Geschwindigkeitsberechnung findet den optimalen Geschwindigkeitsverlauf für eine gegebene Bahn. **Abschnitt 5.1** beschreibt Verfahren zur Zeitoptimierung von Trajektorien durch die Variation der Stützpunktlage und zeigt die Effekte anhand der SCARA-Kinematik.

Da die für die Zeitoptimierung nötige, häufige Berechnung des zeitoptimalen Geschwindigkeitprofils sehr rechenintensiv ist, wird in **Abschnitt 5.2** ein Näherungsverfahren zur Bahnoptimierung, das auf Zeitprofile verzichtet, entwickelt. Dieses Verfahren setzt direkt auf die Bahnplanung aus Abschnitt 4.1.8 auf. Die Korrelation der mit diesem Verfahren optimierten Bahnen mit den tatsächlich zeitoptimalem Trajektorien wird untersucht.

Die verwendete Kuka SoftSPS bietet im Vergleich zu anderen SPS-Systemen viel Speicher an. Dennoch ist der für globale Variablen zur Verfügung stehende Speicher von ca. 300 kB für die numerische Speicherung mehrerer Trajektorien zu gering. In **Abschnitt 5.3** wird das Bahnplanungssystem so modifiziert, dass Bahnen erst während der Fahrt berechnet und nicht gespeichert werden müssen.

Abschnitt 5.4 beschreibt verfügbare Robotersimulationen und geht auf den Aufbau der erstellten Simulation ein, mit der die Modellierung und der Einfluss von Parametern auf die Trajektoriengenerierung nachgeprüft wurden.

5.1 Zeitoptimierung der Trajektorie

Bahnplanungsverfahren liefern oft die erstbeste oder die kürzeste Bahn. Solche Bahnen sind für mobile, langsame Roboter, die große Distanzen zurücklegen, sinnvoll und schnell abfahrbar, aber nicht für schnelle Industrieroboter mit kurzen Fahrzeiten. Erst die Berücksichtigung der Roboterdynamik wie in Abbildung 5.2 erlaubt der Bahnplanung die Erzeugung optimaler Trajektorien.

Die Optimierung der Fahrzeit unterliegt verschiedenen Randbedingungen wie Roboterdynamik und Hindernissen. In der Regelungstheorie, die den Roboter als nichtlineares gekoppeltes Mehrkörpersystem auffasst, oder in der getrennten Planung der geometrischen Bahn und der Suche nach einem optimalen Geschwindigkeitsprofil auf dieser Bahn lassen sich Lösungen für eine zeitoptimale Bahn finden. Letztere Methode ist jedoch aus der Sicht der Optimierung aufwendig.

Ozaki und Lin [Lin 96] beschreiben eine Optimierung der Gelenk-Trajektorie mit

Berücksichtigung von Hindernissen und priorisierten Randbedingungen. Wegen der lokalen Kontrollierbarkeit kommen B-Splines zum Einsatz. Die Optimierung auf Basis der Komplex-Methode benötigt nicht die Berechnung des Gradienten. Die Kontrollpunkte der Trajektorie werden iterativ verschoben, bis der Leistungsindex sich nicht wesentlich verbessert. Der Leistungsindex berechnet sich aus der gewichteten Fahrzeit und der gewichteten Verletzung von Randbedingungen wie Kollisionen und maximales Drehmoment. Die Gewichte werden dynamisch angepasst, so dass alle Randbedingungen nacheinander, hoch priore vor niedrigeren, optimiert und erfüllt werden. In [Lin 96] wird zur Demonstration ein Roboter mit drei Armsegmenten aufgegriffen, der sich kollisionsfrei bewegen soll. Während den Iterationen wird zuerst die Kollisionsvermeidung optimiert, danach die Randbedingungen maximale Gelenkgeschwindigkeiten und -drehmomente.
Der Vorteil dieses Ansatzes ist, dass er Bahnplanung, Interpolation und Optimierung kompakt und integriert behandelt und eine Priorisierung der Randbedingungen zulässt. Jedoch arbeitet das Verfahren nur im Gelenkwinkelraum und ist auch für Roboter mit mehreren Armsegmenten, die um Hindernisse herum greifen sollen, anwendbar. Das ist bei den KHS-Robotern nicht nötig.

Fourquet [Fourquet 90] zeigt die analytische Modellierung mit der singulären Steuerungstheorie für zeitoptimale Punkt-zu-Punkt-Trajektorien mit Beachtung der Dynamik. Mit Hilfe der Lie-Algebra und des Maximum-Prinzip wird die Komplexität der Berechnung reduziert und auf die Berechnung der Gradienten verzichtet.
Fourquet [Fourquet 90] demonstriert das Verfahren an einem 2-Achser. Die Erweiterung auf Roboter mit mehr Freiheitsgraden ist komplex und das komplette Verfahren ist schwer nachvollziehbar.

Bobrow [Bobrow 88] behandelt eine approximative, nicht-lineare Optimierung der Bahn im kartesischen Raum mit Kollisionsvermeidung und Beachtung der Grenzen der Aktuatoren. Ausgehend von einer parametrierten Initialbahn mit B-Splines werden die Bahnparameter iterativ verändert um zur zeitoptimalen Trajektorie zu gelangen. Die Berechnung der Fahrzeit und die Erstellung des Geschwindkeitsprofils geschieht analog zu der in dieser Arbeit verwendeten Methoden. Es bleiben zwei unbehandelte Randbedingungen, die der maximalen Gelenkstellungen und die des minimalen Abstands der Bahn zu Hindernissen $d(\mathbb{X})$ mit dem Sicherheitsabstand δ, übrig:

$$\begin{aligned} q_{i,min} \leq q_i \leq q_{i,max} \text{ für alle Gelenke i} \\ 0 < \delta \leq d(\mathbb{X}) \ . \end{aligned} \tag{5.1}$$

Die Gradienten der Fahrzeit, des minimalen Hindernisabstands und der Drehmomente werden aus Effizienzgründen numerisch berechnet. Der Bahn werden neue Zwischenpunkte hinzugefügt und anschließend zur Suche einer besseren Lage der Punkte die Verfahren nach Davidon-Fletcher-Powell und Golden Section angewandt. Die Trajektorie konvergiert mit steigender Zahl neuer Zwischenstützpunkte gegen die optimale Kurve. Es zeigen sich jedoch Oszillationen bei dicht aufeinander folgenden Punkten, die das Optimierungsverfahren zu früh abbrechen lassen. Um dem entgegenzuwirken, fließt das Integral über der quadratischen Bahnkrümmung mit in die Minimierung ein. Die Fahrzeit wird durch die ersten hinzugefügten Zwischenpunkte

am stärksten verringert. Weitere Punkte verbessern die Fahrzeit nur gering, führen aber zu einem hohen Rechenaufwand.

Park und Bobrow [Bobrow 98] zeigen eine numerische Methode zur Optimierung, indem die Bewegung als einen Vektor aus Spline-Koeffizienten und der Fahrzeit in einem Vektorraum repräsentiert und der optimale Punkt gesucht wird. Gelenkbewegungen werden relativ zu Spline-Koeffizienten ausgedrückt und spannen mit Gelenkbeschleunigung und -ruck einen linearen Vektorraum auf. Der Leistungsindex berechnet sich aus der Fahrzeit und der Verletzung von Randbedingungen und Kollisionen. Der Gradient des Leistungsindex' wird numerisch über den Differenzenquotient bestimmt und anschließend über die lineare Golden Section-Suche eine eindimensionale Minimierung des Leistungsindex' durchgeführt.
Park und Bobrow [Bobrow 98] arbeiten nur im Gelenkwinkelraum und vernachlässigen Randbedingungen der Last wie den maximalen Bahnruck.

Die vorgestellten Verfahren reduzieren die Bahnoptimierung auf ein Minimierungsproblem. Sie haben folgende gemeinsamen Eigenschaften:

- Entwicklung der zeitoptimalen Trajektorie in mehreren Iterationen.
- Die Trajektorienoptimierung berücksichtigt die Roboterdynamik.
- Die Fahrzeit ist Teil des Güte- / Minimierungskriteriums.
- Verbessert sich die Fahrzeit in einer Iteration, wird diese Bahn als Ausgangsbasis der weiteren Optimierung verwendet.

Die Literatur beantwortet nur teilweise, wie viele Iterationen für eine gute Trajektorie nötig sind, wie groß der Unterschied der gefunden Trajektorie zur theoretisch besten ist und ob ein Verfahren vollständig ist, d.h. ob es immer die optimale Trajektorie bei unendlicher Laufzeit findet.

Im Folgenden wird ein Verfahren hergeleitet, das die Durchlaufzeit der Bahn durch gezieltes Verschieben von Bahnstützpunkten optimiert. Es soll untersucht werden, unter welchen Randbedingungen die Zeitoptimierung verwendet werden kann.

5.1.1 Optimierung der Lage eines Bahnstützpunkts

Gegenstand der ersten Untersuchung ist die Fahrt der SCARA-Kinematik über eine einfache Bahn. Im Vergleich zu den KHS-Kinematiken mit weniger Gelenken bietet der SCARA-Roboter durch Abstütz-, Coriolis- und Zentrifugalkräften mehr Freiheitsgrade bei der Optimierung an. Eine Optimierung soll die in Abbildung 5.3a) dargestellte Impulserhaltung ausnutzen. Für die Impulserhaltung gilt

$$\begin{aligned} F &= \dot{p} = 0 \\ \vec{L} &= I\,\vec{\omega} = \text{const.} \end{aligned} \tag{5.2}$$

mit

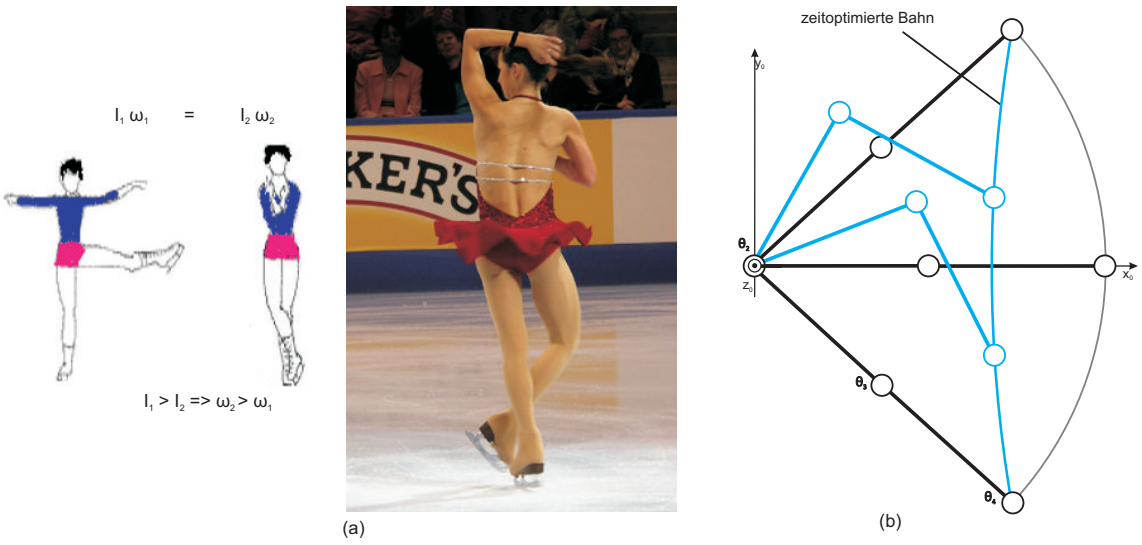


Abbildung 5.3: a) Der Pirouetteneffekt zur Erhöhung der Winkelgeschwindigkeit im Eiskunstlauf basiert auf Impulserhaltung und Coriolisbeschleunigung, aus [Tolan 02] S. 344 und Foto mit freundlicher Genehmigung von Tracy Marks b) analog soll das Anziehen und Ausstrecken des SCARA-Arms eine höhere Beschleunigung herbeiführen

F	Kraft
p	Impuls
$\vec{L}$	Drehimpuls
I	Trägheitsmoment
$\vec{\omega}$	Winkelgeschwindigkeit.

Ausgehend von einer Bahn, die vorerst nur durch die Bewegung des Armgelenks zustande kommt, soll die Fahrdauer optimiert werden. Wird während der Fahrt der Arm eingezogen, verringert sich I und der Arm erfährt aufgrund der Impulserhaltung eine zusätzliche Beschleunigung. Analog wird der Arm beim Ausstrecken am Bahnende verzögert. Die zusätzlichen Beschleunigungen unterstützen die Antriebe. Abbildung 5.3b) skizziert die ursprüngliche und die erwartete, optimierte Bahn.

Ein eindimensionales Minimierungsverfahren mit Intervallschachtelung wie in Abbildung 5.4 dargestellt kann nach Press [Press 92] auf beliebige Dimensionen wie in Abbildung 5.5 erweitert werden. Gegenstand der Minimierung ist die Lage des mittleren Bahnstützpunkts. Er wird so lange iterativ verschoben, bis das Mimierungskriterium Fahrzeit nicht weiter signifikant sinkt.

Nach Walser [Walser 01] und Press [Press 92] kann der Goldene Schnitt die Stelle eines neuen Punkts, der das Intervall teilt, vorgeben. Er muss zu den Intervallgrenzen den relativen Abstand von ca. 0,38197 und 0,61803 besitzen. Die Wahl der initialen Intervallgrenzen findet durch eine parabolische Approximation der zu untersuchen-

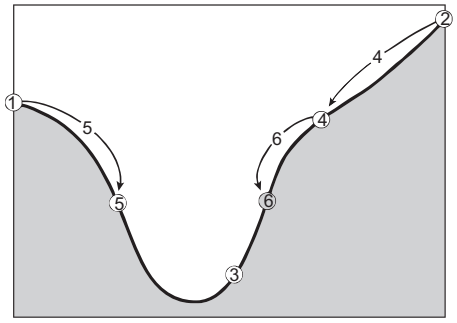


Abbildung 5.4: 1-dimensionales Suchverfahren nach dem Goldenen Schnitt. Das Intervall, in dem das Minimum liegt, wird iterativ verkleinert, aus [Press 92] S. 398

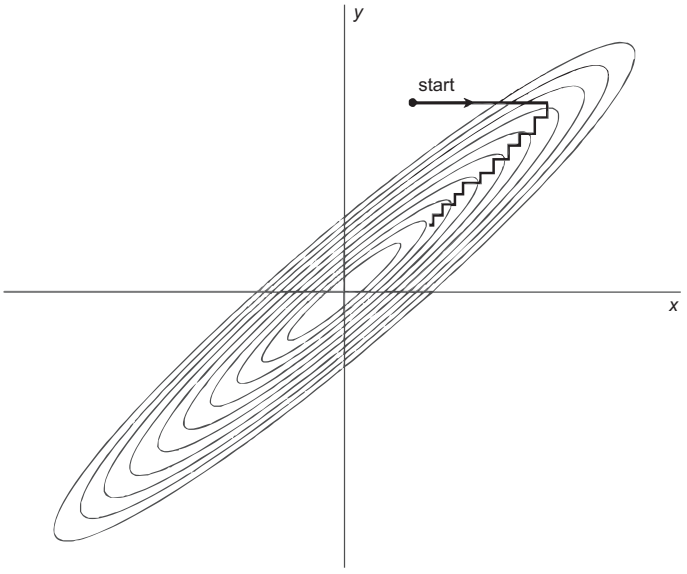
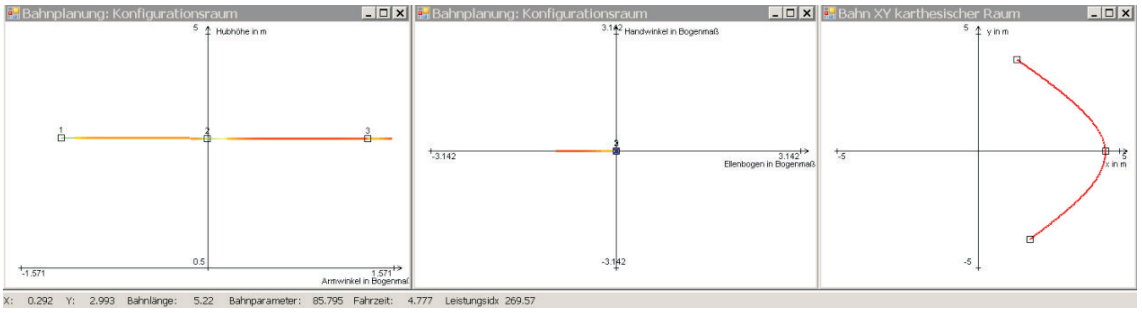
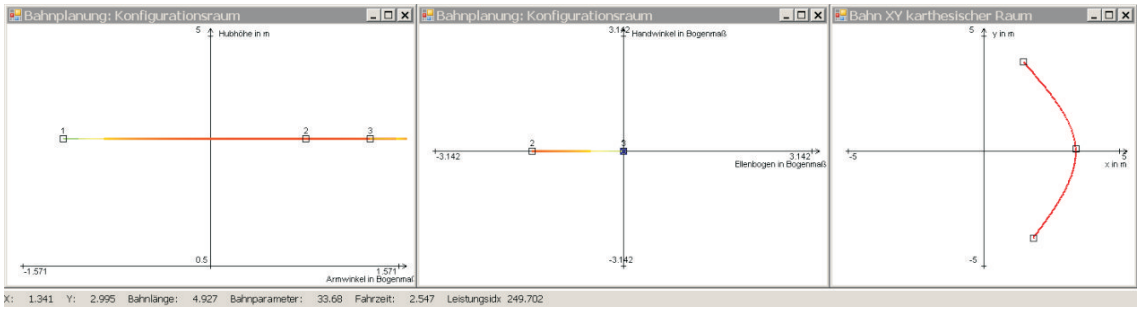


Abbildung 5.5: Anwendung von 1-dimensionalen Suchverfahren im mehrdimensionalen Raum durch sukzessive Minimierung für jede Koordinatenrichtung, aus [Press 92] S. 414



(a)



(b)

Abbildung 5.6: Optimierung der SCARA-Trajektorie durch Verschieben des zweiten Stützpunkts. Das Kriterium ist die Fahrzeit. a) vor und b) nach der Optimierung (aus Simulation)

den Funktion statt. Danach verringert jede Iteration die Intervallgröße auf 61,8% des vorherigen Werts.

Abbildung 5.6 zeigt die Ergebnisse des Versuchs in einer eigen entwickelten Simulation der Bahnplanung (vgl. Abschnitt 5.4). Sie stellt die Lage der Stützpunte und den Bahnverlauf vor und nach der numerischen Optimierung dar. Ausgehend von der ursprünglichen Bahn mit einer Fahrdauer von 4,78 s kann die optimierte Bahn in 2,55 s gefahren werden. Die Bahnänderung ist geringer als in Abbildung 5.3b) erwartet. Ein manuelles Verformen der Bahn, so dass sie sich der Bahn aus Abbildung 5.3b) annähert, erhöht wegen der Beschleunigungsbegrenzung des Ellenbogengelenks die Fahrzeit. Die Kontur der optimierten Bahn hängt von den dynamischen Parametern des Roboters ab.

5.1.2 Probleme der Optimierung mehrerer Bahnstützpunkte

Da eine Bahn im Allgemeinen über beliebig viele Stützpunkte definiert ist, soll eine Bahnoptimierung die Lage mehrerer Stützpunkte berücksichtigen. Hierzu kann das vorgestellte Verfahren die Lage jedes Stützpunktes für sich alleine optimieren. Durch mehrere Iterationen über alle Stützpunkte wird die Fahrzeit der Bahn sukzessive verbessert.

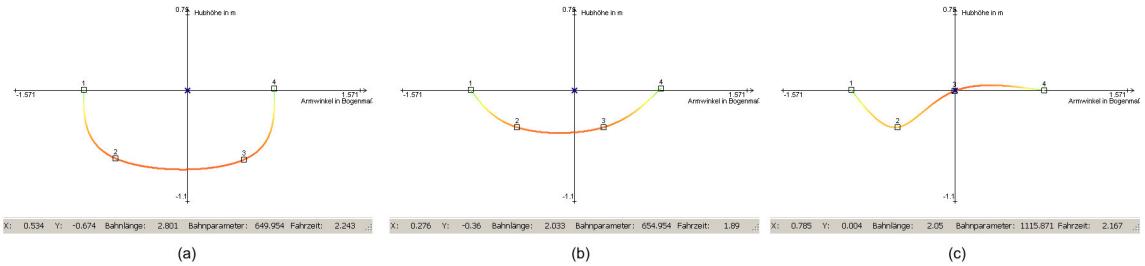


Abbildung 5.7: Die Optimierung von zwei Bahnstützpunkten einer Trjektorie terminiert in einem lokalen Minimum. a) Ausgangsbahn b) optimierte Bahn mit vorzeitigem Abbruch der Optimierung c) manuelles Versetzen eines Stützpunkts auf die global-optimale Bahn erhöht die Fahrdauer (aus Simulation)

Abbildung 5.7 demonstriert die Zeitoptimierung einer Bahn durch Verschieben mehrerer Stützpunkte. Die Bahn aus Abbildung 5.7a) eines 2-achsigen Roboters mit Hub- und Armgelenk ist durch vier Stützpunkte im Gelenkwinkelraum beschrieben. Die Optimierung soll die Bahnkontur in eine Gerade überführen. Die Optimierung bricht jedoch wie in Abbildung 5.7b) vorzeitig ab, da die Fahrzeit ein lokales Minimum erreicht. Ein weiteres Verschieben des Punktes 2 oder 3 in Richtung der Geraden erhöht die Bahnkrümmung am jeweils anderen Punkt. Daher muss die Bahngeschwindigkeit verringert werden. Abbildung 5.7c) belegt dies. Analog zur Bahnplanung mit Potentialfeld-Methoden aus Abschnitt 4.1 stellen lokale Minima auch für die analytische Optimierung der Bahn ein Problem dar.

5.1.3 Aufwandabschätzung

Der Aufwand zur Auswertung der Fahrzeit mit Bahninterpolation und Geschwindigkeitsberechnung liegt nach Abschnitt 4.3.2 in $\mathcal{O}_{fahrzeit} := \mathcal{O}\left(m \cdot n \cdot \log\left(n_a\, n_{grenz}\right)\right)$ mit n_{grenz}: Punkte des numerischen Grenzgeschwindigkeitsprofils, n_a: Punkten zwischen zwei benachbarten Geschwindigkeitsextrema, m: durch die Interpolation eingefügte Zwischenpunkten, n: Zahl der Bahnstützpunkte. Die Fahrzeit muss pro Stützpunkt n, pro Dimension des Raums d und pro Iteration i der numerischen Optimierung ausgewertet werden. Der Gesamtaufwand liegt in $\mathcal{O}\left(i \cdot d \cdot m \cdot n^2 \cdot \log\left(n_a\, n_{grenz}\right)\right)$. Die Zahl der Iterationen i ist durch den maximalen Gelenkbereich und der Auflösung der Achse gegeben. Besitzt eine rotatorische Achse den Wertebereich von 360°und eine Auflösung von $10^{-4\circ}$, dann muss $i = 22$ betragen, um bei einer Intervallhalbierenden Intervallschachtelung den Wertebereich auf die Größenordnung der Auflösung zu reduzieren.

Press [Press 92] stellt effizientere Verfahren zur Lösung von numerischen Minimierungsproblemen vor. Die zweite Untersuchung zeigte jedoch, dass analytische Verfahren nicht immer das globale Minimum der Fahrzeit erreichen sondern vorzeitig in einem lokalen Minimum terminieren. Der folgende Abschnitt stellt ein effizientes Verfahren zur Behebung dieses Problems vor.

5.2 Optimierung der Bahnplanung anhand der Bahnkontur

Die Zeitoptimierung des vorherigen Abschnitts führt nicht zu den gewünschten Ergebnissen. Das Ziel ist ein schnelles Verfahren, das gute Ergebnisse mit beschränkten Rechenkapazitäten liefert, zu finden.

Im Folgenden wird ein Optimierungsverfahren entwickelt, welches ohne die Auswertung der Fahrzeit zu suboptimalen Trajektorien führt. Das Verfahren soll skalierbar sein: Je mehr Ausführungszeit investiert wird, desto besser wird die Trajektorie. Es knüpft direkt an die Wellenfront-Bahnplanung des Abschnitts 4.1.8 an.

Anhand der Simulation wurden folgende Eigenschaften zeitoptimaler Trajektorien beobachtet:

1. Je kürzer die Bahn, desto kürzer ist die Fahrdauer.
2. Hohe Bahnkrümmungen führen zur Verringerung der Geschwindigkeit. Sind sie erforderlich, soll die Bahnkrümmung möglichst konstant sein.
3. Äquidistante Bahnstützpunkte führen zu einer gleichmäßigeren Bahnkrümmung bei kubischen Splines.
4. Die Verwendung von Splines ist besser als lineare Bahnsegmente.
5. Die Verringerung der Anzahl der Stützpunkte verbessert die Bahn.
6. Die optimale Lage der Stützpunkte ist nahe an Hindernissen und in der Nähe der durch die Wellenfront berechneten Stützpunkte.
7. Der global-optimale Verlauf der Bahn kann komplett verschieden sein zu der durch die Wellenfront erzeugten Bahn.

Zunächst wird eine Metrik zur Beurteilung der Güte einer Trajektorie hergeleitet. Sie soll auf die Fahrdauer verzichten und nur auf Eigenschaften der geometrischen Bahnkontur basieren.

In den Eigenschaften eins und zwei sind die Bahnlänge und Bahnkrümmung von Bedeutung. Der von Park und Bobrow [Bobrow 98] vorgestellte Mechanismus des Leistunsindex' wird hier aufgegriffen und an die Anforderungen des Optimierungsproblems angepasst.

Der Leistungsindex der Bahn Idx berechnet sich zu

$$Idx_{bahn} = k_s\, l_s + k_\sigma \sqrt{\frac{1}{N-1} \sum_i \left(Bahnkrümmung_i - \overline{Bahnkrümmung}\right)^2}. \quad (5.3)$$

Er besteht aus der gewichteten Summe aus Bahnlänge und der Standardabweichung der Bahnkrümmung. Die Standardabweichung liefert den Grad der Streuung der

Bahnkrümmung um den Mittelwert der Krümmung. Für die kinematischen Daten der KHS-Roboter zeigte sich für $k_s = 50$ und $k_\sigma = 0{,}1$ (RS3) bzw. $k_\sigma = 1$ (KHS-Einleger) eine hohe Korrelation des Leistungsindex' mit der Fahrdauer. Die Gewichtungen sind abhängig von der Größe des Arbeitsbereiches der Achsen und davon, ob lange Bahnen mit geringer Krümmung oder kurze mit teilweise stärkeren Krümmungen für die Anwendung als gut erachtet werden.

Der Leistungsindex kann während der Interpolation und Geschwindigkeitsberechnung ohne zusätzlichen Aufwand berechnet werden. Die Bahnlänge l_s wird bei der Interpolation berechnet, die Bahnkrümmung ergibt sich aus der 2. Ableitung der Bahn, die auch für die Herleitung der Grenzgeschwindigkeit benötigt wird.

5.2.1 Optimierungsverfahren mit Leistungsindex

Nun wird die Bahn, die durch das Wellenfront-Methode gewonnen wurde, nachbearbeitet. An der Bahn, die zunächst nur aus linearen Segmenten besteht, werden mit einem eigenen Verfahren die Erkenntnisse aus den Eigenschaften vier und fünf angewandt:

Listing 5.1: Runden der Bahnsegmente

```
wellenfrontAusfuehren     // -> kollisionsfreie, lineare Bahnsegmente
berechneBahnsegmentLaengen
berechneLeistungsindex
wiederhole fuer alle Bahnsegmente
        sucheLaengstesBahnsegment
        aendereZuSpline(laengstesBahnsegment)
        wenn bahnKollidiert dann
                teste(1)
                        loescheStuetzpunkt(Startpunkt von
                            laengstesBahnsegment)
                        berechneLeistungsindex
                        ueberpruefeKollision
                teste_ende
                teste(2)
                        loescheStuetzpunkt(Endpunkt von
                            laengstesBahnsegment)
                        berechneLeistungsindex
                        ueberpruefeKollision
                teste_ende
                wenn leistungsindex(1) < leistungsindex(2) und
                    kollisionsfrei(1) dann
                        uebernehme(Bahn aus test(1))
                sonst wenn leistungsindex(2) < leistungsindex(1) und
                    kollisionsfrei(2) dann
                        uebernehme(Bahn aus test(2))
                sonst
                        behalteAltesLinearesBahnsegment
                wenn_ende
        wenn_ende
wiederhole_ende
```

Dieses Verfahren rundet eine Bahn und löscht Bahnstützpunkte, falls Kollisionen auftreten. Die Änderungen werden nur übernommen, wenn die Bahn kollisionsfrei

bleibt und ihr Leistungsindex sich verbessert.

Im nächsten Schritt wird analog die Eigenschaft fünf genutzt:

Listing 5.2: Löschen von Stützpunkten

```
berechneLeistungsindex
wiederhole fuer alle Stuetzpunkte i
        loescheStuetzpunkt(i)
        berechneLeistungsindex(i)
        wenn nicht bahnkollidiert und leistungsindex(i) <
            leistungsindex(alteBahn) dann
                uebernehmeAktuelleBahn
        sonst
                verwerfeAenderungen
        wenn_ende
wiederhole_ende
```

Schritt (5.2) optimiert die Bahn durch Löschen beliebiger Stützpunkte. Nun kann Schritt (5.1) ggf. die Bahn weiter verbessern. (5.1) und (5.2) werden abwechselnd so lange ausgeführt, bis sich keine Verbesserung der Bahn zeigt:

Listing 5.3: Löschen und Runden

```
wiederhole
  berechneLeistungsindex(alt)
  optimiereMitRunden
  optimiereMitLoeschen
  berechneLeistungsindex(neu)
bis leistungsindex(neu) = leistungsindex(alt) wiederhole_ende
```

Die nächste Optimierungsphase nutzt die Eigenschaften drei, sechs und sieben. Wegen der Eigenschaft sieben wird ein zufallsbasiertes Verfahren zum weiten Verschieben der Stützpunkte und Überwinden von lokalen Minima eingesetzt:

Listing 5.4: Zufallsverschiebung der Stützpunkte

```
wiederhole fuer k von 0 bis 100
        berechneLeistungsindex(alt)
        erzeugeZufallszahlZwischen0_1(z)
        wenn z < 0,2 dann
                sucheStuetzpunkMitGeringsterAequidistanz
                loescheStuetzpunkt(punktMitGeringsterAequidistanz)
        wenn_ende
        wenn z < 0,9 dann
                aendereZuSpline(alleBahnsegmente)
        wenn_ende
        wiederhole fuer alle Stuetzpunkte i
                verschiebePunkt(z * k / 100 * GELENKBEREICH)
        wiederhole_ende
        berechneLeistungsindex(neu)
        wenn nicht bahnkollidiert und leistungsindex(neu) <
            leistungsindex(alt) dann
                uebernehmeAktuelleBahn
        wenn_ende
wiederhole_ende
```

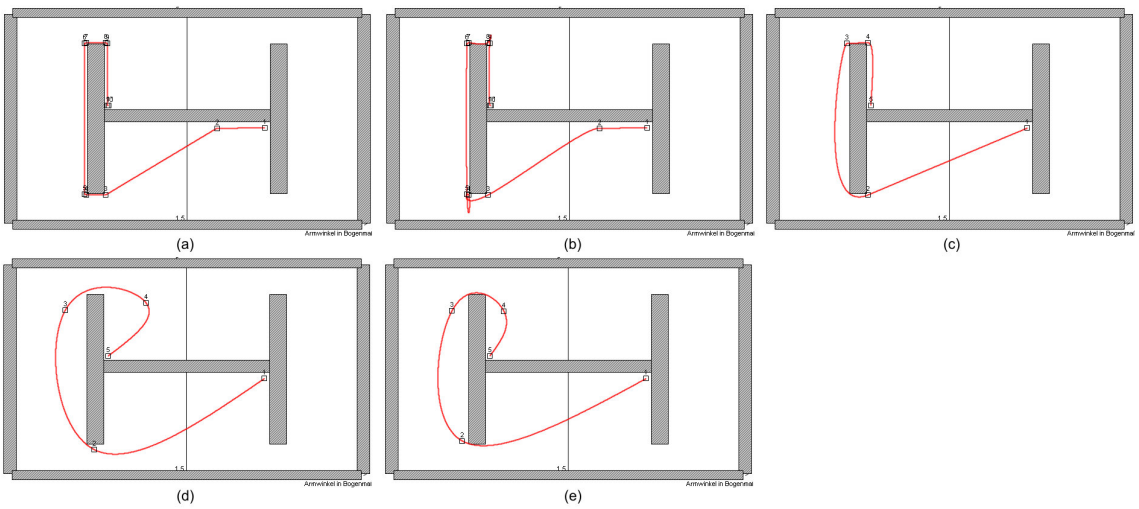


Abbildung 5.8: Bahnoptimierung auf Basis des Leistungsindex' im Benchmark *Double Trap* a) Ergebnis der Bahnplanung mit Wellenfront-Verfahren b) Resultat des Rundens aus Schritt (5.1) c) Bahn nach dem Löschen von unnötigen Punkten mit (5.2) d) nach der einmaligen Ausführung von (5.4), e) nach mehrmaliger Durchführung von (5.4) (aus Simulation)

Eigenschaft sechs verlangt jedoch ein geringes Verschieben der Punkte. Eigenschaft drei fordert das Entfernen von Stützpunkten, die zu ihren Nachbarn nicht äquidistant sind. Die konstanten Werte des abschließenden Optimierungsschritts (5.4) sind nur als Beispiele aufzufassen. Dieser Schritt kann weitere Stützpunkte mit einer schlechten Lage zu ihren Nachbarn entfernen, alle Bahnsegmente zu Splines ändern und Stützpunkte verschieben. Das Verschieben geschieht hauptsächlich in den Nahbereich der Punkte und selten in den kompletten Gelenkbereich.

5.2.2 Versuche in der Simulation

Die Funktionsweise der Bahnoptimierung wird mit Versuchen veranschaulicht. In der Simulation, die in Abschnitt 5.4 näher vorgestellt wird, werden in unterschiedlichen Szenarien Start, Ziel und Hindernisse definiert. Die Hinderniskonfigurationen in den verschiedenen Benchmarks sollen die Grenzen der Leistungsfähigkeit der Optimierung demonstrieren. Die Szenarien sind für einen Roboter mit 2 Gelenken z.B. den KHS-Einleger ausgelegt.

Abbildung 5.8 zeigt die einzelnen Phasen der Bahnoptimierung. Ausgehend von einer kollisionsfreien Bahn mit linearen Bahnsegmenten wird durch Runden, Löschen und Verschieben von Bahnteilen der Leistungsindex minimiert.

Abbildungen 5.9, 5.10 und 5.11 zeigen weitere Szenarien. Während das Benchmark in Abbildungen 5.9 zu erwarteten Ergebnissen führt, versagt das Optimierungsverfahren bei komplexen, engen Hinderniskonfigurationen und hoher Zahl an Stützpunkten wie in Abbildung 5.11. Die Bahn aus Abbildung 5.10c) wird nur gefunden, wenn die Gewichtung der Standardabweichung der Bahnkümmung entsprechend hoch ist.

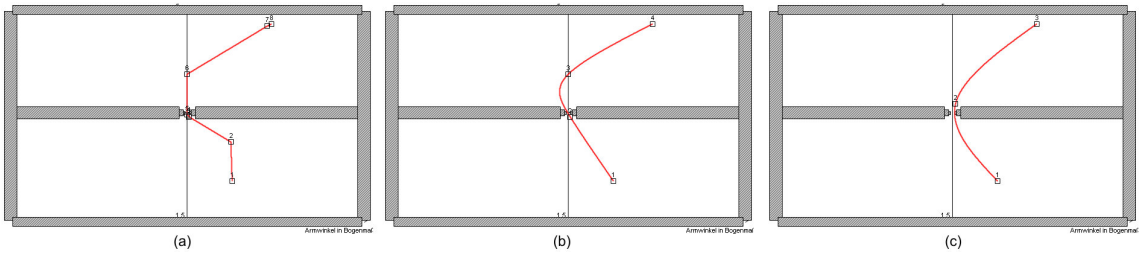


Abbildung 5.9: Das Benchmark *Bottleneck* mit seiner sehr kleinen Durchgangsöffnung stellt hohe Anforderungen an die Auflösung Bahnplanung a) Ausgangsbahn, b) Bahn nach (5.1) und (5.2), c) nach mehrmaliger Durchführung von (5.4) (aus Simulation)

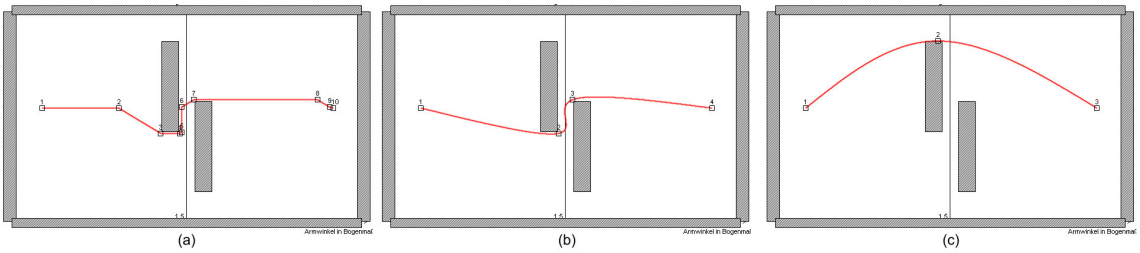


Abbildung 5.10: Die Optimierung führt zu einer besseren aber längeren Bahn a) Ausgangsbahn, b) Bahn nach (5.1) und (5.2), c) nach mehrmaliger Durchführung von (5.4) (aus Simulation)

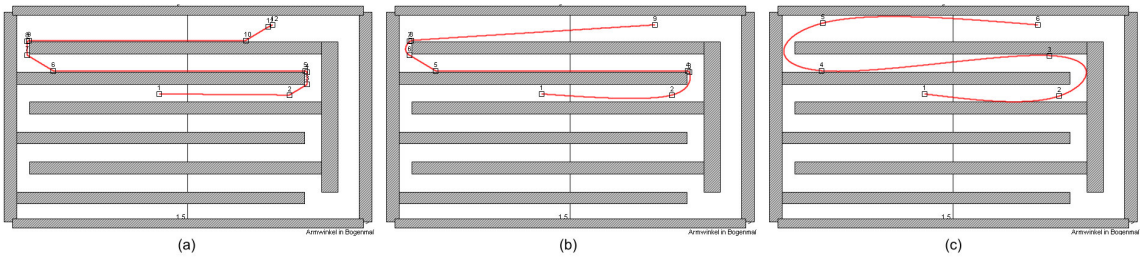


Abbildung 5.11: Benchmark *Detour* a) Ausgangsbahn, b) Bahn nach Durchführung aller Optimierungen, c) Die optimale Bahn wird nur mit manuellem Verschieben der Stützpunkte erreicht (aus Simulation)

5.2.3 Aufwandabschätzung und Bewertung

Da das vorgestellte Verfahren lediglich aus Beobachtungen der Simulation abgeleitet wurde, ist nachzuweisen, ob es effizienter als die analytische Optimierung ist und ob der Leistungsindex ein ausreichend verlässlicher Indikator für die zu optimierende Fahrdauer ist.

Der Leistungsindex setzt sich aus der Bahnlänge und der Bahnkrümmung zusammen. Beide Größen können während der numerischen Bahninterpolation in $\mathcal{O}(m \cdot n)$ berechnet werden. Für Schritt (5.1) ergibt sich $\mathcal{O}(m \cdot n^2)$, Schritt (5.2) $\mathcal{O}(m \cdot n^2)$ und Schritt (5.4) $\mathcal{O}(m \cdot n)$. Der Gesamtaufwand liegt in $\mathcal{O}(m \cdot n^2)$ und ist geringer als der Aufwand des analytischen Verfahrens mit $\mathcal{O}(i \cdot d \cdot m \cdot n^2 \cdot \log(n_a\, n_{grenz}))$. Zudem berücksichtigt die $\mathcal{O}$-Notation nicht, dass die zur Ermittlung der Fahrzeit benötigten Funktionen komplizierter als die für den Leistungsindex sind.

Ist die Korrelation zwischen Leistungsindex Idx und Fahrdauer t hoch, so ist der Quotient $f_k = \frac{Idx}{t}$ annähernd konstant. Als Gütekriterium wird im folgenden Versuch die Standardabweichung von f_k berechnet. Im Double Trap-Szenario werden durch einen Zufallsgenerator Bahnen erzeugt.

Messung 1: 300 beliebige, kollisionsfreie Bahnen mit 3 bis 5 Stützpunkten
$f_{k,1} = \mu_1 = 110{,}44$; $\sigma_{f_k,1} = 51{,}90$

Messung 2: 50 beliebige, kollisionsfreie Bahnen mit 3 bis 5 Stützpunkten, die nach der Bahnoptimierung noch genau 3 Stützpunkte besitzen
$f_{k,2} = \mu_2 = 98{,}95$; $\sigma_{f_k,1} = 23{,}84$

Die beiden Messungen ergeben, dass die Fahrzeit von Bahnen mit weniger Stützpunkten durch den Leistungsindex besser abgeschätzt werden kann als Bahnen mit mehr Stützpunkten. Besitzt eine optimierte Bahn noch 3 Stützpunkte, liegt das Verhältnis von Leistungsindex und Fahrzeit f_k mit einer Wahrscheinlichkeit von 68,3% in dem Intervall $[75,11;\ 122,79]$, bei beliebigen, nicht-optimalen Bahnen liegt f_k mit einer Wahrscheinlichkeit von 68,3% in dem Intervall $[58,54;\ 162,34]$. Diese Stichprobe lässt auf eine allgemein hinreichend genaue Beschreibung der Fahrdauer durch den Leistungsindex schließen. Diese Approximation wird genauer, je weiter die Optimierung fortgeschritten ist und Stützpunkte entfernt hat.

5.3 Optimierung für SPS-Einsatz und Handhabungsaufgaben

Ein Problem der praktischen Implementierung auf einer SPS ist der hohe Speicherbedarf der numerisch gespeicherten Trajektorien aus Abschnitt 4.4.2. Die numerische Speicherung dient einer prinzipiell gleichen Realisierung der s- und t-Bahnregelung durch eine einfache Suche in der Trajektorientabelle. Zudem kann die numerische Bahn im Voraus leicht auf Randbedingungen wie Stetigkeit der Bahnkontur und Einhaltung der Drehmomentgrenzen und Drehzahlen überprüft werden. Für den

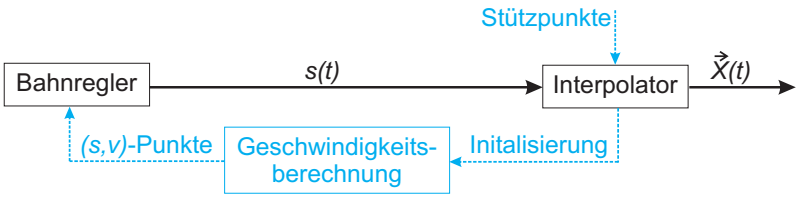


Abbildung 5.12: Die speicheroptimierte Trajektoriengenerierung verzichtet auf die numerische Speicherung und interpoliert online. Der Bahnregler erhählt bei der Initialisierung eine Beschreibung des Geschwindigkeitsprofils.

Einsatz auf einer SPS müssen Bahninterpolation und -regelung durch speichereffiziente Verfahren ersetzt werden.

Bei Handhabungsaufgaben wird nicht nur das überschwingungsfreie Erreichen der Ziellage sondern auch ein bestimmter Anfahrtswinkel zum Ziel z. B. beim Greifen oder dem Kopfwechsel (Werkzeugwechsel der KHS-Maschinen) gefordert. Die bisherige Interpolation mit kubischen Splines berücksichtigt diese Randbedingung nicht. Splines bieten aber wie in Abschnitt 4.2.1 beschrieben zwei offene Randbedingungen, um einen eingespannten Rand mit vorgegeben Bahnrichtungen im Start- und Endpunkt der Bahn zu realisieren.

Der praktische Einsatz der KHS-Roboter sieht nicht nur die Planung und Fahrt einer einzigen Bahn vor sondern je nach den veränderlichen Randbedingungen wie die Lage der Hindernisse und der Handhabungsaufgabe sollen Bahnen verkettet werden. Ein typisches Szenario ist der Transport einer Last zum Ziel mit der anschließenden Leerfahrt. Hat sich vor der Leerfahrt die Hinderniskonfiguration geändert, so ist die Planung einer neuen Bahn notwendig. In KHS-Anwendungen ist die neue Hinderniskonfiguration meist im Voraus bekannt und die Planung der Folgebahn mit einem glatten Übergang kann ebenfalls im Voraus stattfinden.

5.3.1 Speichereffiziente online-Interpolation

Die Bahnspeicherung kann durch eine online-Interpolation ersetzt werden. Wie in Abbildung 5.12 bildet der Interpolator den Bahnparameter s auf die Lage der Roboterhand während der Fahrt ab. Für eine zeitoptimale Fahrt muss der Bahnregler einen entsprechenden $s\,(t)$-Verlauf bereitstellen. Außerdem muss der Interpolator bei der kubischen Spline-Interpolation den Bahnparameter s korrekt auf das Interpolationssegment i und den lokalen Bahnparameter t abbilden, obwohl die komplette Raumkurve nicht bekannt ist.

Ordnet der Interpolator den Bahnparameter s dem zu interpolierenden Bahnpunkt über Geradenapproximation des Polygonzugs zu

$$\Delta s_i \approx \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (x_{i+1} - z_i)^2}\,, \tag{5.4}$$

so ergeben sich wie in Abbildung 5.13 dargestellt Ungenauigkeiten. Vor allem in Bahnsegmenten mit hohen Krümmungen ist die Abweichung der interpolierten Kur-

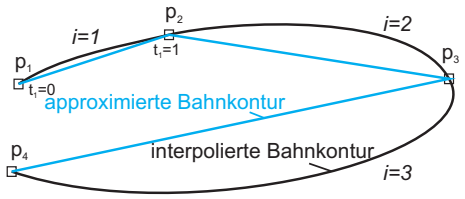


Abbildung 5.13: Der durch Geraden approximierte Bahnparameter s ist ungenau in Bereichen mit hoher Bahnkrümmung und führt zu Fehlern bei der Zuordnung der Bahngeschwindigkeit zum Bahnparameter .

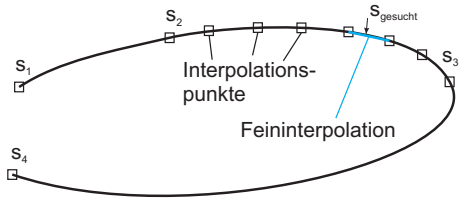


Abbildung 5.14: Den Bahnstützpunkten p_i wird die exakte Bahnlänge s_i zugeordnet, innerhalb eines Bahnsegments findet eine numerische Integration der Bahnlänge über wenige Zwischenstützpunkte statt, zwischen zwei Zwischenpunkten wird eine lineare Feininterpolation verwendet.

ve und die Kurvenlänge deutlich größer als die Approximation. Dieser Fehler vergrößert sich durch die Integration entlang der kompletten Bahnkontur und resultiert in einer verzerrten Zuordnung der Bahngeschwindigkeit zu der Bahnposition.

Als Lösung werden in einer ersten Variante die einzelnen Bahnsegmente bei der Initialisierung numerisch integriert und die Bahnlänge den Bahnstützpunkten zugeordnet. Somit kann im Interpolator dem Bahnparameter schnell das korrekte Bahnsegment i zugeordnet werden. Anschließend muss eine numerische Integration der Bahn dieses Bahnsegments stattfinden, um den lokalen Bahnparameter t auf den Bahnpunkt $\vec{X}$ abzubilden. Abbildung 5.14 stellt das Verfahren dar.

Eine weitere Variante dieses Verfahrens ohne die zeitaufwändige Integration über das Bahnsegment stellt die einfache lineare Interpolation des lokalen Bahnparameters dar:

$$t = \frac{s - s\left(p_i\right)}{s\left(p_{i+1}\right) - s\left(p_i\right)} \,. \tag{5.5}$$

Die Bahnstützpunkte werden bei dieser Variante exakt angefahren, die mit den Stützpunkten verknüpfte Bahnstecke und die mit s-Regelung gefahrene Durchlaufzeit der Bahn sind die gleichen wie mit der numerisch gespeicherten Bahn. Jedoch zeigt sich bei dieser Interpolation eine nicht konstante Bahngeschwindigkeit zwischen Bahnstützpunkten bei einem konstanten $\dot{s}$ des Bahnreglers. Die erste Variante zeigt bei der s-Regelung die gleiche Bewegung der Roboterhand wie mit numerisch ge-

speicherten Trajektorien und ist der zweiten Variante vorzuziehen. Bei Steuerungen mit wenig Rechenleistung kann die zweite Variante eingesetzt werden.

5.3.2 Umsetzung der Geschwindigkeit direkt im Bahnregler

Nach der Initialisierung der Interpolation findet die Geschwindigkeitsberechnung statt. Die Fahrt über die Bahn wird simuliert und die online-Interpolation liefert die für die Dynamik- und Geschwindigkeitsberechnung nötigen Gelenkzustände. Auf Basis der Ergebnisse der Geschwindigkeitsberechnung wurden die Beschleunigungsprofile aus Abschnitt 4.3.1 integriert und die Bahngeschwindigkeit dem Bahnparameter in der Trajektorientabelle zugeordnet. Die t-Regelung führte zu der zeitoptimalen Abfahrt der Trajektorie.

Soll auf die Trajektorientabelle verzichtet werden, können die (s, v)-Stützpunkte direkt im Bahnregler mit folgender, einfachen Erweiterung des Reglers verwendet werden:

1. suche den (s_i, v_i)-Punkt im (s, v)-Profil mit $s_i > s_{ist}$ und $|s_i - s_{ist}|$ ist minimal
2. verwende v_i als v_{max} des Bahnreglers

Mit diesem Verfahren wird der jeweils nächste Punkt des (s, v)-Geschwindigkeitsprofils als Reglerparameter verwendet und die maximale Bahngeschwindigkeit in diesem Zeitabschnitt $v_{max} = v_i$ verwendet. Die Ruckbegrenzung der Beschleunigungsprofile wird nun durch die Ruckbegrenzung des Bahnreglers ersetzt.

Ein Nachteile dieser Methode ist, dass eine Bahn nicht mit dem exakt gleichen Geschwindigkeitsverlauf rückwärts gefahren werden kann, da der Bahnregler immer verzögerungsfrei auf v_{max} regelt. Außerdem können direkt nach Minima im Grenzgeschwindigkeitsprofil mit der verzögerungsfreien Beschleunigung die dynamischen Grenzen des Roboters verletzt werden. Dies wurde durch Beschleunigungsprofile mit verzögerter Beschleunigung aus Abschnitt 4.3 behoben. Bei der online-Berechnung kann als Lösung neben einem Grenzgeschwindigkeitsminimum ein weiterer Stützpunkt $(s_i + \Delta s, v_i)$ mit gleicher Geschwindigkeit eingefügt werden. Dieser Stützpunkt hält die Bahngeschwindigkeit über die kurze Strecke Δs konstant und sorgt ebenfalls für eine verzögerte Beschleunigung.

Konsequenzen für die Bahnplanung

Die Bahnplanung benötigt die Auswertung des Leistungsindex' zur Beurteilung der Güte einer Bahn. Da die online-Interpolation zur Berechnung der kompletten Bahn für die Geschwindigkeitsberechnung während der Initialisierung ausgeführt wird, können mit nur geringem zusätzlichen Aufwand die Bahnkrümmung und -länge bestimmt und der Leistungsindex berechnet werden.

Für die Bahnoptimierung müssen lediglich wenige die Bahn beschreibenden Stützpunkte und der Leistungsindex gespeichert werden. Das Speichern der kompletten, interpolierten Bahnen ist nicht nötig.

Abbildung 5.15: Spline-Interpolation mit vorgegebenen Bahnrichtungen in den Endpunkten (aus Simulation)

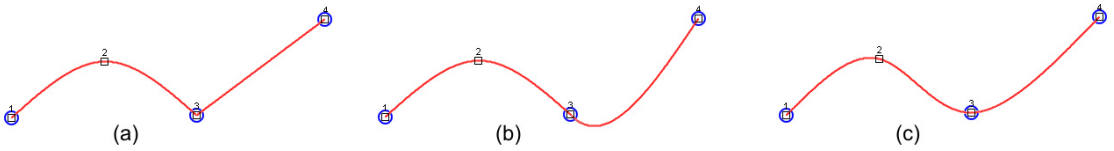


Abbildung 5.16: Verkettung von Bahnen mit a) separierter Planung beider Bahnen und Knick im gemeinsamen Stützpunkt b) Anpassung des Startwinkels der Folgebahn an die vorausgehende Bahn und c) Planung der beiden Teilbahnen als eine gemeinsame Bahn (aus Simulation)

5.3.3 Vorgabe des Anfahrtswinkels bei Stützpunkten

Bei Handhabungsaufgaben kann ein festgelegter Anfahrtswinkel zum Ziel erforderlich sein, z. B. wenn ein Objekt von oben gegriffen werden soll. Abbildung 5.15 stellt eine Bahn mit vorgegebenen Bahnrichtungen im Start- und Endpunkt dar. Zusätzlich zur Lage müssen die Endpunkte Informationen zum Winkel besitzen.

Der Quellcode zur Initialisierung der gemischten Spline- und Linearinterpolation ist in Anhang D dargestellt. Für den Spline jeder Gelenkdimension können die Werte der Ableitung am Splinebeginn und -ende vorgegeben werden. Werden keine Vorgaben gemacht, wird ein natürlicher Spline, dessen Enden gegen eine Gerade konvergieren, erzeugt.

5.3.4 Verkettung von Bahnen und Makrostützpunkte

In industriellen Roboteranwendungen sind Zielpositionen, an denen die Maschine bestimmte Arbeiten verrichten soll, meist im voraus bekannt und berechenbar. Die Anfahrtsposition der Palette bei KHS-Beladern ändert sich z. B. bei jeder neuen Lage um die Lagenhöhe. Mit den bisherigen Verfahren kann für diesen Anwendungsfall eine Bahn bis zum Boden erstellt und mit dem Bahnregler jeweils unterschiedlich weit gefahren werden. Ändert sich jedoch nach dem Erreichen eines Ziels die Hindernisanordnung, so muss eine neue Bahn erstellt werden. Um eine Leerlaufzeit des Roboters am Ziel wegen der zeitaufwändigen Bahnberechnung zu vermeiden, soll die neue Bahn während der Fahrt der aktuellen Bahn berechnet und mit ihrem Ende verkettet werden.

An den Ziel- und Startpositionen der einzelnen Bahnen, den **Makrostützpunkten**, entsteht ein Bahnübergang. Für den Übergang ergeben sich drei Varianten:

1. Die Folgebahn wird getrennt erstellt,

2. der Startwinkel der Folgebahn wird an die vorausgehende Bahn angeglichen oder

3. alle Teilbahnen werden als gemeinsame Bahn interpoliert.

Die Abbildung 5.16 stellt die drei Möglichkeiten an einem Beispiel mit vorgegebenen Makrostützpunkten dar. Die erste Variante erzeugt keinen C_2-stetigen Übergang der Bahnen, aber sie kann während der Fahrt der ersten Teilbahn berechnet werden, da die angehängte Bahn keine Rückwirkung auf die vorherige ausübt. Die praktische Anwendung liegt z. B. im Stapeln von Lagen auf einer Palette mit einem Hubsäulenroboter, der beim Ablegen eine Lage kurz still steht. Die zweite Variante führt zu einem glatten Übergang der Bahnen im Makrostützpunkt, indem der Rand des folgenden Splines angepasst wird. Dieses Verfahren kann in Schiebern von Beladem aschinen Anwendung finden, bei denen aus der Bewegung heraus ein neues Ziel angefahren werden soll. Die dritte Möglichkeit erzeugt einen Übergang mit geringerer Bahnkrümmung. Diese Bahn ist aus Optimierungssicht die beste, aber sie muss komplett im voraus bei Stillstand des Roboters geplant werden, da sich die Gestalt der Folgebahn auf den Verlauf der kompletten Bahn auswirkt.

Die Umsetzung der Verkettung basiert auf der mehrfachen Ausführung der Bahnplanung mit den Makrostützpunkten als Start und Ziel und mit ggf. unterschiedlichen Hindernissen je Bahn. Der glatte Übergang zwischen den Bahnen ergibt sich aus der Vorgabe des Anfahrtswinkels des vorherigen Abschnitts.

5.4 Simulation

Eine von der Hardware losgelöste Simulation erleichtert die Implementierung von Anwendungen, so dass die Kernfunktionalität früher realisiert werden kann. Mit einer Simulation können Robotermodelle der Kinematik, Dynamik und Physik für die spätere Implementierung in numerische Algorithmen überführt und getestet werden. Auf Basis der Simulationsergebnisse werden Testkonzepte und Benchmarks für den Roboter erstellt.
Im Folgenden werden verfügbare Robotersimulationen beschrieben und auf die Eigenentwicklung einer Simulation eingegangen.

5.4.1 Verfügbare Robotersimulationen

Dixon, Moses, Walker und Dawson [Dawson 01] erläutern vier Alternativen für Softwareplattformen: Proprietäre Roboter-Steuerungssprachen vom Hersteller des Roboters, Eigenentwicklung von Frameworks wie MCA in Hochsprachen, grafische Programmierumgebungen und Bibliotheken für gängige Sprachen zur Anpassung an wechselnde Hardware. In [Dawson 01] wird wegen der Plattform-Unabhängigkeit, der Vermeidung der Controller-Programmierung und der Erweiterbarkeit unter den Alternativen für die Implementierung eines eigenen Robotik-Toolkits für Mathlab/Simulink entschieden. Es bietet eine grafische Benutzerschnittstelle (GUI) an und ist echtzeitfähig. Es ist jedoch für den PUMA 560 ausgelegt. Dadurch ist die Übertragung auf einen anderen Roboter mit erheblichem Aufwand und nur bei einer offenen Architektur der Zielhardware möglich.

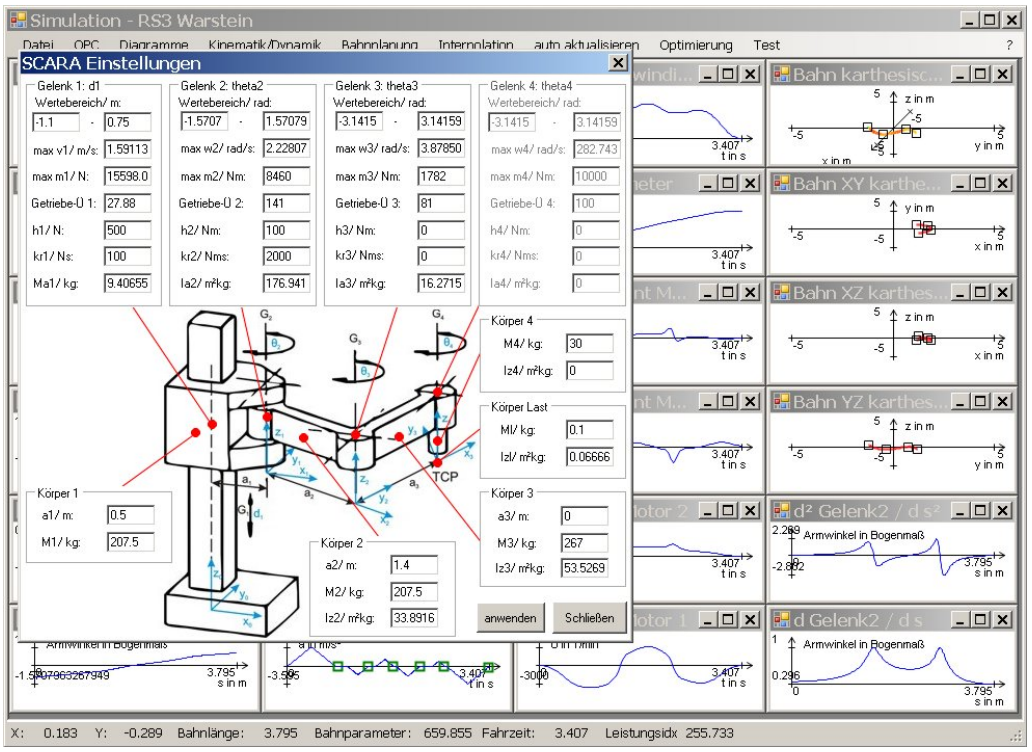


Abbildung 5.17: Screenshot der eigenen Simulation

MCA (Modular Controller Architecture) [MCA 07] dient als Framework für die Steuerung mobiler Roboter. Es ist in C++ implementiert und besitzt eine offene Architektur. MCA bietet eine 3D-Simulation für die Fahrt des Roboters in einer virtuellen Umgebung an. Da die Simulation fest mit dem Framework verknüpft ist, ist eine Anpassung auf die KHS-Roboter ebenfalls aufwändig. Zudem liegt ihre Hauptaufgabe in der Simulation der Roboter-Hardware und Sensorik. Eine Simulation für KHS-Roboter sollte die Effekte der Bahnplanung und -interpolation aufzeigen und weniger die Virtualisierung der Hardware.
Microsoft bietet mit Robotics Studio [Microsoft 07] ein .NET-Werkzeug für die 3D-Visualsierung und Ansteuerung von Robotern an. Als Programmiersprache können alle Sprachen von Microsoft Visual Studio und eine neue Sprache mit grafischer Notation (Visual Programming Language, VPL) verwendet werden. Robotics Studio beinhaltet eine Hardwareanbindung zu Lego Mindstorm-Robotern und ist mehr dem Hobby-Bereich als der Industrierobotik zugeordnet. Es bietet eine hardwarebeschleunigte Simulation der Physik an. Robotics Studio bietet keine vorgefertigten Komponenten für die Bahnplanung.

Die vorgestellten Simulationen sind nur mit großem Aufwand an die Bahnplanung anpassbar, deshalb wird die Simulation aus Abbildung 5.17 komplett neu erstellt. Zudem ist weniger eine Simulation der KHS-Roboter gewünscht, sondern eine Visualisierung der internen Vorgänge der Bahnplanung. Die Visualisierung soll sowohl zur Verifikation der bisher entwickelten Rechenmodelle als auch zur Gewinnung neuer Erkenntnisse aus der Bahnplanung dienen.
Als Programmiersprache wird Microsoft Visual Basic gewählt. Sie erlaubt die zügi-

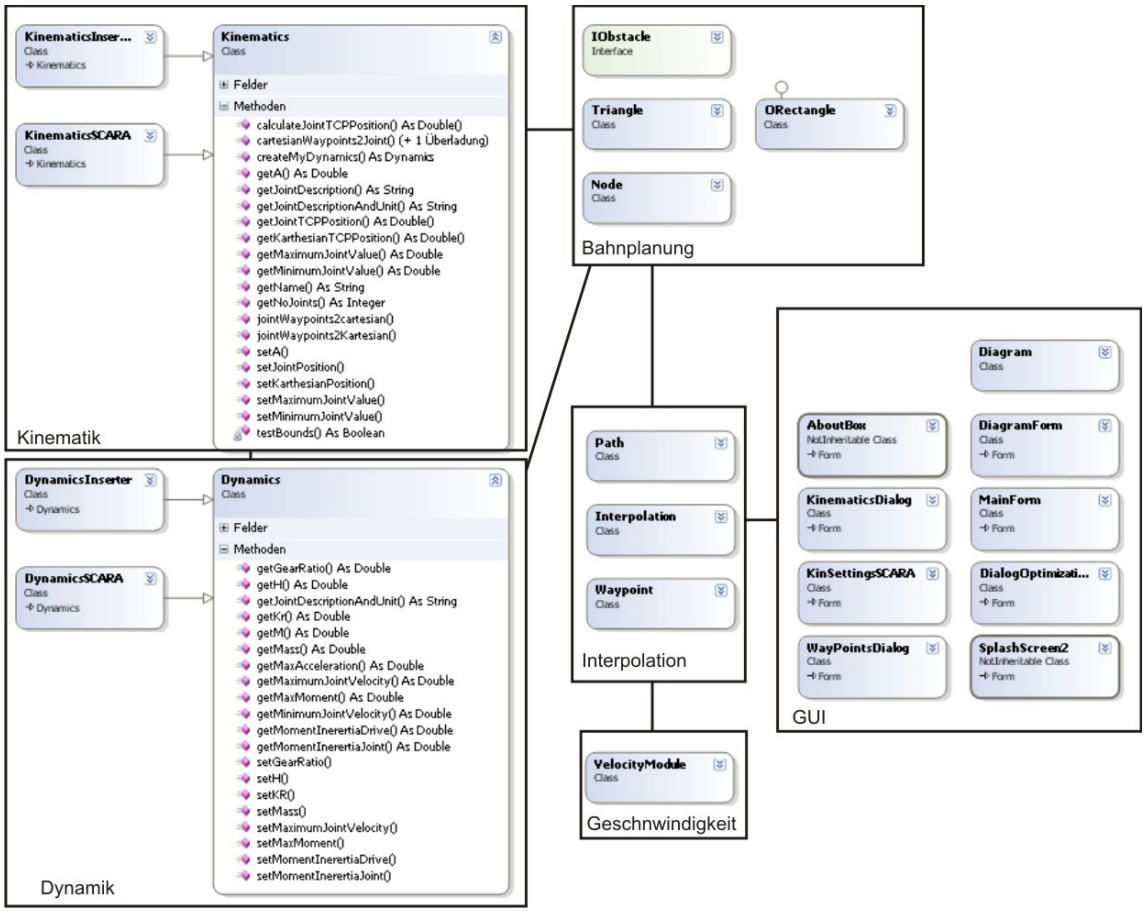


Abbildung 5.18: Softwarearchitektur der Simulation

ge Entwicklung von einfachen Anwendungen mit ansprechender Benutzeroberfläche. Zudem kann der Quellcode von der KHS weiterentwickelt werden, da die HMI der KHS-Anlagen ebenfalls mit Visual Basic erstellt werden und Erfahrungen mit dieser Programmiersprache vorhanden sind.

5.4.2 Softwarearchitektur der eigenen Simulation

Im folgenden wird kurz die Architektur der Simulation beschrieben. Eine Übersicht der Funktionen ist im Anhang E dargestellt.

In der Softwarearchitektur aus Abbildung 5.18 spiegeln sich die einzelnen Modelle des Roboters wider: Kinematik, Dynamik, Bahnplanung, Interpolation, Zeitprofile, etc. Diese Module werden sich in abgeänderter Form in der Robotersteuerung wieder finden.

- Die Pakete **Kinematik** und **Dynamik** implementieren die direkte und inverse Kinematik und die Bewegungsgleichung.
- **Bahnplanung** erstellt Stützpunkte aus der Beschreibung des Hindernisraums.
- Das Paket **Interpolation** basiert auf den Wegpunkten des Typs WayPoint und erzeugt eine interpolierte Bahn.

- Das Paket **Geschwindigkeit** nimmt die Erstellung der Zeitprofile vor.
- Die **GUI** ist für alle Benutzereingaben und die Darstellung der Diagramme verantwortlich.

5.5 Fazit

In diesem Kapitel wurde die Bahnplanung mit Hilfe einer neu entwickelten Simulation optimiert.

Die von Park und Bobrow [Bobrow 98] vorgestellten Methoden zur Optimierung der Durchlaufzeit der Trajektorie wurden mit den numerischen Algorithmen von Press [Press 92] implementiert und bestätigt. Falls die Lage mehrerer Stützpunkte optimiert werden soll, zeigte diese aufwändige Optimierung jedoch nicht die gewünschten Ergebnisse und kann in lokalen Minima terminieren.

Um dieses Problem zu lösen wurde ein eigenes Verfahren zur Bahnoptimierung entwickelt, das bereits bei der geometrischen Kontur der Bahn ansetzt und auf die rechenintensive Auswertung der Fahrdauer verzichtet. Seine Leistunsfähigkeit wurde sowohl mit theoretischer Betrachtung nachgewiesen als auch in der Simulation demonstriert.

Da die numerische Speicherung interpolierter Bahnen in einer SPS mit begrenztem Speicher nicht praktikabel ist, wurden die Verfahren der Interpolation, Geschwindigkeitsberechnung und Bahnregelung modifiziert. Die Bahnplanung kann mit minimalem Speicher arbeiten, jedoch erhöht sich die Rechenlast.

6. Softwareentwurf der Robotersteuerung

Nachdem die vorgestellten Modelle und Verfahren mit der Simulation abgesichert wurden, können sie unter Einbezug der Randbedingungen der Kuka-Steuerung auf das Zielsystem übertragen werden. Für die Implementierung muss herausgearbeitet werden, welche Teile der Modellierung und der Simulation auf die SPS portiert werden können und welche Funktionen online, einmalig beim Start der Maschine oder offline mit einem externen Programm berechnet werden sollen.

Neben der Kernfunktionalität muss das Steuerungsprogramm den manuellen Eingriff des Bedienpersonals und ein manuelles Fahren ermöglichen. Da nicht nur Software-Entwickler sondern auch Mitarbeiter mit weniger Expertise wie Monteure auf der Außenbaustelle, das Service-Personal oder sogar der Kunde die Steuerung beeinflussen möchten, ist das Bahnplanungssystem leicht verständlich und leicht anpassbar zu implementieren.

Die folgende Gliederung und der Entwicklungsprozess ist an das V-Modell XT der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik

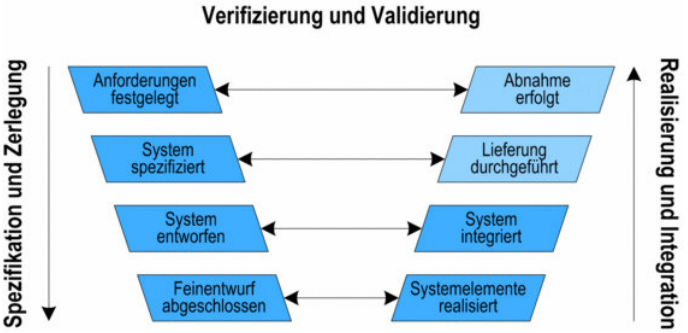


Abbildung 6.1: Zerlegungsschritte in der Systementwicklung des V-Modell-XT, aus [KBSt 07] S. 36

in der Bundesverwaltung [KBSt 07] angelehnt, wobei aufgrund des geringen Umfangs des Projekts die V-Modell-Vorgaben nicht komplett umgesetzt wurden. Das V-Modell ist ein modularisiertes Vorgehensmodell für das Projektmanagement und die Systementwicklung. Abbildung 6.1 zeigt eine mögliche Abfolge von Entscheidungspunkten bei der Systementwicklung.

Als Modellierungssprache für den Entwurf wird die UML (Unified Modelling Language) eingesetzt. UML bietet im Vergleich zu ER-Modellierung (Entity Relationship) oder SDL (Specification and Description Language) differenziertere Ausdrucksmöglichkeiten für die Bescheibung des folgenden Entwurfs. UML unterstützt mit unterschiedlichen grafischen Notationen die Modellierung des zu erstellenden Systems aus den unterschiedlichen Sichten Anforderungen, Struktur und Verhalten. Sommerville [Sommerville 01] gibt eine Einführung in die Beschreibungstechniken.

Wesentliche Inhalte der Systemspezifikation in **Abschnitt 6.1** sind die funktionalen und nicht-funktionalen Anforderungen an das zu entwickelnde Gesamtsystem. Eine erste Grobarchitektur des Systems wird entwickelt und in einer Schnittstellenübersicht beschrieben. Das zu entwickelnde System sowie weitere zu entwickelnde Unterstützungssysteme werden identifiziert und den Anforderungen zugeordnet.

Ausgehend von den funktionalen und nicht-funktionalen Anforderungen an das System wird eine geeignete Systemarchitektur in **Abschnitt 6.2** entworfen. In einem ersten Schritt werden richtungsweisende Architekturprinzipien festgelegt und mögliche Entwurfsalternativen untersucht. Entsprechend der gewählten Entwurfsalternative wird die hierarchische Zerlegung (Dekomposition) des Systems in einzelne Elemente durchgeführt.

Abschnitt 6.3 zeigt abschließend Implementierungsaspekte und Konzepte von Algorithmen auf.

6.1 Spezifikation

Die Spezifikation beinhaltet die Anforderungen an das zu entwickelnde System und seine Einordnung in das Gesamtsystem.

6.1.1 Funktionale Anforderungen

In diesem Abschnitt werden die funktionalen Anforderungen an das neu zu entwickelnde System beschrieben.

Anwendungsfall-Überblick

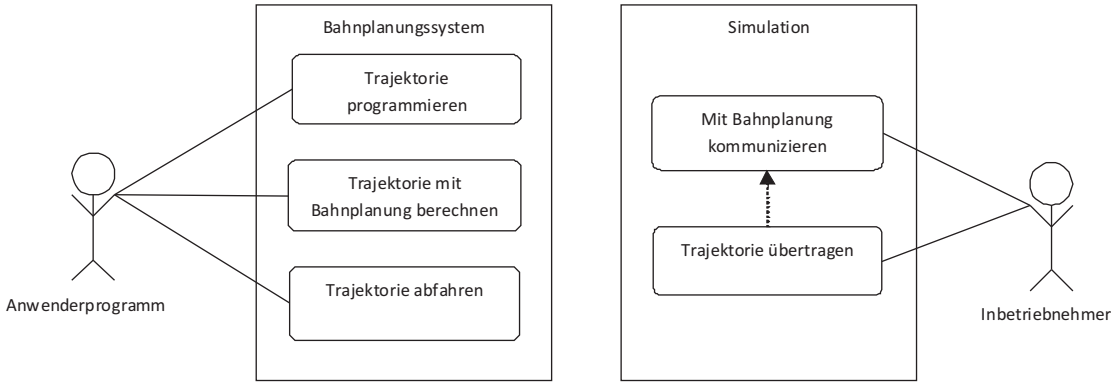


Abbildung 6.2: Anwendungsfall-Überblick mit dem zu entwickelnden System *Bahnplanungssystem* und *Simulation*, den Aktoren *Anwenderprogramm* und *Inbetriebnehmer*

Das Anwendungsfalldiagramm in Abbildung 6.2 beschreibt eine externe, funktionale Sicht auf das System. Als Aktoren treten das **Anwenderprogramm** und der **Inbetriebnehmer** auf. Das **Anwenderprogramm** soll verschiedene Trajektorien erzeugen können, dem **Inbetriebnehmer** soll über die Simulation eine Visualisierung der Bahnplanung zur Verfügung gestellt werden.

Übersicht aller funktionalen Anforderungen

FAF1 Trajektorie programmieren
Kurzbeschreibung: Das Bahnplanungssystem muss das Programmieren von Bahnstützpunkten und Randbedingungen wie maximaler Bahnruck ermöglichen.
Akteur: Anwenderprogramm
Vorbedingungen: Kinematische Parameter (und dynamische für zeitoptimales Fahren) müssen gesetzt sein
Eingabe: Stützpunkte mit Gelenkstellungen, Bahngeschwindigkeit, -ruck, -beschleunigung sowie allgemeine maximalen Bahnruck, maximale Bahnbeschleunigung
Standardablauf: 1. Das Anwenderprogramm übergibt der Bahnplanung die Stützpunkte. 2. Das Bahnplanunggsystem überführt die Eingaben in eine interne Repräsentation. Es führt anschließend die Bahninterpolation durch. 3. Die erfolgreiche Berechnung wird quittiert.
Ausnahmen: ungültige Lage von Stützpunkte → Beschränkung durch die kinematischen Endanschläge
Ausgabe: Quittierung der Berechnung
Nachbedingungen: Die Bahn ist bereit für die Fahrt.
Beschreibung: Das Anwenderprogramm kann eine Tajektorie über Stützpunkte und Randbedingungen definieren. Danach wird die Interpolation zur Ermittlung notwendiger Bahngrößen wie Bahnlänge und die Geschwindigkeitsberechnung ausgeführt.

FAF2 Trajektorie mit Bahnplanung berechnen
Kurzbeschreibung: Aus einer Hindernisbeschreibung und dem Ziel muss das Bahnplanungssystem eine kollisionsfreie Trajektorie liefern.
Akteur: Anwenderprogramm
Vorbedingungen: Kinematische Parameter (und dynamische für zeitoptimales Fahren) müssen gesetzt sein
Eingabe: Hinderniskanten im Gelenkwinkelraum, Ziel- und Startlage
Standardablauf: 1. Das Anwenderprogramm übergibt der Bahnplanung die begrenzenden Kanten der Hindernisse, Start und Ziel. 2. Das Bahnplanungssystem überführt die Eingaben in den Konfigurationsraum, sucht und optimiert mit dem Wellenfront-Verfahren eine Trajektorie. 3. Geschwindigkeitsberechnung 4. Die erfolgreiche Berechnung wird quittiert.
Ausnahmen: ungültige Lage von Stützpunkten → Beschränkung durch die kinematischen Endanschläge oder Fehler, wenn Bahn nicht möglich
Ausgabe: Quittierung der Berechnung
Nachbedingungen: Die Bahn ist bereit für die Fahrt.
Beschreibung: Das Anwenderprogramm kann eine Tajektorie über den Ziel-Stützpunkt, Randbedingungen und Hindernisse definieren. Die Bahnplanung führt zu einer kollisionsfreien Folge an Stützpunkte. Danach wird die Interpolation zur Ermittlung notwendiger Bahngrößen wie Bahnlänge und die Geschwindigkeitsberechnung ausgeführt.

FAF3 Trajektorie abfahren
Kurzbeschreibung: Der Bahnregler muss in s- oder t-Regelung die Stellungen der einzelnen Gelenke für die Antriebe liefern.
Akteur: Anwenderprogramm
Vorbedingungen: Trajektorie wurde berechnet. Für t-Regelung: Dynamikparameter gesetzt und Geschwindigkeitsverlauf berechnet.
Eingabe: Parameter des Bahnreglers: $v_{max}, a_{max}, r_{max}, s_{soll}$, s- oder t-Regelung gewünscht
Standardablauf: 1. Das Anwenderprogramm übergibt dem Bahnregler eine neue Solllage auf der Bahn. 2. Der Bahnregler regelt den Sollzustand ein.
Ausnahmen: Bremsweg ist kürzer als der zur Verfügung stehende Fahrweg → Ziel wird überfahren.
Ausgabe: Quittierung beim Erreichen des Zielzustands
Nachbedingungen: Istzustand = Zielzustand.
Beschreibung: Das Anwenderprogramm kann über den Bahnregler die Trajektorie in s(einfach)- oder t(zeitoptimal)-Regelung abfahren. Der Bahnregler mit dem Interpolator liefern die Stellgrößen für die Einzelachssteuerung.

FAF4 Mit Bahnplanung kommunizieren
Kurzbeschreibung: Die Simulation muss Betriebsdaten aus dem Bahnplanungssystem auslesen und visualisieren können.
Akteur: Inbetriebnehmer
Vorbedingungen: Bahnplanungssystem auf dem Kuka-Rechner, OPC-Server und Simulation auf einem PC laufen und sind vernetzt.
Eingabe: Adressdaten der SPS-Steuerung und der Bahnplanung
Standardablauf: 1. Der Inbetriebnehmer verbindet die Simulation mit den Variablen des Bahnplanungssystems. 2. Die Simulation visualisiert Betriebsdaten wie aktuelle Bahn, Soll- und Ist-Position auf der Bahn.
Ausnahmen: Kommunikationsfehler → Fehlermeldung
Ausgabe: Anzeige von Stützpunkten des Bahnplanungssystems in der Simulation, aktuelle Soll- und Ist-Gelenkstellung
Nachbedingungen: -
Beschreibung: Die Kommunikation zwischen Bahnplanungssystem und Simulation über OPC gestattet eine Visualisierung der internen Vorgänge der Bahnplanung. Dies ist ein nützliches Instrument zur Überprüfung der Trajektorie vor der Fahrt.

FAF5 Trajektorie übertragen
Kurzbeschreibung: Die Simulation muss offline erstellte Trajektorien auf das Bahnplanungssystem übertragen können.
Akteur: Inbetriebnehmer
Vorbedingungen: siehe FAF4
Eingabe: siehe FAF4, Bahnstützpunkte aus Datei oder in der Simulation direkt erstellt
Standardablauf: 1. siehe FAF4 2. Der Inbetriebnehmer sendet die Stützpunkte an die SPS-Steuerung 3. Das Bahnplanungssystem fürt die Berechnungen aus FAF1 durch
Ausnahmen: Kommunikationsfehler → Fehlermeldung
Ausgabe: Quittierung in Bahnplanungssystem und Simulation
*Nachbedingungen:*Eine Bahn wurde dem Bahnplanungssystem hinzugefügt.
Beschreibung: Trajektorien können offline in der Simulation erstellt, optimiert, in Dateien gespeichert und auf das Bahnplanungssystem übertragen werden.

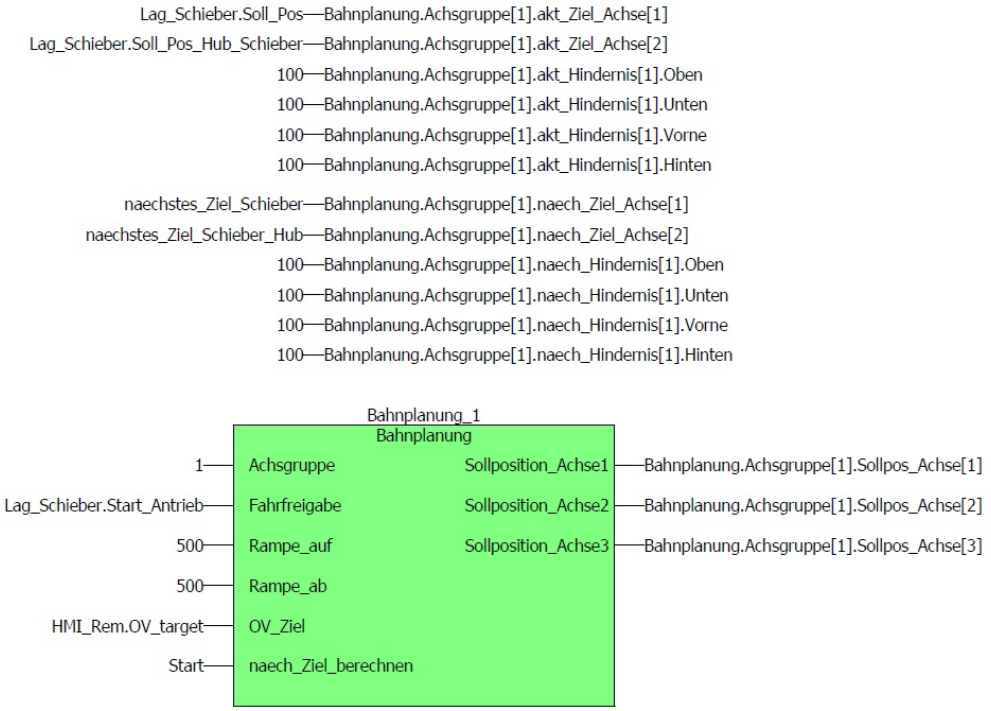
6.1.2 Nicht-funktionale Anforderungen

Die funktionalen Anforderungen des letzten Abschnitts klärten, *was* das zu erstellende System leisten soll. Die nicht-funktionalen Anforderungen beschreiben Randbedingungen und verfeinern die funktionalen Anforderungen. Hier wird das *wie* geklärt, eine Beschreibung der Qualität der Software. Ziel der Entwicklung ist die Überführung der Qualitätsanforderungen in quantifizierbare funktionale Anforderungen, die sie sich in der Implementierung widerspiegeln.

Übersicht aller nicht-funktionalen Anforderungen

NF1 Performanz
Kurzbeschreibung: Bahnen sollen während der Bewegung des Roboters berechnet und gewechselt werden können. Hierfür müssen die Verfahren eine kurze Berechnungsdauer aufweisen.
Erläuterung: Auf der Kuka-Plattform laufen nebenläufig das Steuerungsprogramm und Windows-Tasks. Die Bahnplanung muss Bahnen schnell (Größenordnung von Sekunden) berechnen und darf die übrigen Taks der Steuerung nicht merklich verzögern oder verdrängen. Eine Berechnung und Umschaltung zwischen den Bahn soll auch während der Fahrt möglich sein und die Bewegung nicht beeinträchtigen.
Lösungsidee: Die Kuka SoftSPS ermöglicht ein prioritätsbasiertes Scheduling. Der Bahnregler und weitere, zur Fahrt notwendige Komponenten, sollten eine hohe Priorität zugewiesen werden, während die Bahnberechnung im Hintergrund stattfinden.

NF2 Softwareproduktionsumgebung / Implementierungssprache
Kurzbeschreibung: Als SPU soll MultiProg von Kuka / KW-Software eingesetzt werden.
Erläuterung: MultiProg erlaubt das Entwickeln von SPS-Programmen nach DIN EN 61131-3 für Kuka SoftSPS-Systeme. Es dient auch als Programmier- und Debugwerkzeug.
Lösungsidee: Die erwähnten Komponenten verwenden

NF3 Wiederverwendbarkeit
Kurzbeschreibung: Das Bahnplanungssystem muss leicht wieder verwendet und erweitert werden können.
Erläuterung: Die Bahnplanung soll leicht in vorhandene Steuerungsprogramme integriert werden können. Die Software-Schnittstelle für das Anwendungsprogramm soll wie in folgender Abbildung gestaltet sein.    Hindernisse sollen über globale Variablen initialisiert werden. Der Bahnplanung wird anschließend nur das Ziel und einen Skalierungsfaktor der Geschwindigkeit vorgegeben. Das System soll zusätzlich auf mehrere Achsgruppen (Roboter) angewandt werden können und eine zweite, folgende Bahn im Voraus planen können.
Lösungsidee: Die Kernfunktionalität des Systems sollte als Black-Box in der Sprache ST implementiert werden. Dieser Code muss in der Regel nicht verändert werden. Vom Anwender zu parametrierende Vorgänge wie Auswahl der Kinematik, Initialisierung der dynamischen Parameter oder die Steuerung des Bahnreglers sollten in der leichter verständlichen, grafischen Funktionsbausteinsprache implementiert werden.

6.1.3 Lebenszyklusanalyse und Gesamtsystemarchitektur

Das Gesamtsystem besteht aus den zu erstellenden Systemen: **Bahnplanungssystem** und **Simulation**. Beide sind wie in Abbildung 6.3 vereinfacht dargestellt in die Umgebung eingebettet. Das **Bahnplanungssystem** kommuniziert mit dem **Anwenderprogramm** und der **Einzelachssteuerung** und befindet sich mit ihnen in der **SoftSPS**. Es tauscht über den **OPC-Server** Daten mit der **Simulation** aus.

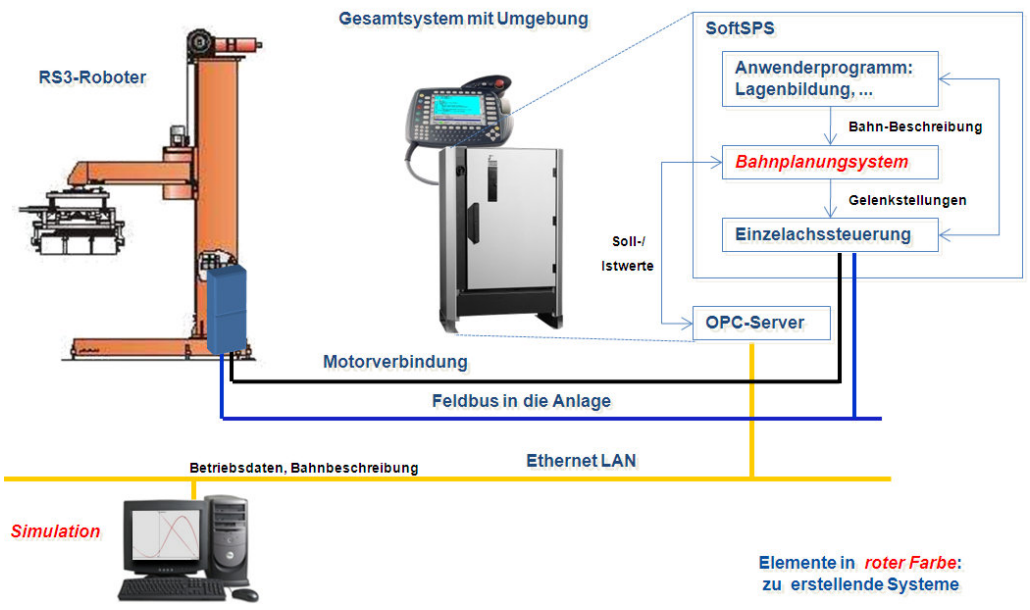


Abbildung 6.3: Gesamtsystemarchitektur

Schnittstellenübersicht

- Das Anwenderprogramm übergibt der Bahnplanung eine **Bahn-Beschreibung** in Form von Bahnstützpunkten, Randbedingungen und Hindernisbeschreibungen. Zudem steuert es die Abfahrt durch Vorgaben an den Bahnregler.

- Das Bahnplanungssystem übermittelt **Gelenkstellungen** und Motoreinstellungen an die Einzelachssteuerung.

- Das Bahnplanungssystem überträgt Werte interner Variablen mit **Soll- / Istwerten** über den OPC-Server an Office-Systeme und die Simulation.

- Der OPC-Server tauscht mit der Simulation **Bahnbeschreibungen** aus.

- Die Einzelachssteuerung regelt die Motoren des Roboters über die **Motorverbindung** und den **Feldbus**

6.2 Entwurf

Aus den Anforderungen der Spezifikation, der Gesamtsystemarchitektur und den Eigenschaften der mathematischen Modellierung leitet sich die Gliederung des zu erstellenden Systems ab. Zuerst werden Entwurfsalternativen untersucht und richtungsweisende Architekturprinzipien festgelegt. Basierend auf der Wahl des Architekturprinzips wird die Systemdekomposition entwickelt. Sie beinhaltet die funktionale Gliederung des Systems und die Schnittstellen der Elemente. Querschnittliche Eigenschaften wie ein übergreifendes Loggingkonzept oder die Behandlung von Ausnahmen werden außerdem behandelt.

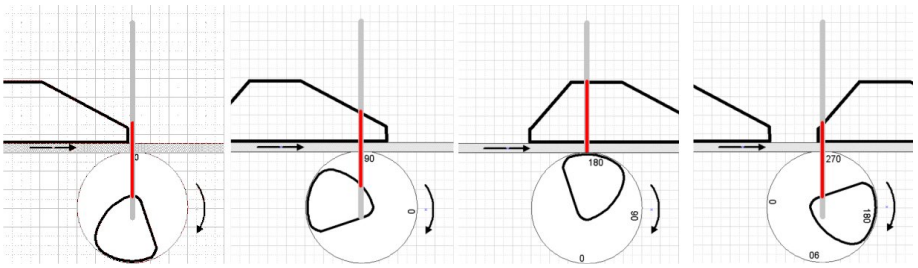


Abbildung 6.4: Königswelle (Masterachse) und Slaveachse sind gekoppelt. Die Positionen der Slaves hängen von dem Master ab, aus [Kuka 07b] Fritz S.3

6.2.1 Architekturprinzipien und Entwurfsalternativen

Die Softwarearchitektur der Simulation kann für die SPS nicht direkt übernommen werden, da sie keine objektorientierte Programmiersprache anbietet und der Architekturtreiber die laufzeit- und speichereffiziente Umsetzung ist. Nur benötigte Funktionen werden auf die SPS übernommen. Zudem bietet die Anbindung der Bahnplanung an die Roboterhardware mehrere Alternativen, die im nächsten Abschnitt 6.2.2 beleuchtet werden.

6.2.2 Hardwareanbindung

Die Bahnplanung liefert zu beliebigen Zeitpunkten die Gelenkstellungen und Geschwindigkeiten. Um einen geringen Bahnfehler und eine schnelle Reaktion zu ermöglichen, sollen Fahrbefehle in kurzen Taktintervallen auf die DSE gegeben werden können und die Übertragungsverzögerung zwischen der Beauftragung einer Bewegung und der Reaktion am Motor soll ebenfalls kurz sein. Folgende drei Alternativen ergeben sich nach den OpenPLC Beschreibungen [PLCOpen 01] und aktuellen Kuka Projekten [Kuka 07b]:

1. CAM-Tabelle

Die Fahrt einer Trajektorie auf Basis von Wertetabellen ist eine klassische Anwendung im CNC-Bereich. Roboter können exakt die Konturen von Werkstücken abfahren, die zuvor in einem CAD-Grafikprogramm am PC erstellt und in CAM-Tabellen exportiert wurden. Dies ist in KMC über die entsprechenden PLCOpen CAM-Bausteine implementiert. Die Terminologie des CAM-Mechanismus stammt aus der Geschichte der Robotik, als Achsenkopplungen der Königswelle mit Kurvenscheiben mechanisch realisiert wurden, siehe Abbildung 6.4.

Zum Füllen einer Tabelle kann eine Berechnungsprogramm z. B. die Simulation oder die Kuka Bahnplanung wie in Abbildung 6.5 verwendet werden: Für die Bahnplanung wird eine simulierte Kinematik erstellt, um ihr einen Kuka-Roboter vorzuspielen. Die simulierten Achsen sind jedoch mit dem physischen CNC-Roboter verbunden. Seine mathematischen Modelle werden in der simulierten Kinematik implementiert. Anschließend werden über die Kuka HMI im Teach-In-Verfahren Stützstellen der Bahnplanung definiert. Während der Fahrt werden die Gelenkstellungen über ein proprietäres Kuka-Tool ausgelesen und in tabellarischer Form in Dateien (pro Achse

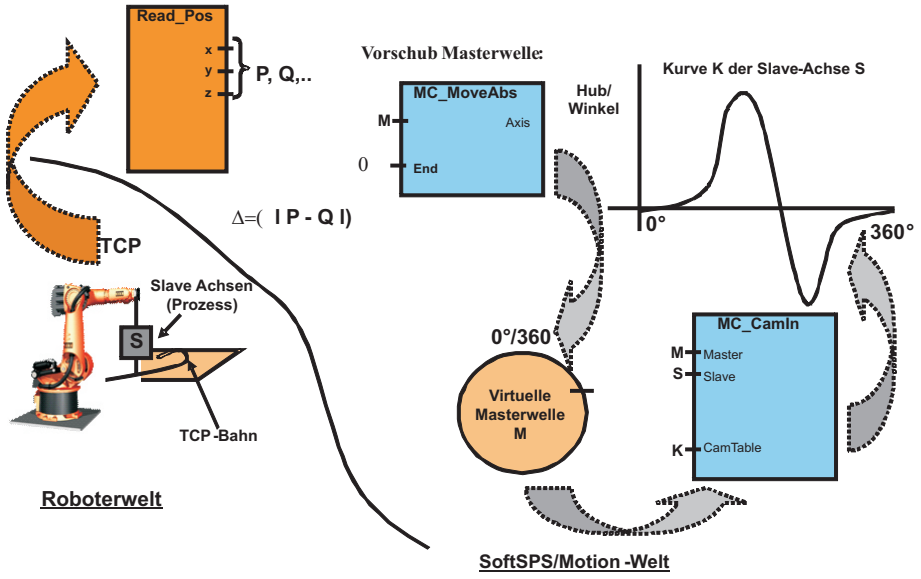


Abbildung 6.5: Bahnen werden bei der Fahrt in CAM-Tabellen gespeichert und sind über eine virtuelle Königswelle abfahrbar, aus [Kuka 07b] Strobel S. 32

und Bahn eine Datei) gespeichert. Alternativ kann die Debug-Funktion von MultiProg die Achsenbewegung in Dateien aufzeichnen.

In MultiProg wird der CamIn-Baustein zum Abfahren einer Tabelle verwendet. Dieser verwendet CamTableID's, die die Position des Verweises in der Tabellen-Index-Datei entspricht. Die Start- und Stoppgelenkstellung muss identisch sein und es muss dort die Geschwindigkeit Null herrschen. Nur in dieser home-Stellung kann ein Wechsel der CAM-Tabelle erfolgen, um eine andere Bahn zu fahren.

Um die Geschwindigkeit und die Position der Achsen während der Fahrt beeinflussen zu können, wird eine virtuelle Masterachse als Königswelle angeboten und jede Tabelle wird mit einer Slaveachse verknüpft. Der Königswelle wird der Wertebereich 0° bis 360° zugeordnet, wobei 0° dem ersten Achswert in der CAM-Tabelle entspricht und 360° dem letzten. Funktionsbausteine für KMC Einzelachsbewegungen, wie z. B. MoveAbsolute, bewegen die virtuelle Masterachse und somit über den CAM-Mechanismus auch die in den Tabellen abgelegten Achsen. Die Geschwindigkeit, mit der die Masterachse gedreht wird, muss anhand der gewünschten Fahrzeit berechnet werden.

Als Beispielprojekt für die CAM-Anwendung kann ein KMC Einleger für Druckpressen (2x 4-Achser) mit der Kuka-Steuerung gefahren werden. Da die Steuerung diese Kinematik nicht kennt, wird über die Kuka Benutzeroberfläche ein bekannter virtueller 4-Achser auf den jeweiligen physischen Roboter aufgeschaltet. Danach kann die Fahrt der CAM-Tabellen über KRL in ProConOS beauftragt werden und Königswellenpositionen mit Programmen verknüpft werden. Es wird erreicht, dass der Benutzer nur die Kuka HMI sieht und das SPS Programm verborgen im Hintergrund läuft. Auch Aus- und Eingaben werden zur Kuka HMI weitergeleitet. In

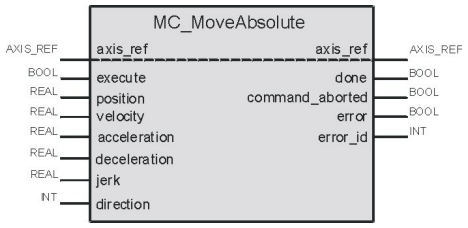


Abbildung 6.6: Blockschaltplan des PLCOpen MoveAbsolute Function Blocks, aus [Group 06] S. 65

der KHS wird zusätzlich zur Kuka-HMI eine eigene, einheitliche KHS-HMI mit größerem und auf die Maschinen zugeschnittenem Funktionsumfang verwendet, welche mit der SPS kommuniziert.

Die Vorteile des CAM-Verfahrens sind die Verwendung der Kuka-Bahnplanung im Teach-In-Verfahren und das für das Bedienpersonal bekannte Kuka HMI. Die in Dateien abgelegte Bahn ist im Nachhinein editierbar, um bestimmte Gelenkstellungen exakt vorzugeben. Der 5 MB reservierte Speicher für CAM-Tabellen reicht für ca. 1000 Bahnen. Da die Ausführung der Tabellen im CNC-Kern ohne vorgeschaltete Filter geschieht, betragen Übertragungsverzögerung und Takt jeweils 4 ms. Zwischen den Punkten der Tabelle wird automatisch linear oder polynomial interpoliert. Der CAM-Mechanismus ermöglicht einfach eine exakte Rückwärtsfahrt der Bahn, indem die virtuelle Königswelle rückwärts bewegt wird. Das ist bei der Kuka Bahnplanung aufgrund des Verschleifens nicht möglich.

Der Nachteil ist, dass die Bahnen nicht dynamisch änderbar sind, sondern fest in Dateien abgelegt sein müssen. Alle Bahnen müssen einen gemeinsamen Punkt (home) mit der Geschwindigkeit Null haben. Die Bahnen sind nur dort umschaltbar, somit ist die Wahl dieses Punktes kritisch. Bei Fehler muss home angefahren werden, bei manuellem Verfahren kann mit der letzten Bahnposition neu synchronisiert werden.

2. Einzelachs-Befehle

KMC Einzelachs-Befehle wie MoveAbsolute in Abbildung 6.6 sind in Software implementierte Achsregler. Sie werden beauftragt, die Achse eine gegebene Distanz unter Einhaltung der Randbedingungen wie maximale Geschwindigkeit, Beschleunigung, Ruck, etc. zu bewegen. Die Fertigstellung eines Jobs wird von ihnen quittiert.

MoveAbsolute-Bausteine sind zwar einfach und sicher zu verwenden, aber die Übertragungsverzögerung beträgt ca. 40 ms, da interne Filter zur Glättung der Bewegung ausführt werden. Der Baustein ist nur jeden dritten SPS-Zyklus (12 ms) neu beauftragbar, da die Flanke am Eingang *Execute* zweimal geändert werden muss und einen SPS Zyklus für die Ausführung reserviert werden sollte.

3. Achsenoffset im HLI

Das HLI (High Level Interface) ist physischer Speicher zur Kommunikation zwischen ProConOS und MC-Kern. Ein Befehl zur Inbetriebnahme ermöglicht die Kalibrierung von Achsen durch das Setzen eines Offsetwerts (Double-Integer) direkt bei der

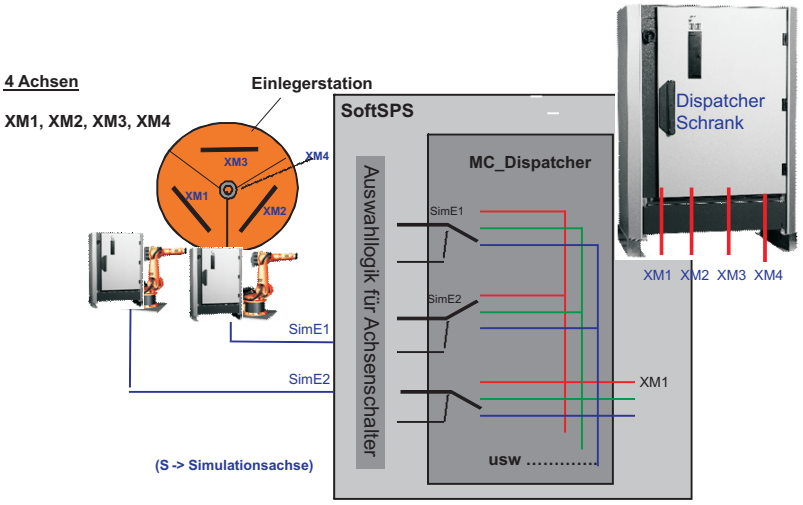


Abbildung 6.7: Kuka Dispatcher-Projekt mit drei Manipulatoren am Drehtisch und getrennter Steuerung mit dynamischer Achsenzuordnung, aus [Kuka 07b] Strobel S. 31

DSE. Der Offset wird relativ zur Achsposition des letzten Fahrbefehls gesetzt. Somit ist eine vorausgehende Fahrt mit z.B. MoveAbsolute nötig.

Die Abbildung 6.7 zeigt ein Kuka Beispielprojekt: Um einen Drehtisch mit drei orientierbaren Werkstücken stehen drei Roboter mit dynamisch zuordbaren Achsen auf dem Drehtisch. Die Zuordnung geschieht über einen vierten Schaltschrank mit Dispatcher. Somit müssen sich die Roboter nicht gegenseitig kennen, um Fahrbefehle und Positionen der Werkstück-Motoren auszutauschen, sondern bekommen dynamisch den Motor ihres nächsten Werkstücks zugewiesen. Die direkte HLI-DSE Ansteuerung über Hilscher-Bus ist notwendig um Roboter und Drehtisch synchron zu bewegen.

Die schnelle Reaktion der Achsen innerhalb vier ms wird durch dieses Verfahren ermöglicht. Werden sprunghafte Werte an das HLI übergeben, wird die Strombegrenzung der Antriebe aktiviert und sie erzeugt einen Achsfehler.

Fazit

Das CAM-Verfahren zeigt sich für den Einsatz mit KHS Maschinen als zu unflexibel, da aufgrund der dynamischen Palettenhöhe bei dem KHS Einleger viele Bahnen nötig sind. Es kann mit Einzelachsbefehlen wie MoveAbsolute kombiniert werden, so dass fixe Teile der Bahn über CAM und der variable Rest über Einzelachsbefehle gefahren wird. Da eine simulierte Kinematik für die Kuka-Bahnplanung erstellt werden kann, können die Ergebnisse dieses Verfahrens gut als Vergleich der Leistungsfähigkeit der selbst erstellten Bahnplanung dienen.
Die Ansteuerung der DSE über die HLI-Offsetfuntion ist die Variante mit der geringsten Reaktionszeit der Motoren. Die Übergabe von fehlerhaften oder sprungartigen Werten kann verhindert werden, indem ein Baustein für einen Plausibilitätstest der Steuerbefehle vorgeschaltet wird.

Die Einzelachs-Befehle wie MoveAbsolute sind in zeitkritischen Szenarien nicht gut anwendbar. Eine zu große Differenz der Reaktionszeit erschwert die Synchronisation zwischen einzelnen Robotern und Aktuatoren der Anlage. Die Reaktionszeit von ca. 100 ms ist jedoch nicht kritisch in KHS-Anwendungen, da alle Gelenke eines Roboters diese Verzögerung erfahren und das System insgeamt synchron bleibt. Die Einzelachs-Bausteine werden für die erste Erprobung der Bahnplanung an einem Roboter verwendet, da in der KHS Expertise mit diesem Mechanismus' besteht. Zur Erhöhung der Leistung kann später die HLI-Ansteuerung erprobt werden.

Programmierschnittstelle

Die Gestaltung der Programmierschnittstelle zielt auf die Frage ab, wie die Bahnplanung verwenden werden kann. Ein SPS-Programmierer sollte möglichst keine Arrays direkt manipulieren oder im Quellcode programmieren müssen, da ohne Kenntnisse der internen Vorgänge unvorhersehbare Auswirkungen entstehen können. Stattdessen sollen bekannte Mechanismen der Einzelachssteuerung angeboten werden, z.B. ähnlich zum CAM-Verfahren wird der Bahn eine virtuelle Masterachse (Wertebereich 0,0 - 100,0 ≅ entspricht ursprünglicher Fahrtzeit von Startzeit bis Zielzeit) zugeordnet. Sie soll mit MoveAbsolute-ähnlichen Funktionsblöcken manipuliert werden, z.B. Stopp falls Synchronisationsereignisse noch nicht eingetreten sind, Rückwärtsfahrt bei einer Kollision, Drosseln und Steigern der Geschwindigkeit zur Synchronisation mit anderen Aktuatoren der Anlage.

Das Vorgeben von Stützpunkten, Bahneigenschaften und Befehle für die Abfahrt wird mit einem Funktionsbaustein wie in den nicht-funktionalen Anforderungen beschrieben durchgeführt. Für die erste prototypische Implementierung soll die Programmierschnittstelle möglichst einfach gestaltet sein und kein Definieren von Bahnstützpunkten und Geschwindigkeiten zulassen, sondern diese Größen sollen automatisch berechnet werden.

6.2.3 Dekomposition des Systems

Nach der Einführung der Architekturprinzipien wird in diesem Abschnitt die Dekomposition vorgenommen. Sie zeigt die funktionale Zerlegung in abgeschlossene Elemente und die statische Struktur des Systems. Das Entwurfsergebnis wird als Graph der zu realisierenden SW-Elemente sowie ihrer Beziehungen untereinander dokumentiert.

Abbildung 6.8 zeigt die Systemdekomposition mit den zu entwickelnden Elementen (Software-Einheiten) und bereits vorhandenen Elementen (externe Einheiten).

Das Gesamtsystem besteht aus der **Robotersteuerung**, dem **Roboter** und der **Simulation**. Die Simulation wurde in Abschnitt 5.4 erläutert und wird hier nicht weiter behandelt. Die Robotersteuerung ist in **Anwenderprogramm**, **Einzelachssteuerung** und dem **Bahnplanungssystem** gegliedert. Das Bahnplanungssystem besteht wiederum aus mehreren Software-Komponenten, in denen sich die Funktionalität der Robotermodellierung, Bahnplanung und Geschwindigkeitsberechnung wiederfinden. Diese Komponenten sind in mehreren, atomaren Module zerteilt.

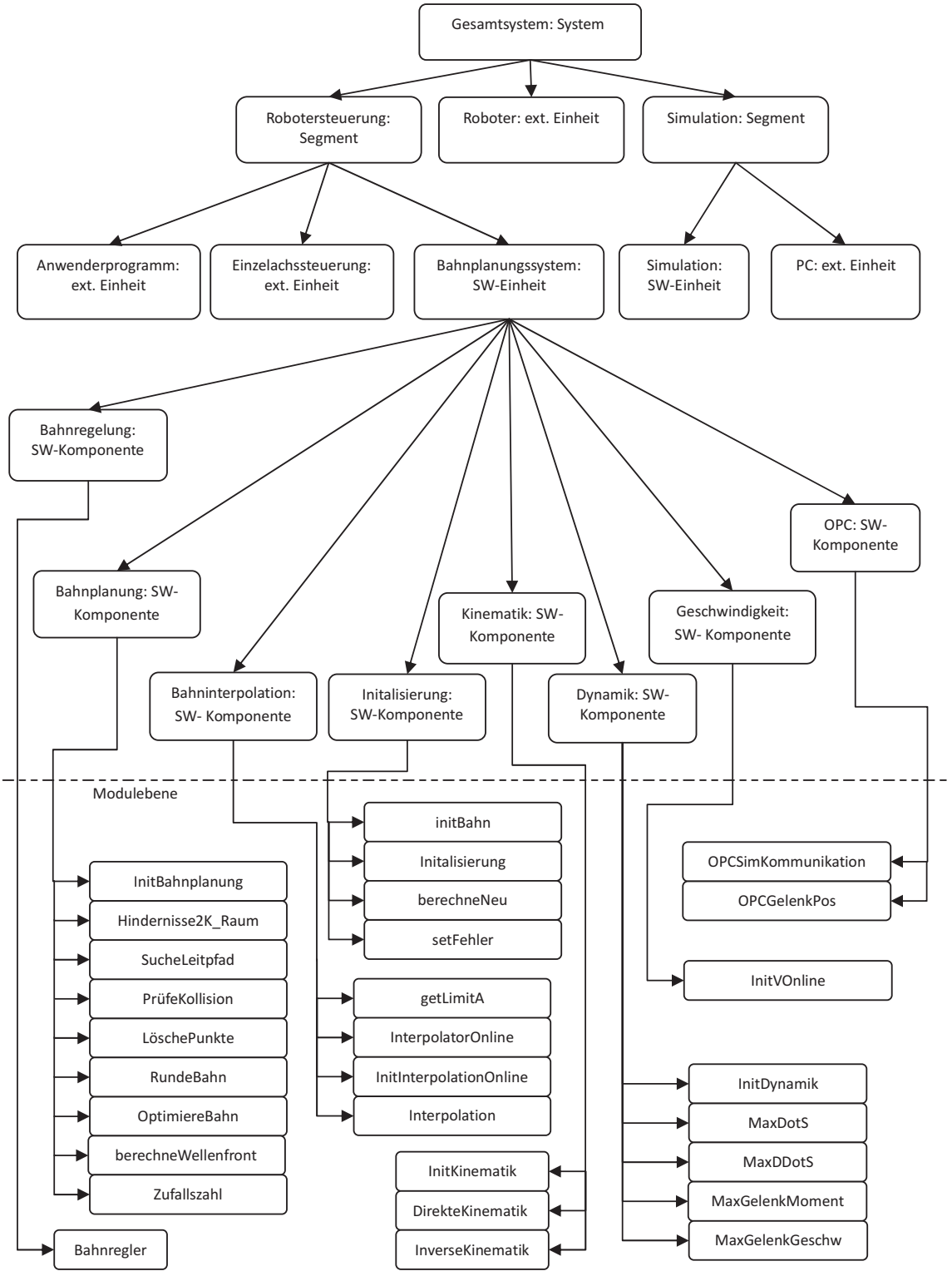


Abbildung 6.8: System-Dekomposition

6.2.4 Querschnittliche Systemeigenschaften

In einem System lassen sich systemelementspezifische und systemübergreifende Eigenschaften unterscheiden. Zu typischen systemübergreifenden Eigenschaften zählen bei SW-Systemen beispielsweise Transaktionsanforderungen, Persistierung von Daten, Anforderungen an Logging und Tracing und die Ausnahme- und Fehlerbehandlung. Für HW-Systeme können dies beispielsweise einheitliche Steckerbelegungen oder systemübergreifende Sicherheitsanforderungen sein.

Die relevanten querschnittlichen Systemeigenschaften werden durch die nichtfunktionalen Anforderungen abgedeckt.

6.2.5 Schnittstellenübersicht

In der Schnittstellenübersicht werden die Schnittstellen des Systems und seiner Systemelemente im Überblick dargestellt. Zur Beschreibung der Schnittstellenübersicht wird jeweils nur die Kommunikation auf einer Ebene betrachtet.

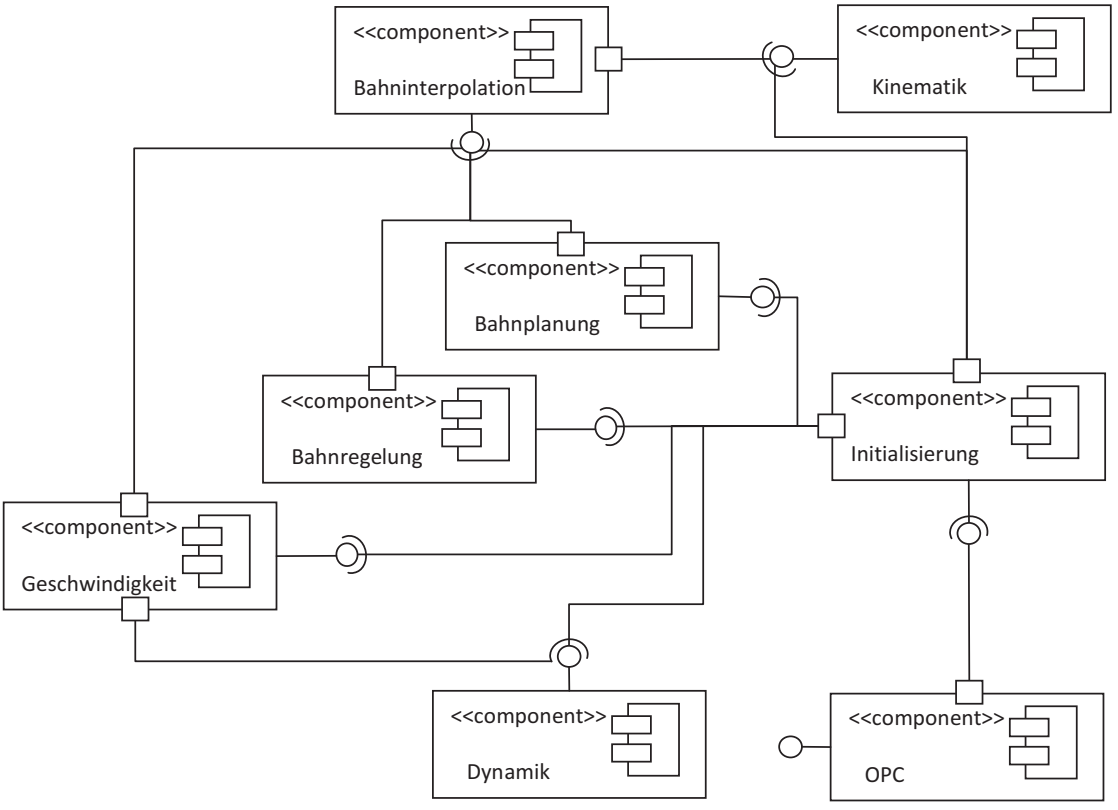


Abbildung 6.9: Komponentendiagramm der SW-Einheiten mit Schnittstellen

Abbildung 6.9 zeigt die Schnittstellen der Softwareeinheiten. Weil Funktionsbausteine der SPS im Gegensatz zu Objekten aus objektorientierten Sprachen wenig Daten selbst halten sondern aus Effizienzgründen vorrangig über globale Variablen kommunizieren, ist das Geflecht aus Schnittstellen und ihrer Verwendung gering ausgeprägt.

Die Komponente **Initialisierung** verwendet die meisten anderen Komponenten sowohl zum erstmaligen Setzten von Parametern als auch zum Anstoßen der Berechnung einer Trajektorie. Die Komponente **Bahninterpolation** hingegen wird von den meisten anderen Komponenten verwendet. Ihre Funktion spielt eine zentrale Rolle. Im Folgenden werden die Schnittstellen detailliert beschrieben.

- Die Schnittstelle von **Bahninterpolation** dient **Initialisierung** zum Berechnen der Interpolationskoeffizienten (2. Ableitung der Stützstellen) und zum Überführen der Bahnstützpunkte in die numerisch interpolierte Bahn. **Bahnplanung** benötigt diese Schnittstelle, um neu interpolierte Bahnen auf Kollisionen zu prüfen. **Bahnregelung** greift über die Schnittstelle auf die interpolierten Gelenkstellungen in der Bahn zu, um diese an die Antriebe zu übergeben. **Geschwindigkeit** greift auf die Schnittstelle zu, um die feininterpolierte Grenzgeschwindigkeit und -beschleunigung während der Berechnung des Geschwindigkeitsprofils abzurufen.
- Die Schnittstelle von **Kinematik** dient **Initialisierung** zum Setzen der roboterspezifischen kinematischen Parameter. **Bahninterpolation** verwendet diese Schnittstelle zum Überprüfen der Endanschläge der Gelenke.
- Die Schnittstelle von **Dynamik** dient **Initialisierung** zum Setzten der roboterspezifischen dynamischen Parameter. **Geschwindigkeit** greift auf die Schnittstelle zu, um Gelenkdrehmomente zur Begrenzung der Bahngeschwindigkeit zu berechnen.
- Die Schnittstelle von **Bahnplanung** dient **Initialisierung** zum erstmaligen Erzeugen der Hindernisbeschreibung und zum Berechnen kollisionsfreier Bahnen.
- Die Schnittstelle von **Bahnregelung** dient **Initialisierung** zum erstmaligen Setzen der Sollwerte und Maximalwerte.
- Die Schnittstelle von **Geschwindigkeit** dient **Initialisierung** zur Berechnung von zeitoptimalen Geschwindigkeitsprofilen.
- **Initalisierung** und **OPC** bieten nur für das Anwenderprogramm und die Simulation Dienste an. **OPC** kann einen Auftrag zur Neuberechnung von der Simulation an **Initalisierung** weiterleiten, Bahnen auslesen und setzen.

6.2.6 Übergreifender Datenkatalog

Im Datenkatalog werden die an den Schnittstellen der SW-Einheiten ausgetauschten Datenstrukturen mit Attributen, Datentypen und Wertebereichen beschrieben. Jede Programmiersprache und Plattform bietet hier eigene Lösungen, die bei der Definition zu berücksichtigen sind.

In Abbildung 6.10 ist eine Übersicht des Datenkatalogs mit globalen Variablen gezeigt. Die Datentypen besitzen folgende Eigenschaften:

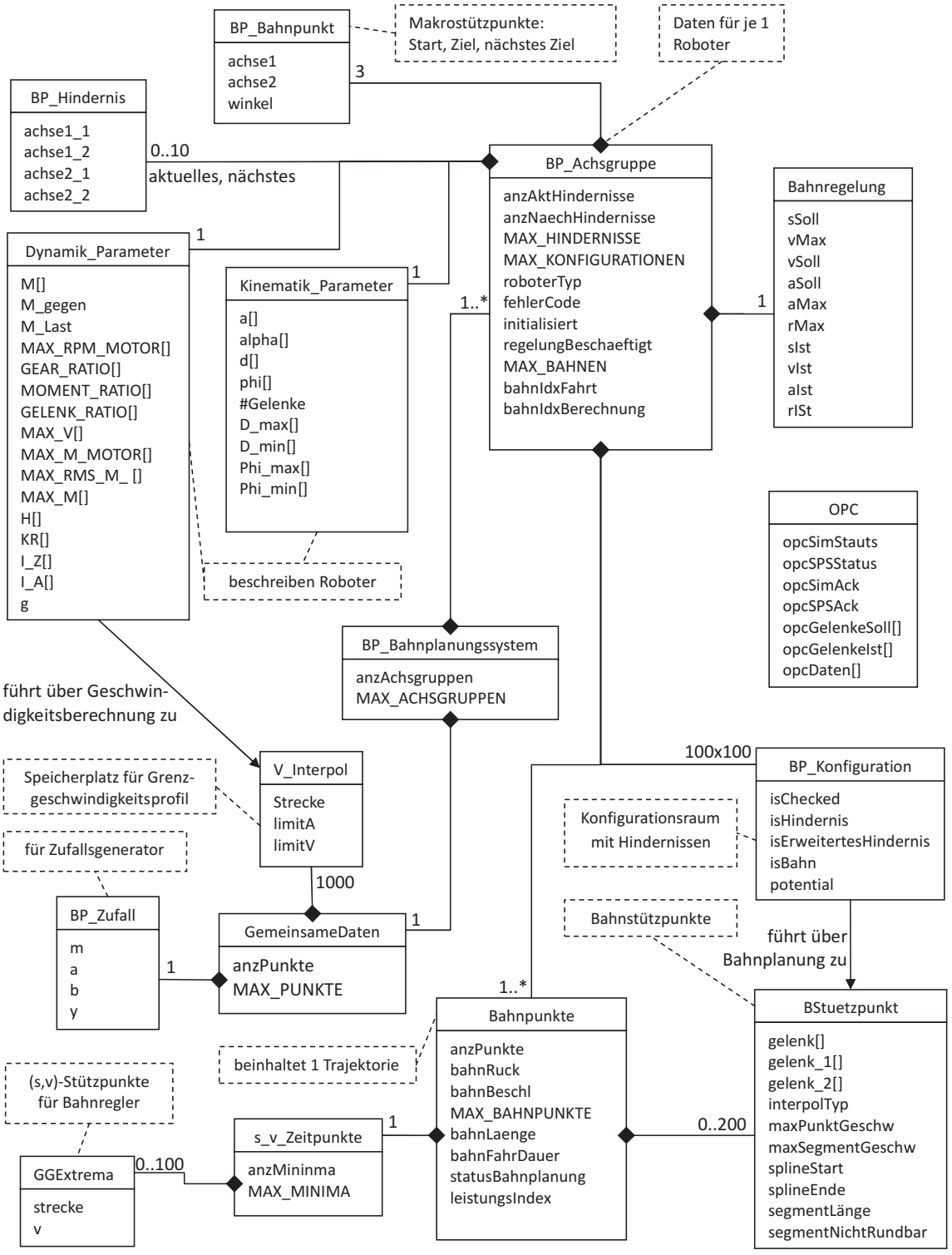


Abbildung 6.10: Globaler Datenkatalog

- **BP Bahnplanungssystem** kapselt sämtliche Daten der Bahnplanung für alle Achsgruppen
- **GemeinsameDaten** beinhaltet den Zufallsgenerator und reserviert Speicher zum numerischen Ablegen des Grenzgeschwindigkeitsprofils, welcher von allen Achsgruppen gemeinsam genutzt wird.
- **BP Zufall** enthält die Daten eines Pseudo-Zufallsgenerators für die Bahnoptimierung. Die SPS-Sprachen beinhalten keinen Mechanismus zur Erzeugung von Zufallszahlen.
- **V Interpol** speichert das Grenzgeschwindigkeitsprofil mit vielen Punkten.
- **BP Achsgruppe** verwaltet Bahnen, Hindernisse und Zielpositionen eines Roboters.
- **BP Hindernis** beinhaltet Informationen zu Hindernissen. In dem durch die Grenzen berandetem Bereich befindet sich ein Hindernis.
- **BP Bahnpunkt** stellt einen Stützpunkt für z. B. Start- oder Zielposition für die Bahnplanung dar.
- **Kinematik Parameter** beinhaltet eine Beschreibung der Roboterkinematik nach DH-Konventionen. Die Anzahl der Gelenke spiegelt sich in den Multiplizitäten der Parameter wieder. Die Grenzen der Gelenkstellungen wird bei der Interpolation verwendet.
- **Dynamik Parameter** kapselt die dynamischen Eigenschaften des Roboters. Die Maschinen- und Antriebsdaten fließen über die Dynamikgleichung in die Geschwindigkeitsberechnung mit ein.
- **Bahnregelung** kapselt den Bewegungszustand der Maschine und Regelparameter, Maximal-, Ist- und Sollwerte der Regelung.
- **BP Konfiguration** beinhaltet eine Konfiguration mit der Beschreibung des Zustandes und des Potentials. Die Hindernisinformation wird beim Diskretisieren des Konfigurationsraums aus **BP Hindernis** gewonnen und das Potential entsprechend der Lage des Ziels aus **BP Bahnpunkt** berechnet.
- **BInterpol** speichert einen interpolierten Bahnpunkt mit den Werten der Gelenkstellungen. Zur Berechnung des Leistungsindex' über die Bahnkrümmung und der zeitoptimalem Geschwindigkeit sind die ersten beiden Bahnableitungen abgelegt. Zusätzlich beinhaltet er Informationen das Grenzgeschwindigkeitsprofils.
- **Bahnpunkte** beinhaltet die Bahnstützpunkte und Randbedingungen einer Bahn. Diese Daten ergeben sich aus der Benutzereingabe über **OPC**, den Vorgaben des Anwenderprogramms oder aus der automatischen Bahnplanung.
- **BStuetzpunkt** stellt einen einzelnen Bahnstützpunkt dar. Er beinhaltet für die Interpolation nötige Zwischendaten und Informationen zur Interpolationsart.

- **s v Zeitpunkte** enthält die s, v-Stützpunkte, die aus der Analyse des Grenzgeschwindigkeitsprofils **GemeinsameDaten** resultieren.
- **GGExtrema** kapselt einen s, v-Stützpunkt. Diese Punkte werden bei der zeitoptimalen Fahrt für v_{max} von **Bahnregelung** verwendet.
- **OPC** dient zur Zwischenspeicherung der OPC-Daten und zur Kommunikation mit der Simulation.

6.2.7 Systemverhalten

Nach der funktionalen und strukturellen Beschreibung des Softwaresystems wird nun sein Verhalten dargestellt.

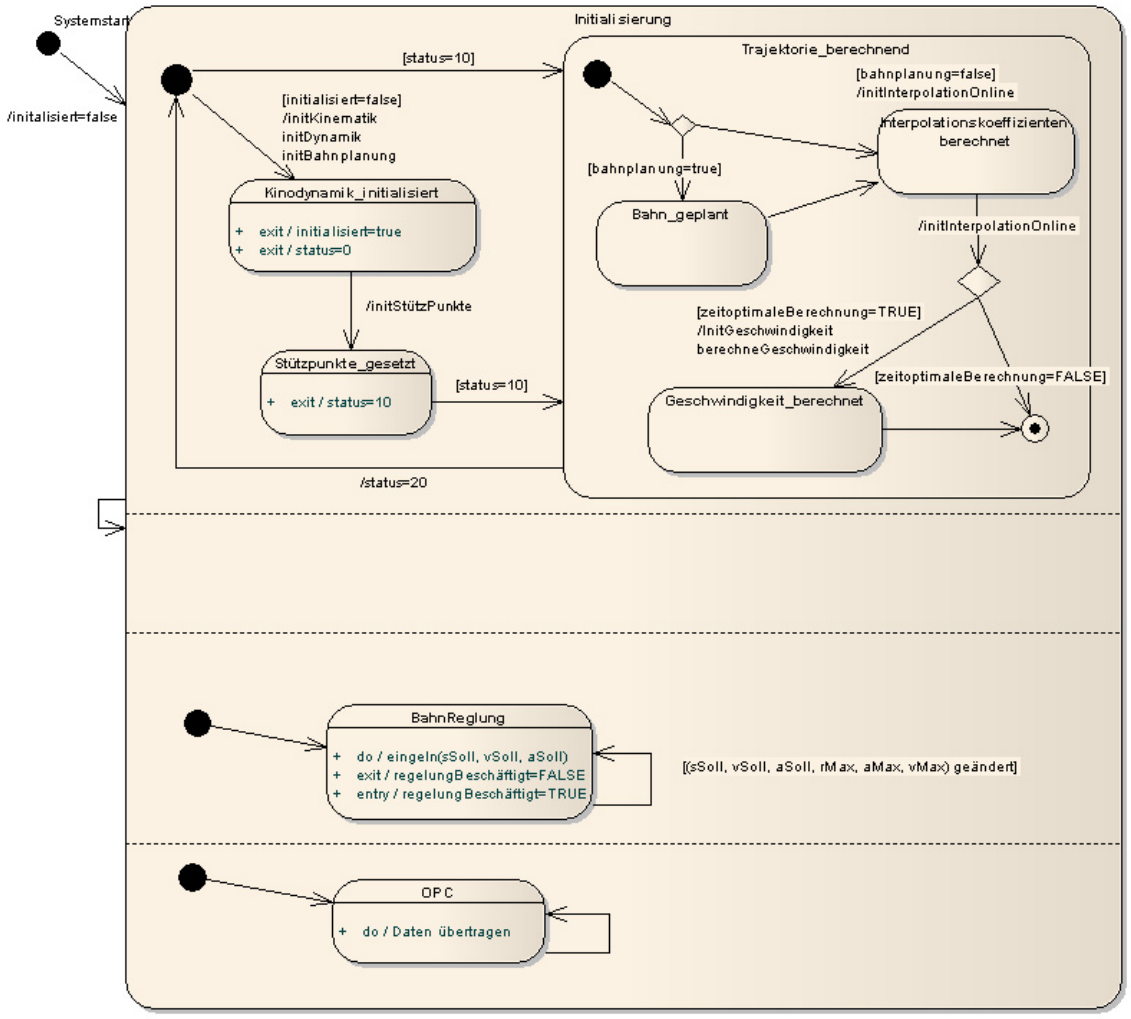


Abbildung 6.11: Verhalten des Bahnplanungssystems

Das Zustandsmodell aus Abbildung 6.11 zeigt das Verhalten der Bahnplanung. Die **Bahnregelung**, **OPC** und die **Initialisierung** laufen nebenläufig, wobei sich **Bahnregelung** mit **Initialisierung** synchronisieren muss. Diese drei Systemteile sind als aktive Tasks zu realisieren. Der Synchronisierungsmechanismus wird mit Flags und einer Status-Variablen implementiert. Wenn das Anwenderprogramm aus Bahnstützpunkten eine Trajektorie berechnen lassen will, muss es die Stützpunkte setzen oder durch die Bahnplanung berechnen lassen und anschließend mit der Status-Variablen die Neuberechnung der Interpolationskoeffizienten und der s, v-Punkte bei der Initialisierung beauftragen. Erst nach dieser Berechnung kann der Bahnregler den aktuellen Bahnparameter an den Interpolator übergeben und den Roboter bewegen.

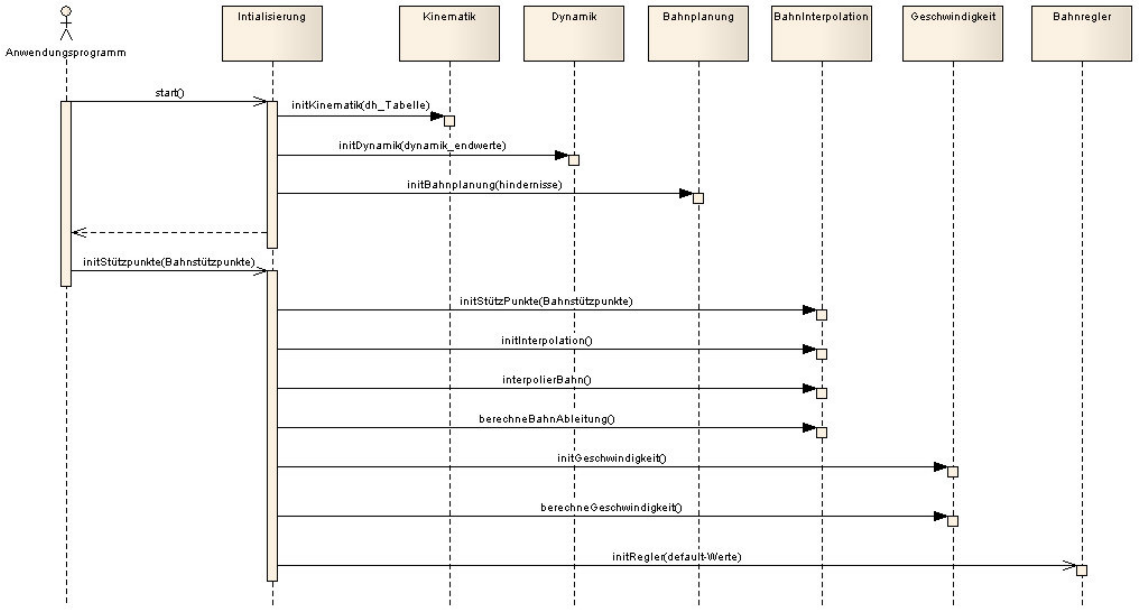


Abbildung 6.12: Exemplarischer Ablauf zur Erstellung einer neuen Trajektorie

6.2.8 Designabsicherung

Die Designabsicherung dient zur Validierung des Entwurfs. Die vorgestellten Konzepte werden dabei auf Plausibilität, Vollständigkeit und Eindeutigkeit überprüft. Als Überprüfungsverfahren bieten sich das Durchspielen von Szenarien und die Erstellung eines Prototypen an.

Szenario-basierte Architekturevaluierung

Abbildung 6.12 zeigt ein Sequenzdiagramm, das den Start des Systems mit dem Berechnen einer Bahn zeigt. Das Anwendungsprogramm stößt in der Initialisierung die Roboterparametrierung und die Bahnberechnung an.

Abbildung 6.13 stellt den Vorgang der Bahnplanung dar. Schlägt eine Kollisionsprüfung fehl, wird die Bahnplanung mit einem Fehler beendet, da in den Optimierungsfunktionen **rundeBahn** und **löschePunkte** selbst eine Kollsionsprüfung stattfinden. Der dargestellte Ablauf wird so lange wiederholt, bis sich der Leistungsindex der Bahn nicht mehr verbessert.

Prototypische Entwicklung von Systemteilen

Zur Absicherung des Entwurfs und der Hardwareanbindung des Bahnplanungssystems wird ein Prototyp erstellt. Er besteht aus dem Bahnplanungssystem, einem Vorlageprojekt der Einzelachssteuerung und einem Versuchsaufbau der Hardware. Da die SPS-Simulation nicht funktioniert, sobald Bibliotheken für die Einzelachssteuerung (MC-Kern) in einem Software-Projekt verwendet werden, muss die Steuerung auf einen Kuka-Rechner getestet werden. Abbildung 6.14 zeigt den Kuka-Schaltschrank mit provisorisch angebrachten Motoren. Der Versuchsaufbau zeigte die erwarteten Ergebnisse und bot erste praktische Laufzeittests.

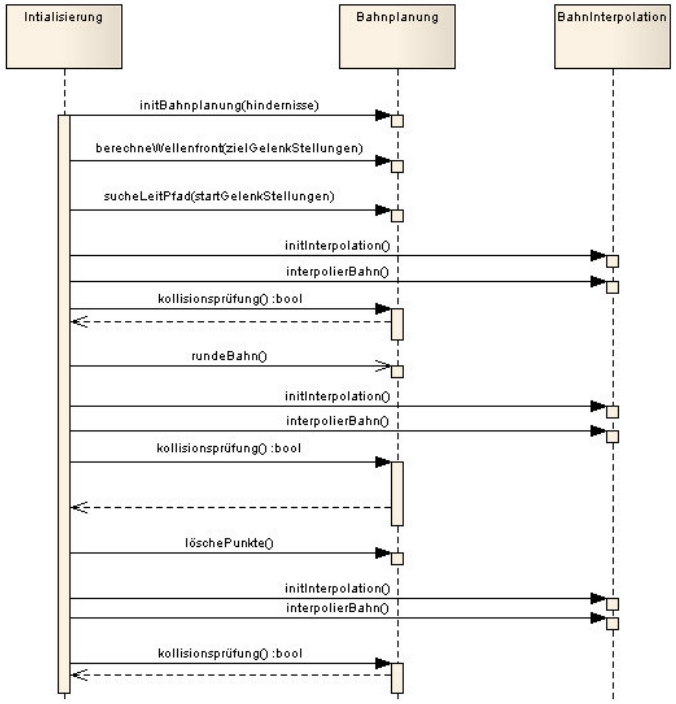


Abbildung 6.13: Exemplarischer Ablauf einer Iteration der Bahnplanung

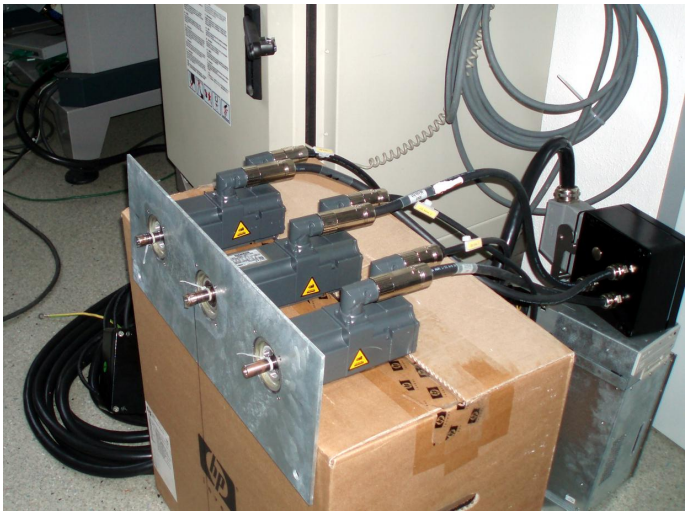


Abbildung 6.14: Versuchsaufbau mit drei Motoren. Die SPS-Simulation kann den NC-Kern zur Ansteuerung der Antriebe über MoveAbsolute nicht simulieren, hier muss direkt auf dem Kuka-Rechner getestet werden

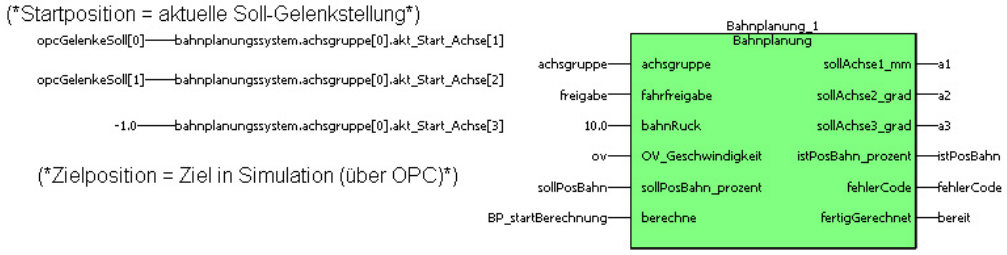


Abbildung 6.15: Schnittstelle zur Bahnplanung über Funktionsbaustein

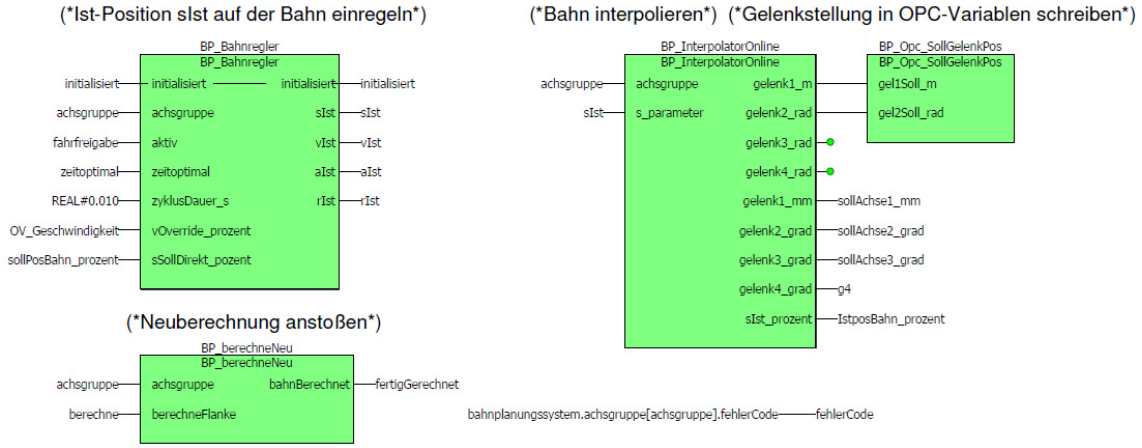


Abbildung 6.16: Aufbau des Bahnplanung-Funktionsbausteins aus Abbildung 6.15

Nachteilig zeigte sich die große Reaktionsverzögerung der Achsen mit ca. 200 ms. Sie ließ sich auf Drehzahlfilter des MC-Kerns zurückführen. Ohne Filter zeigten die Motoren eine schnellere (20 ms) aber sehr ruckartige Bewegung. Die direkte HLI-Ansteuerung führte auch zu ruckhaften Bewegungen, so dass im Folgenden vorerst die Einzelachssteuerung mit MoveAbsolute-Bausteinen durchgeführt wird.

6.3 Implementierung

Dieser Abschnitt zeigt die Konzepte von Algorithmen und die Integration der Bahnplanung in vorhandene Steuerungsprogramme auf.

Das Bahnplanungssystem wird als Bibliothek in vorhanden Projekte eingebunden. Abbildung 6.15 zeigt die Schnittstelle des Bahnplanungssystems, die sich in einem Taks befinden muss und zyklisch ausgeführt wird. Dieser Baustein ist wie in Abbildung 6.16 dargestellt aufgebaut. Die Ausgabe des Bahnreglers s_{ist} wird direkt für den Interpolator verwendet. Der Regler greift dabei über die globale Datenstruktur BP Bahnplanungssystem auf die (s, v)-Stützpunkte zu, der Interpolator auf die durch die Initialisierung berechneten Splinekoeffizienten. Bei einer Neuberechnung wird der Task **Initialisierung** zur Ermittlung der nötigen Koeffizienten und der Geschwindigkeiten angestoßen.

Globale Variablen, Tasks und die Ausgabe der Gelenkwerte müssen korrekt mit dem Projekt instanziiert und verbunden werden.

Folgendes Codefragment zeigt ein einfaches Anwendungsprogramm, das den Roboter über den Bahnregler eine berechnete Trajektorie endlos abfahren lässt:

Listing 6.1: Anwenderprogramm der Bahnplanung

```
if regelungBeschaeftigt or not aktiv then
  return;
end_if;
case zustand of
  0: bahnregelung.sSoll := bahn.bahnLaenge;
    zustand := 1;
  1: bahnregelung.sSoll := 0.0;
    zustand := 0;
end_case;
```

Wenn wie in Code (6.1) die Regelung den Leerlauf signalisiert, hat die Regelung ihr Ziel erreicht. Dann kann ein neuer Fahrauftrag abgesetzt werden.

7. Testergebnisse mit KHS-Robotern

Nachdem die Rechenverfahren erstellt und ihre Funktionalität in der Simulation verifiziert wurde, wird in diesem Kapitel die Praxistauglichkeit des Bahnplanungssystems untersucht. Es sollen in Versuchen Soll-Trajektorien mit Ist-Trajektorien verglichen werden, um Fehlerquellen zu finden und Verbesserungen an der Steuerung vorzunehmen. Die Versuche lassen sich in drei Kategorien einteilen:

1. Erfassung von unbekannten Roboterparametern,
2. Verhalten des Roboters und der Steuerung wie z. B. Bahntreue
3. Einsatz des Bahnplanungssystems für den Produktionsbetrieb.

Während Experimente des ersten Typs unter Laborbedingungen durchgeführt werden, sind die restlichen Experimente, insbesondere die Anwendung der Bahnplanung in Kombination mit einer Handhabungsaufgabe, Feldversuche.

Die zeitoptimale Geschwindigkeitsberechung liefert nur korrekte Ergebnisse, wenn alle Roboterparameter genau bekannt sind. **Abschnitt 7.1** geht auf Versuche zur Bestimmung von unbekannten Parametern an einem RS3-Roboter ein.

Abschnitt 7.2 untersucht den Bahnfehler und demonstriert die online-Bahnplanung.

7.1 Empirische Erfassung der Fehler durch Modell-Idealisierungen

Gegenstand des Versuchs ist ein KHS RS3-Roboter mit Kuka KRC2-Steuerung. Die Bahnplanung wird in ein Vorlageprojekt mit mit KHS-eigenen Funktionsbausteinen zur Einzelachssteuerung integriert. Es sollen unbekannte Roboterparameter und Ungenauigkeiten der Modellierung erfasst werden.

Versuchsbeschreibung

Das Trägheitsmoment des Armsegments und die Reibungskoeffizienten sind bei dem RS3-Roboter nicht bekannt. Das in Abschnitt 3.3 eingeführte Reibungsmodell beinhaltet für jedes Gelenk einen konstanten und geschwindigkeitsproportionalen Korrekturfaktor, der in das Gelenkmoment einfließt.

Die Differenzen zwischen Ist-Drehmomente und berechneten Soll-Drehmomente während der Fahrt über eine berechnete Bahn lassen auf die unbekannten Größen schließen. Die Ist-Drehmomente werden mit dem Motorstrom und einem kt-Faktor berechnet: $m = kt \cdot I_{max} \cdot k_{I,rel}$. Der kt-Faktor und der maximale Beschleunigungsstrom I_{max} sind in den Motordatenblättern zu finden, der relative Motorstrom $k_{I,rel}$ kann über das HLI oder KMC-Funktionsbausteinen ausgelesen werden.

Durchführung

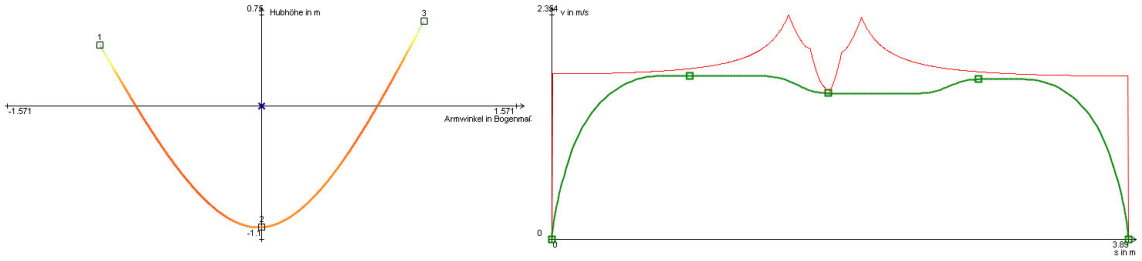


Abbildung 7.1: Einfache Bahn und berechnete Geschwindigkeit für RS3-Roboter zur Überprüfung der Dynamikberechnung (aus Simulation)

Zunächst wird die einfache Bahn aus Abbildung 7.1 mit dem RS3 abgefahren. Abbildung 7.2 zeigen die Aufzeichnungen in MultiProg.

Auswertung

Abbildung 7.3 zeigt die aufbereiteten Messdaten des Versuchs. Die Vernachlässigung der Reibung und die nicht exakte Einstellung der Maschinendaten der Kuka-Steuerung führen zu Abweichungen in der Drehmomentberechnung und das Dynamikmodell berechnet somit falsche Bahngeschwindigkeiten.

Aus den Differenzen der Drehmomentkurven werden die Korrekturfaktoren der Gelenke abgeleitet. Zudem werden in den Maschinendaten die Motorregler härter eingestellt, so dass Motorbeschleunigung und -strom kurzzeitig die vorgegebene Grenzen überschreiten können.

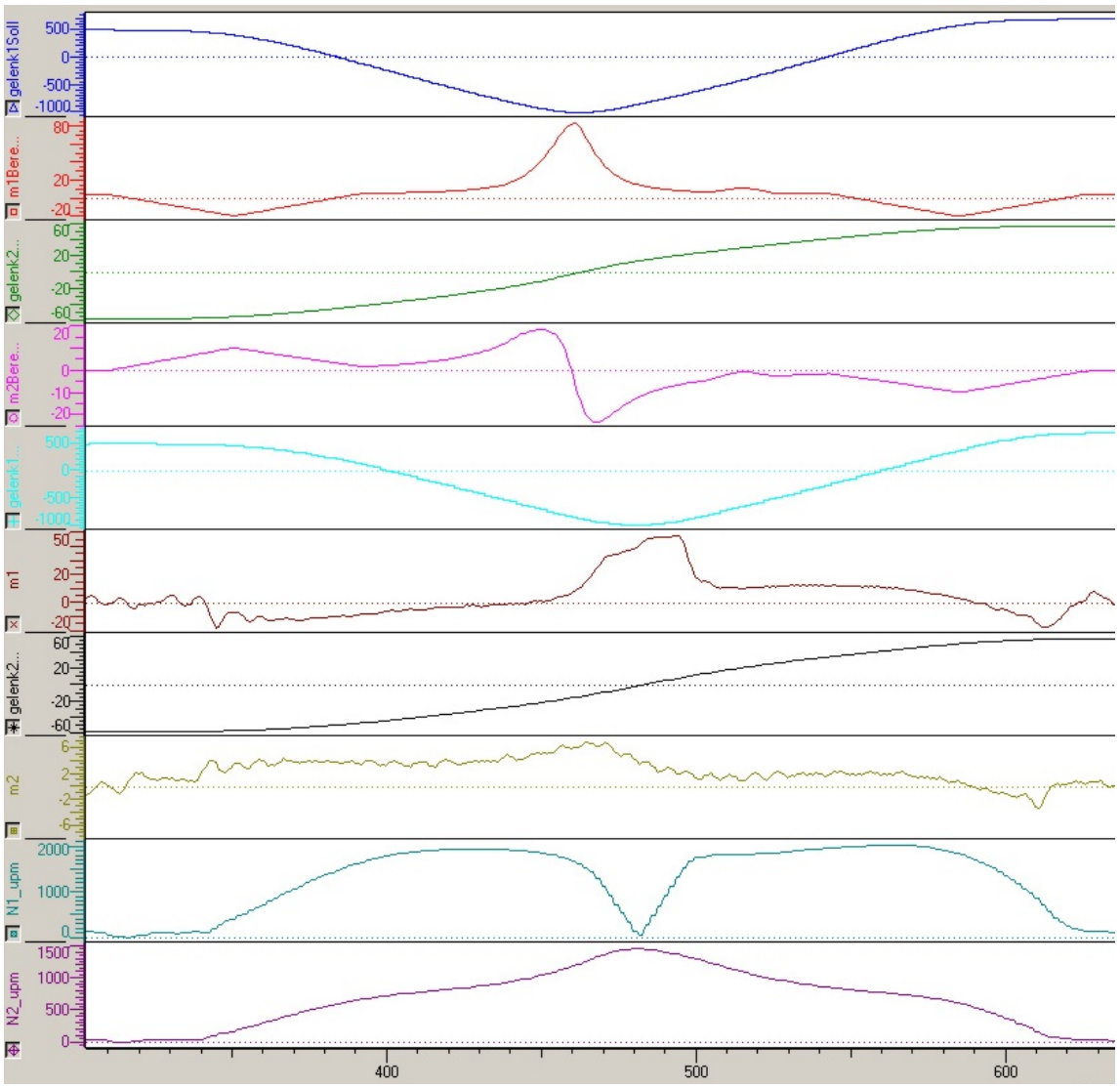


Abbildung 7.2: Oszilloskop-Aufzeichnung der Soll- und Istwerte in der IDE MultiProg während der Fahrt über die Bahn: Soll-Stellung Gelenk 1, berechnetes Motordrehmoment 1, Soll-Stellung Gelenk 2, berechnetes Motordrehmoment 2, Ist-Stellung Gelenk 1, Ist-Drehmoment Motor 1, Ist-Stellung Gelenk 2, Ist-Drehmoment Motor 2, Ist-Drehzahl Motor 1, Ist-Drehzahl Motor 2. Soll- und Ist-Drehmomentkurven weichen deutlich voneinander ab .

Abbildung 7.4 stellt die Messung der Drehmomente nach der Anpassung dar. Die Korrelation der berechneten Drehmomente mit den tatsächlichen ist deutlich höher. Die Geschwindigkeitsberechung wird somit realistischere Trajektorien erzeugen.

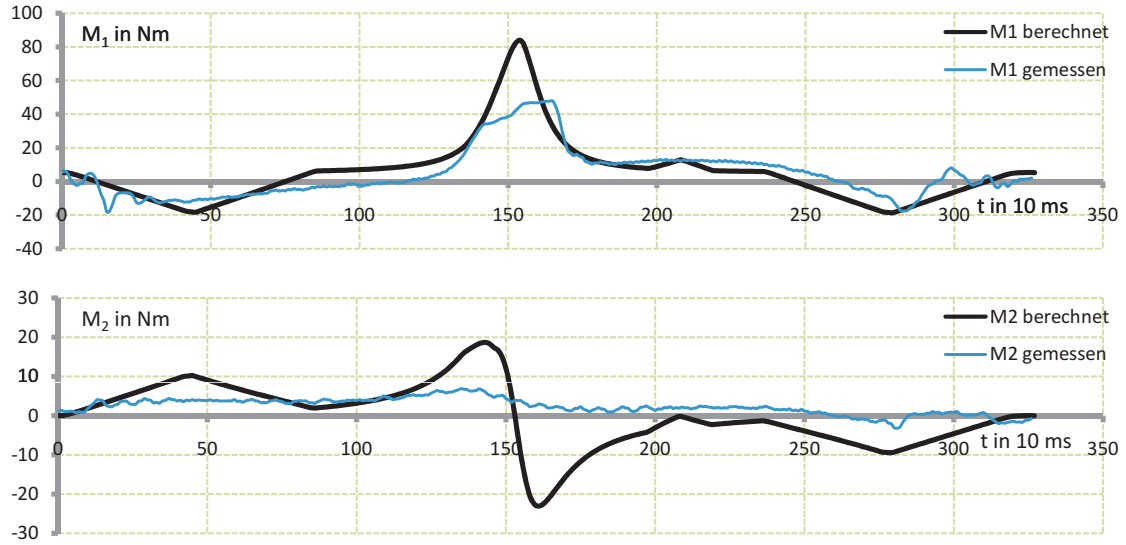


Abbildung 7.3: Berechnete und gemessene Drehmomentkurve des Hub- und Armmotors mit Berücksichtigung der zeitlichen Verzögerung der Messwerte .

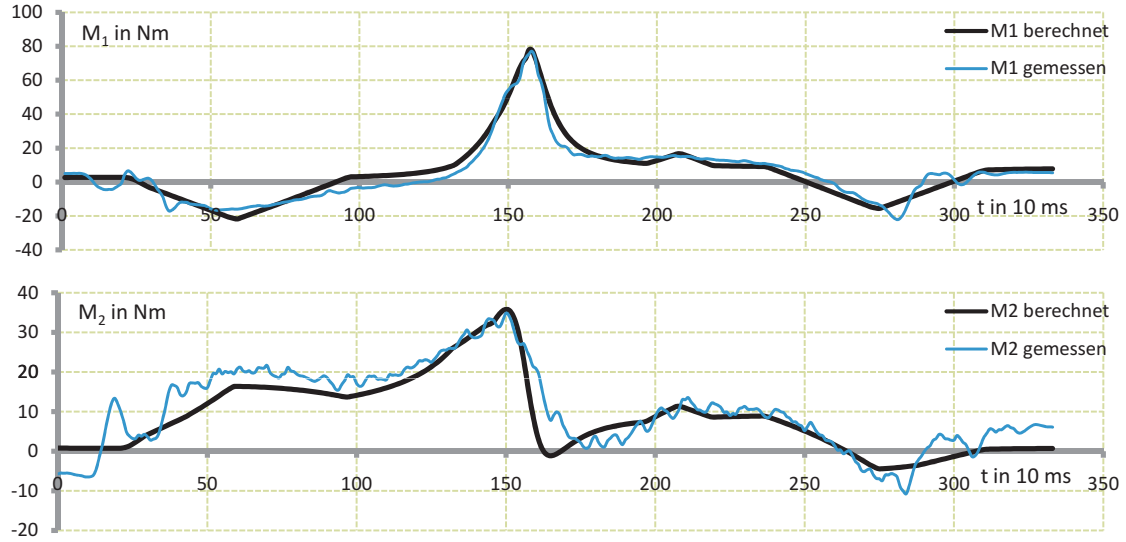


Abbildung 7.4: Berechnete und gemessene Drehmomentkurve des Hub- und Armmotors des RS3-Roboters nach Anpassung der Reibungskoeffizienten des Dynamikmodells, der Reglerparameter der Kuka-Steuerung und mit Berücksichtigung der zeitlichen Verzögerung der Messwerte

7.2 Verhalten der Bahnplanung

In den folgenden Versuchen kommt wieder der RS3-Roboter zum Einsatz. Es wird der Bahnfehler untersucht und die Fahrt über eine online geplante Bahn demonstriert.

Versuchsbeschreibung

Mit der Simulation wird ein Bahn erstellt und ihre Stützpunkte über OPC in die Robotersteuerung übertragen. Anschließend wird aus den übertragenen Informationen in der SPS die Trajektorie erstellt und ihre geometrische Kontur zurück in die Simulation übertragen.

Durchführung

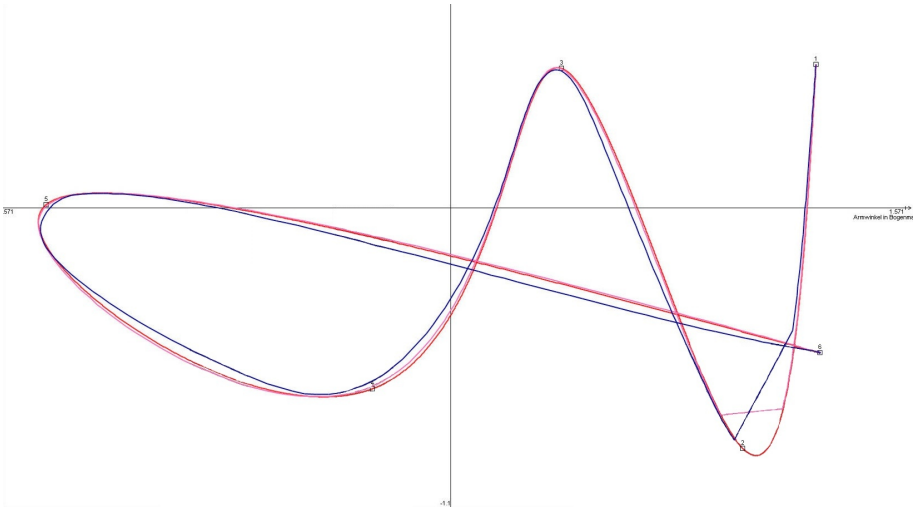


Abbildung 7.5: In Simulation berechnete Bahn 2 (dunkelrot), in SPS berechnete Bahn (hellrot) und gefahrene Bahn (blau). Die OPC-Übertragung von SPS zur Simulation zeigt einen Ausfall .

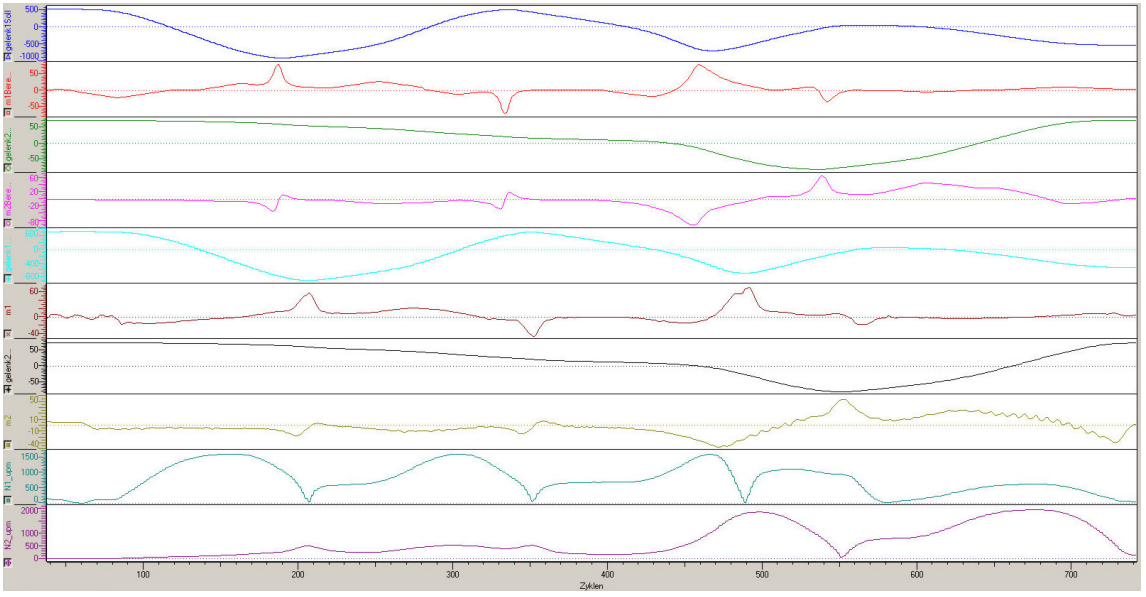


Abbildung 7.6: Oszilloskop-Aufzeichnung der IDE

Während der Fahrt überträgt die Bahnplanung in der SPS die aktuellen Gelenkstellungen an die Simulation. Die Gelenkstellungen werden über die KHS-Einzelachsbausteine ausgelesen und werden ursprünglich über das HLI von dem MC-Kern an die SPS übertragen.

Abbildung 7.5 stellt die Bahn der Simulation, die Bahn der SPS und die gemessene Bahn dar. Die SPS-Bahnen zeigen einen Sprung in der Nähe von Stützpunkt zwei. Dieser kommt durch einen Übertragungsfehler in der OPC-Kommunikation zu Stande und unterstreicht, dass die Bahnplanung nicht als Windows-Programm laufen

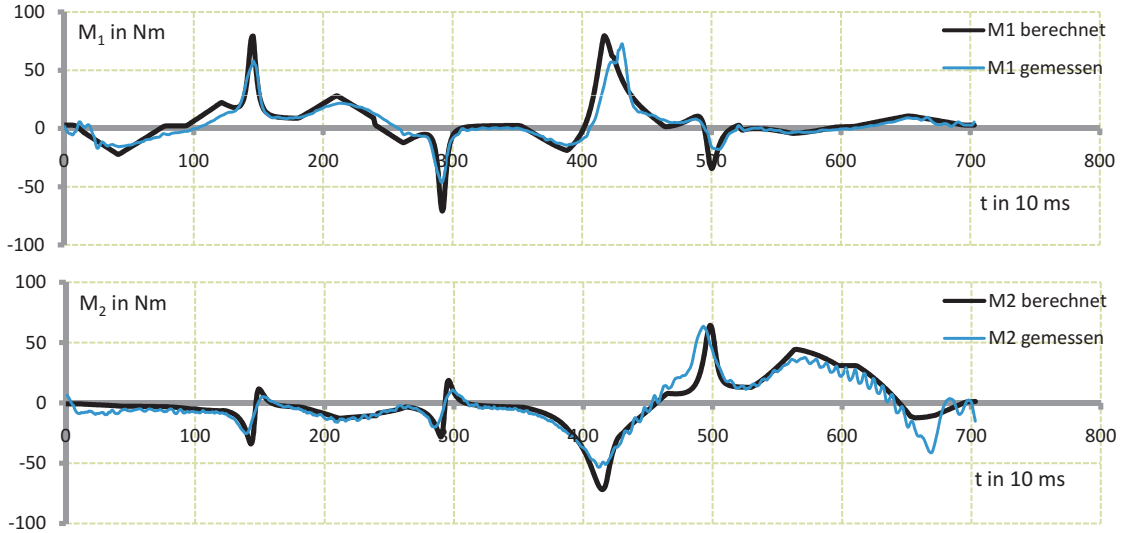


Abbildung 7.7: Berechnete und gemessene Drehmomentkurve für Bahn 2

und über OPC der SPS die Gelenkstellungen übergeben kann. OPC ist nicht echtzeitfähig, die Bahnplanung muss im Echtzeitkontext der SoftSPS ausgeführt werden. Die Abweichungen zwischen Soll- und Istbahn beträgt max. 3 cm im kartesischen Raum.

Abbildung 7.6 zeigt die Oszilloskop-Aufzeichung in MultiProg und Abbildung 7.7 die aufbereiteten Drehmomente.

Auswertung

In Bereichen mit hoher Bahngeschwindigkeit zeigt sich ein Bahnfehler. Dieser ist durch das unterschiedliche Regelverhalten der linearen Hubachse und der rotatorischen Armachse in der SoftSPS und durch die Übertagungsverzögerung zwischen der Beauftragung einer Achse und dem Erreichen der Ziellage von ca 200 ms zu begründen. Der Bahnfehler betrug in diesem Versuch maximal 3 cm. Dieser Bahnfehler ist nicht kritisch, da in KHS-Anwendungen hauptsächlich auf das exakte Erreichen der Ziellage von Bedeutung ist. Aufgrund des Bahnfehlers sollte für die Bahnplanung ein Sicherheitsabstand von mindestens 3 cm beim Definieren von Hindernissen beachtet werden.

Zudem müssen die motorspezifischen Maschinendaten exakt stimmen, aber das Einstellen der über 200 Reglerparameter pro Motor gestaltet sich als langwierig.

Versuchsbeschreibung

Im folgenden Versuch wird die online-Bahnplanung erprobt. Hierzu werden Hindernisse in den Arbeitsraum des Roboters aufgebaut. Der Roboter soll von beliebigen Positionen zu einem Ziel zwischen den Hindernissen fahren.

Durchführung

Die Roboterhand wird im Handbetrieb zunächst in die Nähe der Hinderniskanten gefahren. Die Gelenkstellungen werden im Debug-Modus von MultiProg abgelesen und als Kontur der Hindernisse der Bahnplanung mitgeteilt. Nachdem die Hindernisse vermessen und in den Konfigurationsraum übertragen wurden, kann die Bahnplanung mit Angabe von Start und Ziel Bahnen berechnen. Als Start dient immer die aktuelle Gelenkstellung, das Ziel ist eine feste Position zwischen den Hindernissen.

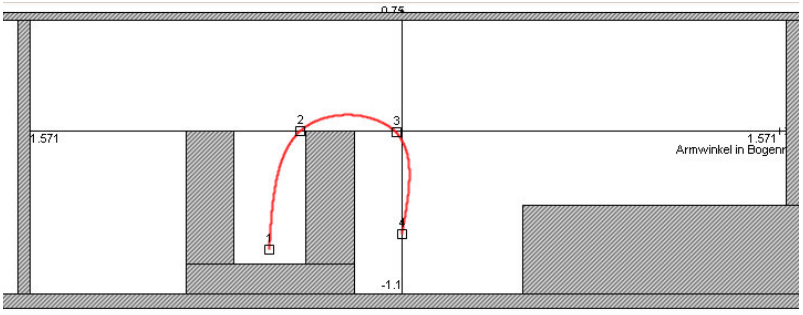


Abbildung 7.8: Berechnete Bahn zur Fahrt um Hindernis (aus Simulation)

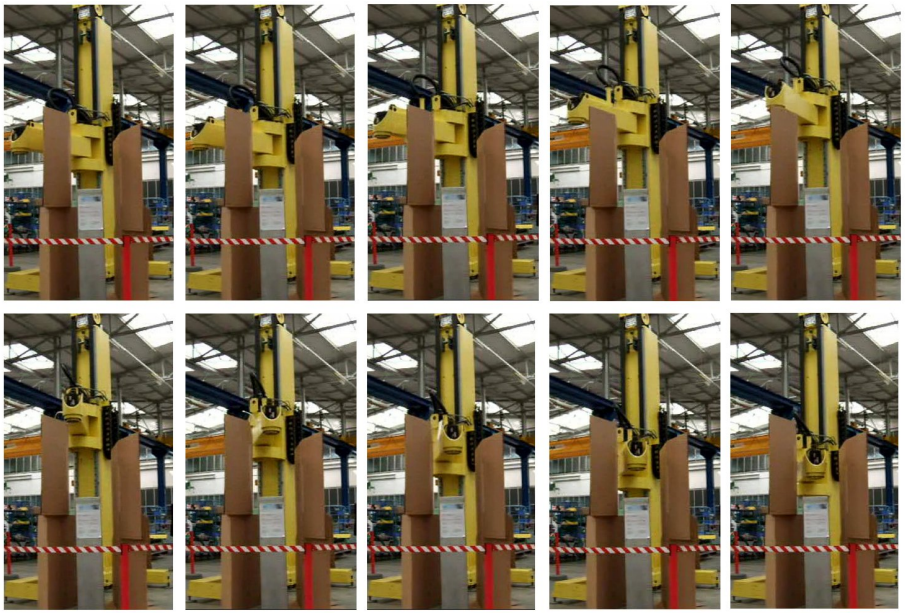


Abbildung 7.9: Fahrt des Testroboters über die online-berechnete Bahn aus Abbildung 7.8

Abbildungen 7.8 und 7.9 zeigen exemplarisch eine Bahn und die Fahrt im Hindernisbehafteten Raum.

Auswertung

Die Bahnplanung funktioniert wie erwartet und der Roboter bewegt sich kollisionsfrei zwischen den Hindernissen. Eine direkte Manipulation am Bahnregler oder eine unbedachte Umschaltung zwischen s- und t-Reglung führt dazu, dass der Roboter auf dem küzesten Weg zu der neuen Zielposition fährt und mit Hindernissen kollidiert. Hier muss die Software für den Praxiseinsatz mit weiteren Sichheitsmechanismen ausgestattet werden.

8. Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war die Entwicklung eines Systems zum zeitoptimalen Führen von Industrierobotern auf Bahnen im Arbeitsraum mit Hindernissen, wie es z.B. bei Handhabungsaufgaben beim Beladen und Packen gefordert wird. Das System sollte an einem Hubsäulenroboter praktisch erprobt werden. Die Funktionalität des Bahnplanungssystems wurde in die Bahnplanung mit Kollisionsvermeidung und in die Geschwindigkeitsberechnung mit Optimierung zerlegt. Die erste Funktion erstellt die Bahn um Hindernisse zum Ziel und die Geschwindigkeitsberechnung führt zu Trajektorien, welche die kinematischen und dynamischen Grenzen des Roboters nicht verletzen. Die Optimierung erzeugt einen weichen Bahnverlauf und minimiert die Fahrdauer. Ein Bahnregler setzt die Solltrajektorie auf die Antriebe um.

Die Grundprinzipien der eingesetzten Verfahren entsprechen dem Stand der Technik: Standardliteratur wie [Craig 05], [Truckenbrodt 94] vermitteln einen Überblick über die komplette Robotik und die Robotermodellierung, [Olomski 89] beschreibt die Erstellung des zeitoptimierten Geschwindigkeitsprofils und [Latombe 96] führt in Bahnplanungsverfahren ein. Für die Implementierung auf einer SPS-Steuerung und der online-Berechnung der Trajektorien wurden alternative Wege hinsichtlich ihrer Anwendbarkeit, Laufzeitkomplexität und Speicheranforderungen analysiert und mit Konzepten der aktuellen Forschung und mit neu erstellten Verfahren modifiziert und erweitert.

8.1 Zusammenfassung der Ergebnisse

Auf Basis der Hindernisbeschreibung erstellt die Bahnplanung eine kollisionsfreie Bahn von einem Startpunkt zum Ziel. Diese Bahn ist über eine Folge von Bahnstützpunkten definiert und wird anhand ihrer geometrischen Eigenschaften ohne die zeitintensive Berechnung der Fahrzeit optimiert. Alternativ kann der Benutzer Bahnstützpunkte numerisch, mit Teach-In oder über ein externes Programm vorgeben.

Die Trajektorie besteht aus der geometrischen Raumkurve und dem Zeitprofil. Die

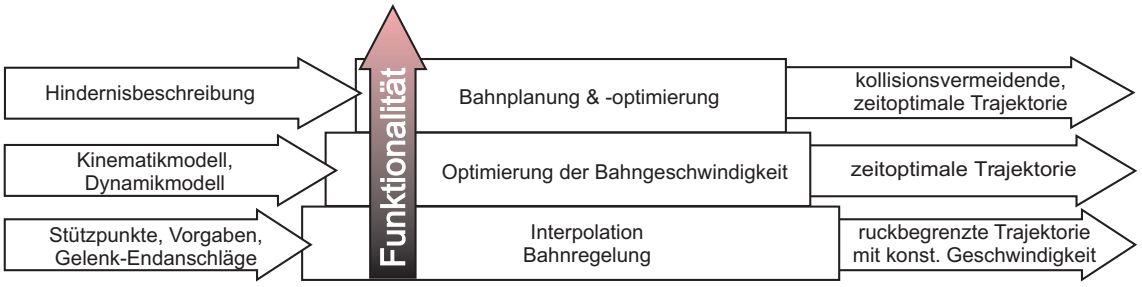


Abbildung 8.1: Bahnplanungssystem aus funktionaler Sicht. Interpolation und Bahnregelung bilden die funktionale Basis, Optimierung und Bahnplanung setzen darauf auf .

Bahnparametrierung ermöglicht eine getrennte Betrachtung und Optimierung dieser beiden Komponenten. Die Raumkurve basiert auf mehreren Bahnsegmenten, die sich jeweils aus der linearen oder der Spline-Interpolation zwischen den Bahnstützpunkten ergeben.

Der Benutzer gibt das Zeitprofil vor, indem er konkrete Bahngeschwindigkeiten und -beschleunigungen mit den Stützpunkten manuell verknüpft oder generelle Grenzwerte für Bahnruck- und beschleunigung vorgibt und das Geschwindigkeitsprofil automatisch berechnen lässt. Dieses Profil wird auf die minimale Fahrdauer optimiert und berücksichtigt Benutzervorgaben und dynamische Randbedingungen wie Drehmoment- und Drehzahlgrenzen der Antriebe.

Die Beachtung der dynamischen Randbedingungen resultiert in dem stetigen Verlauf der Trajektorie. Die Stetigkeit zeigt sich sowohl in der geometrischen Kontur der Bahn als auch in dem beschränkten Beschleunigunsanstieg im Geschwindigkeitsprofil. Sie garantiert, dass der Roboter der Soll-Trajektorie folgen kann.

Die praktischen Versuche am Roboter und Diskussionen mit KHS-Mitarbeitern zeigten, dass das Bahnplanungssystem sich nicht nur an wissenschaftlichen Maßstäben und seiner Funktionalität messen lässt, sondern nicht-funktionale Anforderungen wie Zuverlässigkeit, einfache Inbetriebnahme, Verständlichkeit und Wiederverwendbarkeit notwendig für die sinnvolle Verwendung im Produktionsbetrieb sind.

Aus Anwendersicht lässt sich das Bahnplanungssystem wie in Abbildung 8.1 in drei Ebenen gliedern. Mit der Interpolation und der Bahnregelung können bereits einfache Trajektorien gefahren werden, die zeitoptimale Berechnung und die Bahnplanung erweitern die Funktionalität.

Nicht alle Anforderungen aus Kapitel 6 konnten aus Zeitgründen im Rahmen dieser Arbeit erschöpfend realisiert werden. Die Implementierung des Systems beanspruchte den Großteil der Zeit. Ob Ausnahmesituationen und die Fehlerbehandlung wie in Abbildung 8.2a) vom Anwenderprogramm individuell oder vom Bahnplanungssystem allgemein behandelt werden sollen, wird erst ein längerer praktischer Einsatz des System zeigen können.

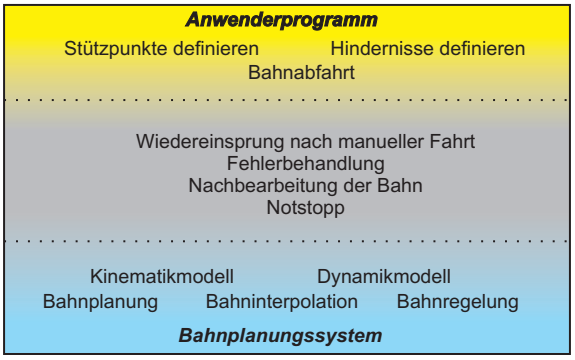


Abbildung 8.2: Die Systemgrenze zwischen Bahnplanungssystem und Andwenderprogramm ist nicht eindeutig definiert

8.2 Erweiterungen und Folgearbeiten

Während der Bearbeitung des Problems der zeitoptimalen Bahnplanung wurden nebenläufig auch weitere Defizite aufgedeckt, die Folgearbeiten ermöglichen.

- Die zeitoptimale Fahrt eines Roboters ist nicht immer gewünscht. Die Last und ihre Randbedingungen sollten als Kriterien für die weitere **Optimierung** des aktuellen Bahnplanungssystems für den Prouktionseinsatz modelliert werden, da ihre Eigenschaften auf die Trajektorie Auswirkungen hat. Roboter mit Unterdruckgreifern müssen z. B. oft langsamer als mit der zeitoptimalen Geschwindigkeit fahren, um die Last nicht zu verlieren. Diese Aufgabenspezifischen Randbedingungen können zu unterschiedlichen Ausprägungen des Optimierungsverfahrens je nach Einsatz des Roboters führen.

- Analog zur **humanzentrierten Bedienung** sollte ein Weg gefunden werden, die Erfahrung des Bedieners und die analytische Bahnoptimierung zu kombinieren. Ein Mensch kann mit seiner jahrelangen Erfahrung der Bahnberechnung assistieren, denn er kann intuitiv kollisionsfreie und gute Bahnen erkennen und erzeugen. Eine vollautomatische Berechnung deckt nicht alle Sonderfälle ab und ist komplex auf neue Szenarien zu transferieren.

- Mittelfristig sollte die zeitoptimale Bahnplanung eines Roboters zu einer mehrziel-optimierten Bahnplanung eines **Multi-Robotersystems** erweitert werden, wie es in KHS Verpackungsanlagen vorkommt und z. B. vier Säulenroboter sich in einem gemeinsamen Arbeitsraum bewegen. Ein Modell dieser kundenspezifischen Systeme muss dennoch möglichst allgemein gehalten werden und die zeitliche Koordinierung der Roboter und sonstigen Aktuatoren optimieren. Zur Zeit wird diese Koordinierung manuell durch den Entwickler durchgeführt und führt nicht zu optimalen Bewegungen des Gesamtsystems.

- Die **Systemsoftware** und SPS-Sprachen für Kuka Steuerung haben im Vergleich zu aktuellen PC-Programmiersprachen nur sehr beschränkte Ausdrucks- und Testmöglichkeiten. Eine komplexes Programm wie Bahnplanung verlangt nach ausgiebigen und automatisierten Tests mit einer entsprechenden Visualisierung der internen Vorgänge, auch um neue Erkenntnisse aus der Bahnplanung und die Wirkung der Parameter zu gewinnen. Die Implementierung einer

Simulation als PC-Programm und die anschließende Portierung der Kernfunktionalität in SPS-Sprachen wie in dieser Arbeit erweist sich als umständlich und fehlerträchtig. Die Kuka-SoftSPS ist zukunftsweisend, aber die Relikte der altmodischen SPS sind gegen Paradigmen moderner Programmiersprachen nicht zeitgemäß, da sie den Freiraum der Lösungsmöglichkeiten und die Kreativität des Programmierers beschränken. Der Einsatz von Hochsprachen in eingebetteten Systemen stellt kein Widerspruch dar: Wie in [Kretschmann 06b] gezeigt, kommt in Smart Cards als Programmiersprache Java zum Einsatz und erfüllt Sicherheitsanforderungen im Sinne von Security (Datensicherheit). Um auch Safety-Eigenschaften (Personen- und Sachschutz) zu gewährleisten, müssen Programmiersprachen um Echtzeit-Mechanismen erweitert werden.

- Komplette Anlagen abdeckende **Simulationen** zur Analyse des dynamischem Systemverhaltens sind kein fester Bestandteil der Softwareentwicklung der KHS-Palettier- und Verpackungsanlagen. Außer im Bereich der automatischen Lagenbildung findet die Optimierung bisher auf Basis der Erfahrung der Mitarbeiter und empirisch mit der praktischen Erprobung direkt an der Anlage statt. Eine Simulation lässt sich für Versuche wesentlich leichter modifizieren als die aufgebaute Anlage und ermöglicht Untersuchungen, während das reale System noch nicht existiert oder sein Aufbau zu aufwändig wäre. Simulationen tangieren durch eine mögliche kostengünstige und gefahrlose Ausbildung und Know-How-Transfer an den Service auch andere Geschäftsbereiche. Obwohl KHS Individual-Anlagen liefert, wäre die Fertigstellung der kompletten Anlage innerhalb von drei Monaten ohne den Einsatz von Standardkomponenten nicht möglich. Hier sollte untersucht werden, wie die geometrischen Modelle der mechanischen Konstruktion um Dynamik und Softwareschnittstellen erweitert werden können, wie passende Modelle gefunden und verifiziert werden können und in welchem Kontext die Simulationsergebnisse auf die Realität tranferierbar sind.

- Ein Anwendungsprogrammierer muss physische Hindernisse exakt beschreiben, damit erzeugte Bahnen kollisionsfrei sind. Dieser fehleranfällige Prozess sollte durch eine Verarbeitung des geometrischen Modells oder direkt durch **Sensoreinsatz** automatisiert werden. Kamerasysteme sind bei Entladern von KEG-Fässern bereits im Einsatz. Trotz der höheren Kosten und des Wartungsaufwands werden Sensorsysteme zukünftig auch in der industriellen Robotik mehr Verwendung finden und Fertigungssystemen langfristig mehr Autonomie verleihen. Dies ist auch Gegenstand der aktuellen Grundlagenforschung.

Literaturverzeichnis

[Anton 04] Stefan Anton. Inverse Kinematik am Robotersimulationsprogramm EASY-ROB. 2004.

[Aurnhammer 04] Andreas Aurnhammer. *Optimale stochastische Trajektorienplanung und Regelung von Industrierobotern.* Dissertation, Universität der Bundeswehr München: Fakultät für Luft- und Raumfahrttechnik, 2004.

[Berns 05] K. Berns. Vorlesungskript zu Eingebettete Systeme und Robotik. agrosy.informatik.uni-kl.de, 2005.

[Berns 06] H. Schäfer; K. Berns (Hrsg.). *RAVON - An autonomous Vehicle for Off-road Navigation.* European Land Robot Trial - ELROB, 2006.

[Bobrow 88] J. E. Bobrow. Optimal robot pathplanning using the minimum-time criterion. In *Robotics and Automation, IEEE Journal of*, S. 443–450, 1988.

[Bobrow 98] J.-K. Park; J. E. Bobrow. Minimum-time trajectory planning for a robot manipulator amid obstacles. In *Proceedings of the 4th Japan-France Congress and 2nd Asia-Europe Congress on Mechatronics*, S. 369–374, 1998.

[Boor 77] C. De Boor. *A Practical Guide to Splines.* Springer, 1977.

[Chen 03] Heping Chen; Ning Xi; Yifan Chen. Multi-objective optimal robot path planning in manufacturing. In *Intelligent Robots and Systems, 2003 IEEE/RSJ International Conference on*, S. 1167 – 1172, 2003.

[Christaller 03] Alois Christaller, Thomas; Knoll. *Robotik.* Fischer Verlag, 2003.

[Craig 05] John J. Craig. *Introduction to Robotics - Mechanics and Control.* New Jork: Pearson Prentice Hall, Ausgabe 3. Aufl., 2005.

[Croft 00] D. Constantinescu; E.A. Croft (Hrsg.). *Smooth and Time-Optimal Trajectory Planning for Industrial Manipulators Along Specified Paths.* University of British Columbia, 2000.

[Dawson 01] W.E. Dixon; D. Moses; I.D. Walker; D.M. Dawson. A Simulink-based Robotic Toolkit for Simulation and Control of the PUMA 560 Robot Manipulator. In *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, S. 2202–2207, 2001.

[de Casteljau 59] P. de Casteljau. Outillage Methodes Calcul. Andre Citroen Automobiles SA Paris, 1959.

[Fourquet 90] J.Y. Fourquet. Time-Optimal PTP Motion including Dynamics for Robotic Manipulators. In *Industrial Electronics Society, IECON '90., 16th Annual Conference of IEEE*, S. 220–225, 1990.

[Group 06] ;KUKA Robot Group. Produktdokumentation: KUKA.eXtended Motion (XM) - System zur Steuerung kundenspezifischer Mechaniken. Kuka, 2006.

[Hartenberg 55] J. Denavit; R.S. Hartenberg. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *ASME Journal of Applied Mechanics, S. 215-221*, 1955.

[Hernando 02] E. Hernando, M.; Gambao. Visibility analysis and genetic algorithms for fast robot motion planning. In *Intelligent Robots and System, IEEE/RSJ International Conference on*, S. 2413– 2418, 2002.

[Hirzinger 97] E. Ralli; G. Hirzinger. Robot path planning using Kohonen maps. In *Intelligent Robots and Systems IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, S. 1224–1229, 1997.

[Hsu 00] D. Hsu, R. Kindel, J. Latombe, S. Rock. Randomized kinodynamic motion planning with moving obstacles, 2000.

[ISG 01] Industrielle Steuerungstechnik GmbH ISG. *ISG Motion Control Platform für PLCOpen - Version: KMC-LS2 unter Verwendung der PLC-Umgebung Multiprog der Fa. KW-Software*, 2001.

[Johanni 88] Rainer Johanni. *Optimale Bahnplanung bei Industrierobotern*. Dissertation, Universität München, 1988.

[KBSt 07] Koordinierungs-und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung KBSt. V-Modell-XT 1.2.1. www.v-modell-xt.de, 2007.

[Khosla 88] S. Ramos; Pradeep Khosla. A comparative analysis of the hardware requirements for the Lagrange-Euler and Newton-Euler dynamics formulations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '88)*, S. 291 – 296, April 1988.

[KHS 06] Maschinen-und Anlagenbau Aktiengesellschaft KHS. Geschäftsbericht. www.khs.com, 2006.

[KHS 07a] Maschinen-und Anlagenbau Aktiengesellschaft KHS. Produktbroschüren: Innopal/Innopack Robotertechnik, Transportanlagen für Ein- und Mehrweggebinde, Innopack Packtechnik, Verpackungsanlagen für eine flexiblere Produktion, Innopal Palettier-Roboter, Innopal Palettier-Roboter für Mehrweggebinde, KHS Anlagentechnik Logistik, Innoline Anlagen-Informations-System AIS. www.khs.com, 2007.

[KHS 07b] Maschinen-und Anlagenbau Aktiengesellschaft Worms KHS. Schulung KHS Verpackungsmaschinen mit SIMATIC Step7 Steuerungssystem, 2007.

[KHS 07c] Maschinen-und Anlagenbau Aktiengesellschaft Worms KHS. Schulung Kuka-Steuerungssystem, 2007.

[Kretschmann 06a] R. Kretschmann. COMROS. In *Seminarbericht: Steuerungsarchitekturen für intelligente Roboter, Universität Kaiserslautern: AG Robotersysteme*, S. 55–72, 2006.

[Kretschmann 06b] R. Kretschmann. Integration des SmartMX Controllers in Arygon NFC-Produkte. Projektarbeit, Universität Kaiserslautern: AG Robotersysteme, 2006.

[Kretschmann 06c] Sebastian Blank; Torsten Gilz; Ines Heck; Marcel Jung; Michael Jung; Ralf Kretschmann. Robotersystem der Gruppe 2. In *Praktikumsbericht: Aufbau eines autonomen Gabelstaplers zur Durchführung von Transportaufgaben, Universität Kaiserslautern: AG Robotersysteme*, S. 65–83, 2006.

[Krieger 04] Daniel Krieger. Aufbau einer dynamischen flexiblen Sicherheitsumgebung für Roboter. Diplomarbeit, Universität Kaiserslautern: Arbeitsgruppe Robotersysteme, 2004.

[Kuka 07a] Kuka. Produktbroschüren. www.kuka.de, 2007.

[Kuka 07b] FPT; Güdel; KHS; Kuka. Vortragsreihe Kuka Inside 2007 - Synergien, Chancen und Mehrwert: KR C2, 16.-17.2007, 2007.

[Latombe 96] Jean-Claude Latombe. *Robot motion planning.* Boston: Kluwer Acad. Publ., Ausgabe 4. Aufl., 1996.

[Lepers 05] Heinrich Lepers. *SPS-Programmierung nach IEC 61131-3. Mit Beispielen für CoDeSys und STEP 7.* Franzis Verlag, 2005.

[Lin 96] H. Ozaki; C.-J. Lin. Optimal B-Spline Joint Trajectory Generation for Collision-Free Movements of a Manipulator under Dynamic Constraints. In *Robotics and Automation, IEEE International Conference on*, S. 3592–3597, 1996.

[MCA 07] MCA. Modular Controller Architecture. www.mca2.org, 2007.

[Microsoft 07] Microsoft. Microsoft Robotics Studio. www.microsoft.com, 2007.

[Olomski 89] Jürgen Olomski. *Bahnplanung und Bahnführung von Industrierobotern.* Dissertation, Universität Braunschweig: Institut für Regelungstechnik, 1989.

[PLCOpen 01] Technical Committee 2 Task Force Motion Control PLCOpen. Function blocks for motion control - Version 1.0. Technischer Bericht, PLCOpen - Standardization in Industrial Control Programming, 2001.

[Press 92] W. H. Press. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[Robotersysteme 07] AG Robotersysteme. Universität Kaiserslautern: Webseite der AG Robotersysteme. agrosy.informatik.uni-kl.de, 2007.

[Schäfer 06] Johannes Schäfer. Point based path control of generically constructed manipulators. Diplomarbeit, Universität Kaiserslautern: Arbeitsgruppe Robotersysteme, 2006.

[SEW 07] SEW-Eurodrive GmbH Co KG SEW. Motordatenblätter. www.sew.de, 2007.

[Soderkvist] Tomas Berglund; Hakan Jonsson; Inge Soderkvist. An Obstacle-Avoiding Minimum Variation B-Spline Problem. *gmag*.

[Sohn 04] Martin Sohn. Automatische Steuerungsanpassung und Bahngenerierung von Industrierobotern. Diplomarbeit, Universität Kaiserslautern: Arbeitsgruppe Robotersysteme, 2004.

[Sommerville 01] Ian Sommerville. *Software Engineering*. Pearson Studium, 2001.

[Tolan 02] M. Tolan. Vorlesungskript zu Physik A1. e1.physik.uni-dortmund.de/physik_a1/A1_11.pdf, 2002.

[Truckenbrodt 94] E. Kreuzer; J. Lugtenburg; H. Meißner; A. Truckenbrodt. *Industrieroboter - Technik, Berechnung und anwendungsorientierte Auslegung*. Berlin: Springer, 1994.

[Walser 01] Hans Walser. *The Golden Section*. The Mathematical Association of America, 2001.

[Wikipedia 07] Wikipedia. Webseite von Wikipedia. www.wikipedia.org, 2007.

[Wong 96] Z. Shiller; H. Chang; V. Wong. The Practical Implementation of Time-Optimal Control for Robotic Manipulators. In *Robotics and Computer-Integrate Manufacturing*, S. 29–39, 1996.

[Zucker 86] K. Kant; S.W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. In *International Journal of Robotics Research*, S. 72–89, 1986.

A. Denavit-Hartenberg Transformation

Berechnung der Lage des Endeffektors abhängig von den Gelenkwinkelstellungen: direkte Kinematik $\vec{X} = f(\vec{q})$.

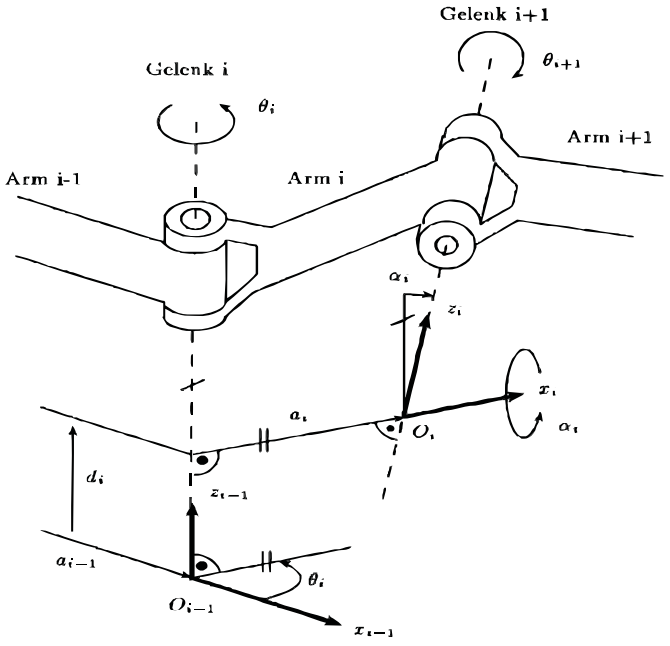


Abbildung A.1: Lage der Koordinatensysteme der Armsegmente, aus [Berns 05]

Rotation φ_i um z_{i-1}-Achse:

$$\underline{R}_{z_{i-1}}(\varphi_i) = \begin{bmatrix} \cos\varphi_i & -\sin\varphi_i & 0 & 0 \\ \sin\varphi_i & \cos\varphi_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

Translation d_i entlang z_{i-1}-Achse:

$$\underline{T}_{z_{i-1}}(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

Translation a_i entlang x_i-Achse:

$$\underline{T}_{x_i}(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

Rotation α_i um x_i-Achse:

$$\underline{R}_{x_i}(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

Verkettung der Matrizen zu einer kompletten Transformation:

$$\underline{A}_{i-1,i} = \underline{R}_{z_{i-1}}(\varphi_i) \cdot \underline{T}_{z_{i-1}}(d_i) \cdot \underline{T}_{x_i}(a_i) \cdot \underline{R}_{x_i}(\alpha_i) = \begin{bmatrix} \cos\varphi_i & -\sin\varphi_i \cdot \cos\alpha_i & \sin\varphi_i \cdot \sin\alpha_i & a_i \cdot \cos\varphi_i \\ \sin\varphi_i & \cos\varphi_i \cdot \cos\alpha_i & -\cos\varphi_i \cdot \sin\alpha_i & a_i \cdot \sin\varphi_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

Verkettung der DH-Matrizen in Reihenfolge der n Armsegmente liefert die Lage des Tool Center Points:

$$\underline{X}_{TCP}\left(\vec{\varphi}, \vec{d}\right) = \prod_{i=1}^{n} \underline{A}_{i-1,i}(\varphi_i, d_i) \tag{A.6}$$

B. Berechnung der inversen Kinematik

Gesucht ist die Gelenkwinkelstellung $\vec{q} = (d_1, \theta_2, \theta_3, \theta_4)$, um den kartesischen Punkt $\vec{X}$ anzufahren, vgl. mit Abbildung B.1: inverse Kinematik $\vec{q} = f^{-1}\left(\vec{X}\right)$.

Vorgegeben sei die Solllage des Endeffektors im Weltkoordinatensystem $\vec{X} = \begin{pmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{pmatrix}$

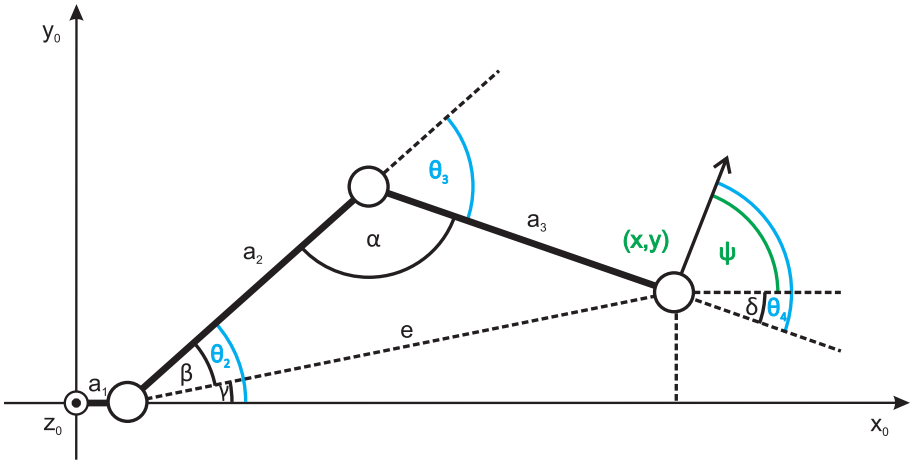


Abbildung B.1: Draufsicht auf den Manipulator mit Bezeichnungen der Winkel und Längen. x, y, ψ sind gegeben, $\theta_2, \theta_3, \theta_4$ werden gesucht .

$$\begin{aligned}
\text{Satz von Pythagoras}: \; e^2 &= (x-a_1)^2 + y^2 \\
\text{Cosinussatz}: \; e^2 &= a_2^2 + a_3^2 - 2\,a_2\,a_3\,\cos\,\alpha \\
\Rightarrow \; \alpha &= \arccos\left(-\frac{(x-a_1)^2 + y^2 - a_2^2 - a_3^2}{2\,a_2\,a_3}\right) \\
\Rightarrow \; \theta_3 = \alpha - \pi &= \arccos\left(-\frac{(x-a_1)^2 + y^2 - a_2^2 - a_3^2}{2\,a_2\,a_3}\right) - \pi
\end{aligned} \tag{B.1}$$

$$\begin{aligned}
\text{analog}: \; \beta &= \arccos\left(-\frac{a_3^2 - a_2^2 - (x-a_1)^2 - y^2}{2\,a_2\sqrt{(x-a_1)^2 + y^2}}\right) \\
\gamma &= \arctan\left(\frac{y}{x-a_1}\right) \\
\Rightarrow \; \theta_2 = \beta + \gamma &= \arccos\left(-\frac{a_3^2 - a_2^2 - (x-a_1)^2 - y^2}{2\,a_2\sqrt{(x-a_1)^2 + y^2}}\right) + \arctan\left(\frac{y}{x-a_1}\right)
\end{aligned} \tag{B.2}$$

$$\theta_4 = \psi - \theta_2 - \theta_3 \tag{B.3}$$

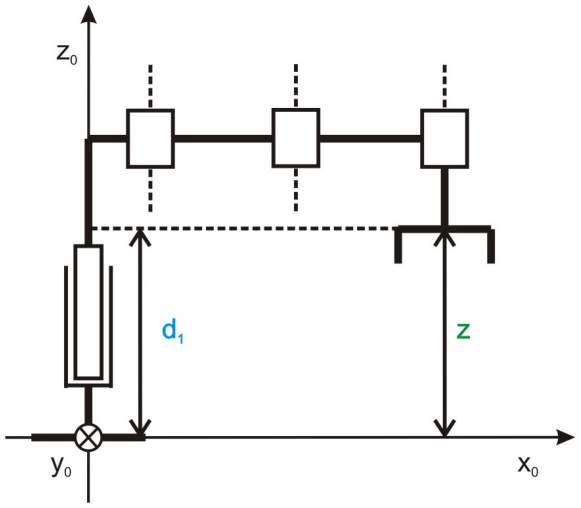


Abbildung B.2: Seitenansicht des Manipulators. z ist gegeben, d_1 ist gesucht .

Abbildung B.2 zeigt den Roboter von der Seite. Die kinematische Beschreibung ist so gewählt, dass d_1 identisch mit z ist.

$$d_1 = z \tag{B.4}$$

Gleichungen B.1 bis B.4 liefern die Gelenkstellungen und beschreiben die inverse Kinematik f^{-1}. Die trigonometrischen Umkehrfunktionen sind nicht eindeutig und können falsche Winkel liefern, Bsp: $\arctan(\tan\pi) \neq \pi$.

Im Folgenden wird die Funktionen atan2 eingeführt, die das Vorzeichen der Argumente ausgewertet und den Quadrant des Ergebnisses eindeutig bestimmt.

$$\text{atan2} : \mathbb{R}^2 \longmapsto]-\pi, \pi] \text{ mit } \text{atan2}(y,x) := \begin{cases} \arctan\frac{y}{x} & \text{für } x > 0 \\ \arctan\frac{y}{x} + \pi & \text{für } x < 0,\ y \geq 0 \\ \arctan\frac{y}{x} - \pi & \text{für } x < 0,\ y < 0 \\ \frac{\pi}{2} & \text{für } x = 0,\ y > 0 \\ -\frac{\pi}{2} & \text{für } x = 0,\ y < 0 \\ 0 & \text{für } x = 0,\ y = 0 \end{cases} \tag{B.5}$$

Mit den dieser Funktion kann die Rückwärtskinematik korrekt beschrieben werden:

$$\begin{aligned} d_1 &= z \\ \theta_2 &= \arccos\left(-\frac{a_3^2 - a_2^2 - (x-a_1)^2 - y^2}{2\,a_2\sqrt{(x-a_1)^2 + y^2}}\right) + \text{atan2}(y, x-a_1) \\ \theta_3 &= \alpha - \pi = \arccos\left(-\frac{(x-a_1)^2 + y^2 - a_2^2 - a_3^2}{2\,a_2\,a_3}\right) - \pi \\ \theta_4 &= \psi - \theta_2 - \theta_3 \end{aligned} \tag{B.6}$$

C. Berechnung der Dynamik

4-achsiger SCARA-Roboter

Aus (3.21) bis (3.26) folgt die Lagrange-Funktion L des SCARA-Roboters mit

$$
\begin{aligned}
&L\left(d1, \theta_2, \theta_3, v_1, \omega_2, \omega_3, \omega_4\right) = \\
&\left(\left(\frac{1}{2}\, M_3\, a_2\, a_3 + a_2\, a_3\ \left(M_4 + M_l\right)\right)\ \left(\cos\left(\theta_2\right)\ \cos\left(\theta_2 + \theta_3\right) + \sin\left(\theta_2\right)\ \sin\left(\theta_2 + \theta_3\right)\right)\right. \\
&\left. + \frac{1}{4}\, a_3^2\ \left(M_3 + 4\ \left(M_4 + M_l\right)\right)\right)\, \omega_2\, \omega_3 \\
&+\left(\left(\frac{1}{2}\, M_3\, a_2\, a_3 + a_2\, a_3\ \left(M_4 + M_l\right)\right)\ \left(\cos\left(\theta_2\right)\ \cos\left(\theta_2 + \theta_3\right) + \sin\left(\theta_2\right)\ \sin\left(\theta_2 + \theta_3\right)\right)\right. \\
&\left. + \frac{1}{8}\ \left(4\, I_{a2} + 4\, I_{z2} + M_2\, a_2^2 + M_3\ \left(4\, a_2^2 + a_3^2\right) + 4\ \left(M_4 + M_l\right)\ \left(a_2^2 + a_3^2\right)\right)\right)\, \omega_2^2 \\
&+ \frac{1}{2}\, I_{a4}\, \omega_4^2 + \frac{1}{8}\ \left(4\, I_{a3} + a_3^2\ \left(M_3 + 4\ \left(M_4 + M_l\right)\right)\right)\ \omega_3^2 \\
&+ \frac{1}{2}\ \left(M_1 + M_2 + M_3 + M_4 + M_{a1} + M_l\right)\ v_1^2 \\
&- g\ \left(M_1 + M_2 + M_3 + M_4 + M_l\right)\ d_1
\end{aligned}
\tag{C.1}
$$

Die Differenzierte Lagrangefunktion des SCARAs für das Drehmoment des Gelenks 1:

$$\begin{aligned}
\frac{\partial L}{\partial \dot{d}_1} &= v_1 \ (I_{a1} + M_1 + M_2 + M_3 + M_4 + M_l) \\
\frac{\partial L}{\partial d_1} &= -g \ (M_1 + M_2 + M_2 + M4 + M_l) \\
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{d}_1}\right) - \frac{\partial L}{\partial d_1} &= \qquad (C.2) \\
(M_{a1} + M_1 + M_2 + M_3 + M_4 + M_l) \ \dot{v}_1 & \\
+g \ (M_1 + M_2 + M_2 + M4 + M_l) &
\end{aligned}$$

analog Gelenk 2 für SCARA:

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\omega}_2}\right) - \frac{\partial L}{\partial \omega_2} = & \\
\frac{1}{2}\, a_2\, a_3 \ (M_3 + 2 \ (M_4 + M_l)) \ (\sin(\theta_2) \cos(\theta_2 + \theta_3) - \cos(\theta_2) \sin(\theta_2 + \theta_3)) \ \omega_3^2 & \\
+a_2\, a_3 \ (M_3 + 2 \ (M_4 + M_l)) \ (\sin(\theta_2) \cos(\theta_2 + \theta_3) - \cos(\theta_2) \sin(\theta_2 + \theta_3)) \ \omega_2\, \omega_3 & \\
+(a_2\, a_3 \ (M_3 + 2 \ (M_4 + M_l)) \ (\cos(\theta_2) \cos(\theta_2 + \theta_3) + \sin(\theta_2) \sin(\theta_2 + \theta_3)) & \\
+\frac{1}{4}\,(4\, I a_2 + 4\, I_{z2} + 4\, I_{z3} + 4\, I_{z4} + 4\, I_{zl} & \\
+M_2\, a_2^2 + M_3 \ \left(4\, a_2^2 + a_3^2\right) + 4 \ (M_4 + M_l) \ \left(a_2^2 + a_3^2\right))\, \dot{\omega}_2 & \\
+(\frac{1}{2}\, a_2\, a_3 \ (M_3 + 2 \ (M_4 + M_l)) \ (\cos(\theta_2) \cos(\theta_2 + \theta_3) + \sin(\theta_2) \sin(\theta_2 + \theta_3)) & \\
+\frac{1}{4} \ \left(4\, I_{z3} + 4\, I_{z4} + 4\, I_{zl} + M_3\, a_3^2 + 4\, a_3^2 \ (M_4 + M_l)\right))\, \dot{\omega}_3 & \\
+ (I_{z4} + I_{zl}) \ \dot{\omega}_4 & \\
\qquad (C.3)
\end{aligned}$$

SCARA-Gelenk 3:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\omega}_3}\right) - \frac{\partial L}{\partial \omega_3} =$$

$$\frac{1}{2}\, a_2\, a_3\, \left(M_3 + 2\, \left(M_4 + M_l\right)\right)\, \left(\sin\left(\theta_2\right)\, \cos\left(\theta_2 + \theta_3\right) - \cos\left(\theta_2\right)\, \sin\left(\theta_2 + \theta_3\right)\right)\, \omega_2\, \omega_3$$

$$+(\frac{1}{2}\, a_2\, a_3\, \left(M_3 + 2\, \left(M_4 + M_l\right)\right)\, \left(\cos\left(\theta_2\right)\, \cos\left(\theta_2 + \theta_3\right) + \sin\left(\theta_2\right)\, \sin\left(\theta_2 + \theta_3\right)\right)$$
$$+\frac{1}{4}\, \left(4\, I_{z3} + 4\, I_{z4} + 4\, I_{zl} + a_3^2\, \left(M_3 + 4\, \left(M_4 + M_l\right)\right)\right))\, \dot{\omega}_2$$

$$+\frac{1}{4}\, \left(4\, Ia_3 + 4\, I_{z3} + 4\, I_{z4} + 4\, I_{zl} + a_3^2\, \left(M_3 + 4\, \left(M_4 + M_l\right)\right)\right)\, \dot{\omega}_3$$

$$+\left(I_{z4} + I_{zl}\right)\, \dot{\omega}_4 \tag{C.4}$$

SCARA-Gelenk 4:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\omega}_4}\right) - \frac{\partial L}{\partial \omega_4} = \tag{C.5}$$

$$\left(I_{z4} + I_{zl}\right)\, \dot{\omega}_2 + \left(I_{z4} + I_{zl}\right)\, \dot{\omega}_3 + \left(Ia_4 + I_{z4} + I_{zl}\right)\, \dot{\omega}_4$$

3-achsiger Hubsäulenroboter

Die dynamischen Gleichungen des Einlegers ergeben sich aus den SCARA-Gleichungen, wenn Gelenk 4 vernachlässigt wird. Die differenzierte Lagrangefunktion des 3-achsigen Einlegers für Drehmoment des Gelenks 1 berechnet sich zu

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{d}_1}\right) - \frac{\partial L}{\partial d_1} =$$
$$\left(I_{a1} + M_1 + M_2 + M_3 + M_4 + M_l\right)\, \dot{v}_1 \tag{C.6}$$

$$+g\, \left(M_1 + M_2 + M_2 + M4 + M_l\right)\, .$$

Gelenk 2:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{d}_1}\right) - \frac{\partial L}{\partial d_1} = \\ \left(\frac{1}{4}M_2 a_2^2 + I_{z2} + (M_3 + M_L)\, a_2^2 + I_{z3} + I_{zL3}\right)\dot{\omega}_2 \\ + (I_{z3} + I_{zL3})\,\dot{\omega}_3 \end{aligned} \tag{C.7}$$

Gelenk 3:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{d}_1}\right) - \frac{\partial L}{\partial d_1} = \\ (I_{z3} + I_{zL3})\,\dot{\omega}_3 \\ + (I_{z3} + I_{zL3})\,\dot{\omega}_2 \end{aligned} \tag{C.8}$$

2-achsiger Hubsäulenroboter

Die Berechnung des 2-achsigen Roboters gestaltet sich analog zur 3-achsigen Maschine, indem das dritte Gelenk vernachlässigt wird. Die Berechnung wird hier nicht weiter erläutert.

D. Numerische Algorithmen

Tridiagonale Spline-Initialisierung

Bei der Spline-Intitialisierung wird die Ableitung am Beginn und Ende des Splines übergeben. Bahnwert, 1. Ableitung und 2. Ableitung sind mit y, y1, y2 bezeichnet. Das Array InterpolType beinhaltet, ob ein Segment linear oder mit Splines interpoliert werden soll; 1 bedeutet linear, 2 Spline. Vor der Spline-Initialisierung wurde die lineare Initialisierung ausgeführt und y1 berechnet. Bei einem Übergang zwischen linear und Spline muss die Steigung des linearen Stücks berücksichtigt und die 2. Ableitung abhängig von dem Abstand der Stützstellen angepasst werden.

Listing D.1: Interpolation-Initialisierung für Splines und lineare Segmente

```
Private Sub initFastQubicSpline(ByVal difstart As Double, _
        ByVal difEnd As Double)
    Dim n As Integer
    n = y.Length
    ReDim y2(n - 1)
    Dim u As Double()
    ReDim u(n)

    Dim splineEnd(n - 1) As Boolean
    Dim splineStart(n - 1) As Boolean

    If interpolType(0) = 2 Then
        splineStart(0) = True
    End If
    For i As Integer = 1 To n - 1
        splineStart(i) = False
        splineEnd(i) = False
        If (interpolType(i - 1) = 1) And (interpolType(i) = 2) Then
            splineStart(i) = True
        End If
        If (interpolType(i - 1) = 2) And (interpolType(i) = 1) Then
            splineEnd(i) = True
        End If
    Next
```

```
    If difstart <> 0 Then
        y2(0) = -0.5
        u(0) = 3.0 * (y1(0) - difstart)
    Else
        y2(0) = 0
        u(0) = 0
    End If

    Dim splineActive As Boolean = splineStart(0)
    For i As Integer = 1 To n - 2
        If splineEnd(i) Then
            splineActive = False
            u(i) = (3 / (2 - 1)) * (y1(i) - (y(i) - y(i - 1) _
                / (2 - 1)))
            y2(i) = (u(i) - 0.5 * u(i - 1)) / (0.5 * y2(i - 1) + 1)
        ElseIf splineActive And (Not splineStart(i)) Then
            y2(i) = -1.0 / (4.0 + y2(i - 1))
            u(i) = (6.0 * (y(i + 1) - 2.0 * y(i) + y(i - 1)) _
                - u(i - 1)) / (4.0 + y2(i - 1))
        Else
            splineActive = True
            y2(i) = -0.5
            u(i) = (3 / (1 - 0)) * ((y(i + 1) - y(i) / (1 - 0) _
                - y1(i - 1)))
        End If
    Next
     If difEnd <> 0 Then
        u(n - 1) = 3.0 * (difEnd - y1(n - 2))
        y2(n - 1) = (u(n - 1) - 0.5 * u(n - 2)) _
            / (0.5 * y2(n - 2) + 1)
    Else
        u(n - 1) = 0
        y2(n - 1) = 0
    End If
    For i As Integer = n - 2 To 0 Step -1
        If (interpolType(i) = 1) And (interpolType(i + 1) = 1) Then
            splineStart(i + 1) = True
        End If
        If (Not splineStart(i + 1)) Then
            y2(i) = y2(i) * y2(i + 1) + u(i)
        End If
    Next
End Sub
```

E. Simulation

Dieser Abschnitt skizziert die Funktionen der erstellten Simulation. Mit ihrer Hilfe wurden Konzepte der Arbeit getestet und verfeinert. Sie ermöglicht die grafische Eingabe von Bahnstützpunkten und visualisiert die internen Vorgänge der Bahnplanung und Geschwindigkeitsberechnung. Die Kernfunktionen der Bahnplanung wurden zunächst in der Simulation implementiert und konnten danach auf die SPS übertragen werden, als ein Roboter für die Tests zur Verfügung stand. Die Simulation wurde in Microsoft Visual Basic 2005 erstellt, die Komponente zur OPC-Kommunikation *Local IO* stammt von Industrial DOT NET, Inc[1]. Diese ist als kostenfreie Demo-Version erhältlich und gestattet die Visualisierung der von der SPS geplanten Bahn.

Die Abbildung E.1 zeigt die Festlegung von Parametern eines Bahnstützpunkts. Die Maximalgeschwindigkeit und der Interpolationstyp können hier gewählt werden. Unterschiedliche Kinematik- und Dynamikparameter können wie in Abbildung E.2 gesetzt und ihre Auswirkungen auf die Drehmoment- und Geschwindigkeitsverläufe beobachtet werden. Das erstellte Szenario mit der Bahn und den Hindernissen kann wie in Abbildung E.3 dargestellt in einer Datei gespeichert und geladen werden. Die Abbildung E.4 zeigt das Menü *Diagramme*. Es ermöglicht das Zuschalten der Geschwindigkeitsberechnung, so dass die Änderung der Bahngeschwindigkeit direkt während des Verschiebens eines Stützpunkts beobachtbar ist. Die Bahnplanung und -optimierung kann mit dem Menü *Bahnplanung* aus Abbildung E.5 Schritt für Schritt ausgeführt werden. Das nächste Menü in Abbildung E.6 gestattet die Wahl zwischen Interpolation im kartesischen Raum oder im Gelenkwinkelraum. Die letzte Abbildung E.7 zeigt die Auswahl von freigegeben Variablen der SPS-Bahnplanung. Diese werden ausgelesen und manipuliert, um Bahnen, Hindernisse und die aktuelle Position der Roboterhand zu übertragen.

[1] www.industrialdotnet.com

Abbildung E.1: Festlegung der Eigenschaften eines Bahnstützpunkts

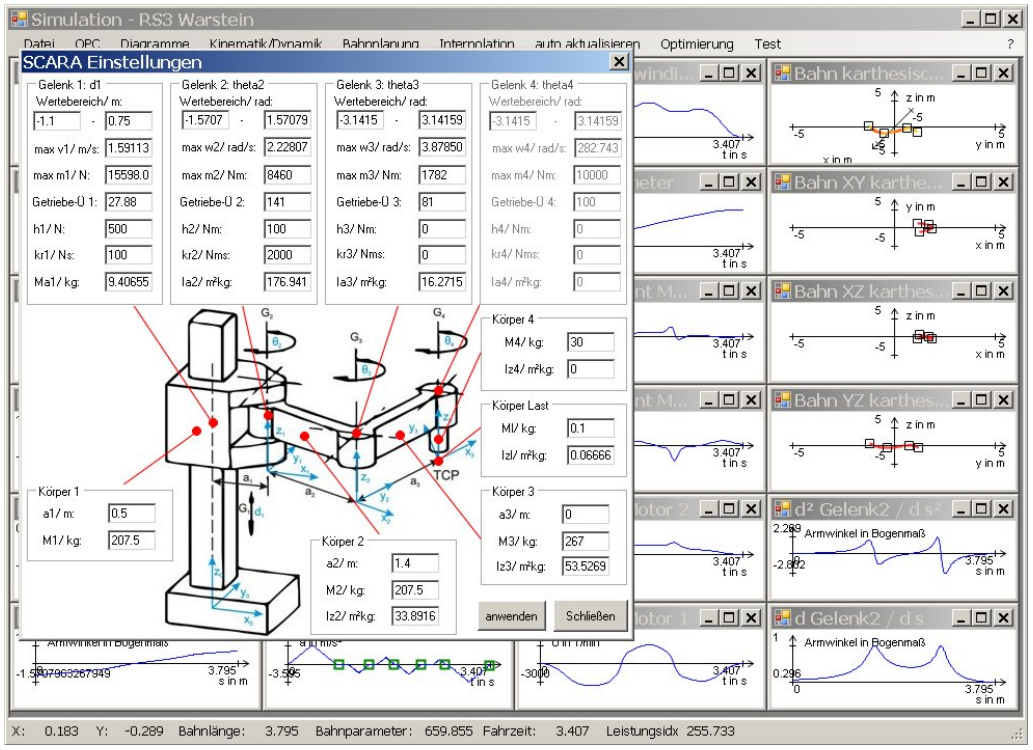


Abbildung E.2: Einstellen der Kinematik- und Dynamikparameter

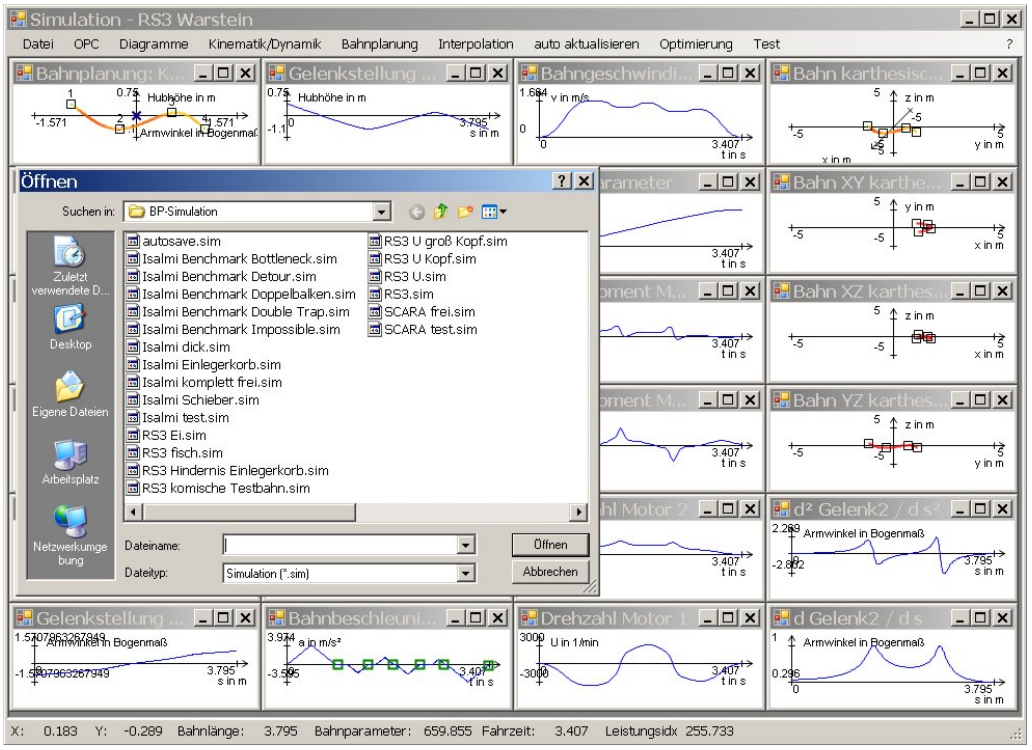


Abbildung E.3: Bahnen und Hindernisse können in Dateien abgelegt werden

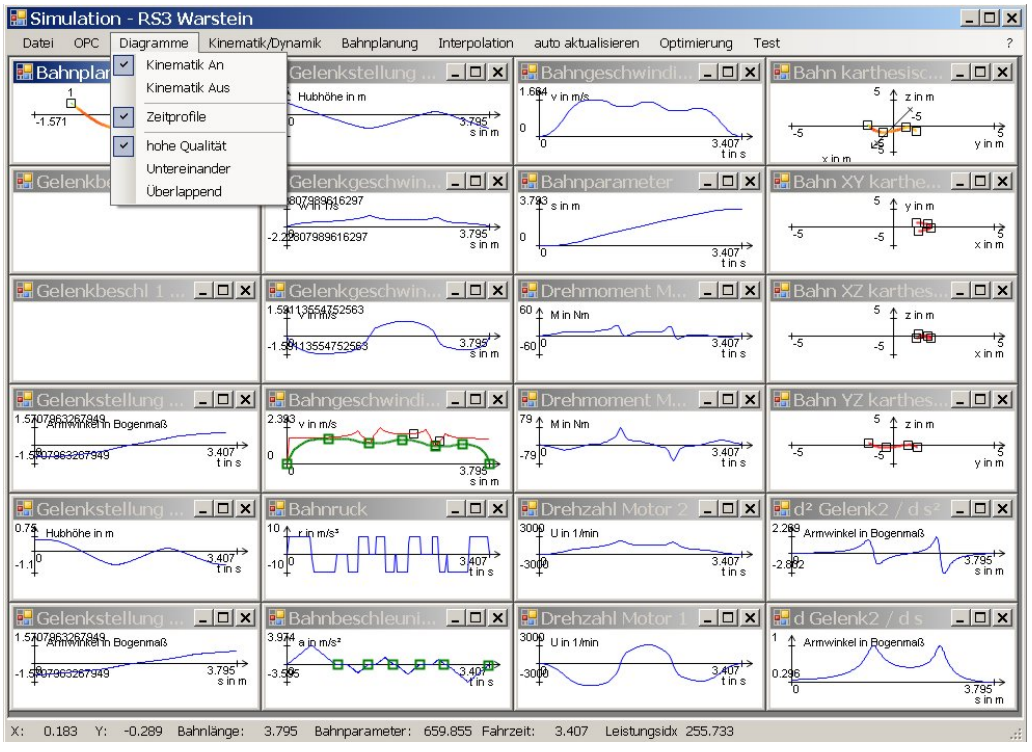


Abbildung E.4: Über das Menü *Diagramme* werden Kinematik- und Zeitberechnung hinzugeschaltet.

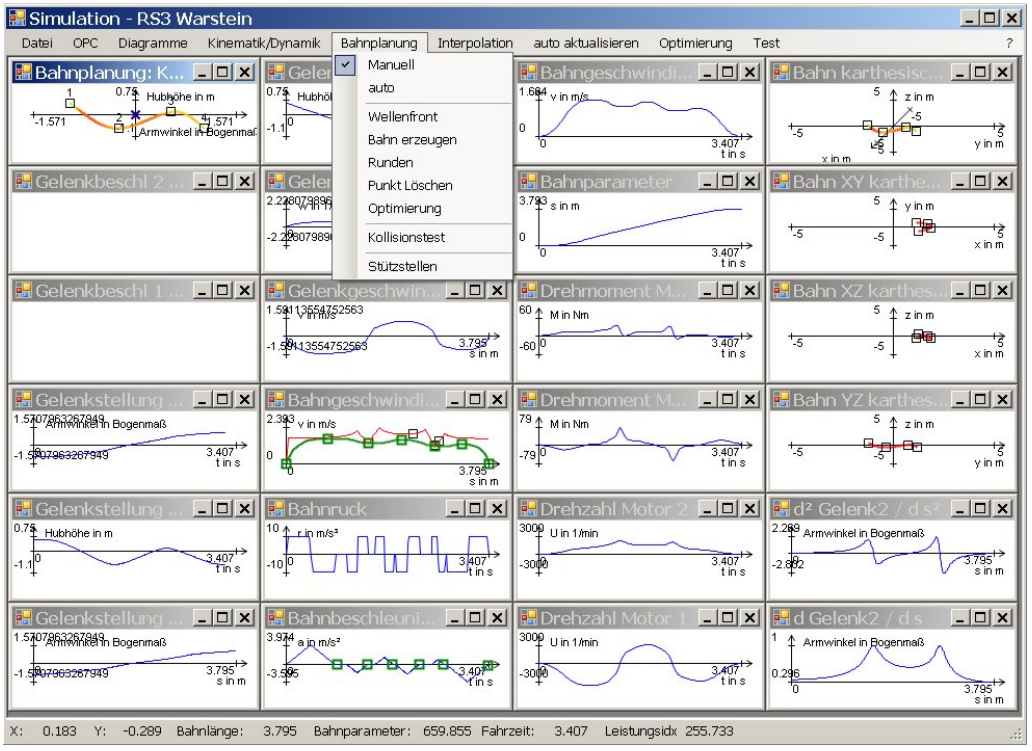


Abbildung E.5: Das Menü *Bahnplanung* bietet Zugang zur den einzelnen Schritten der Bahnplanung und -optimierung.

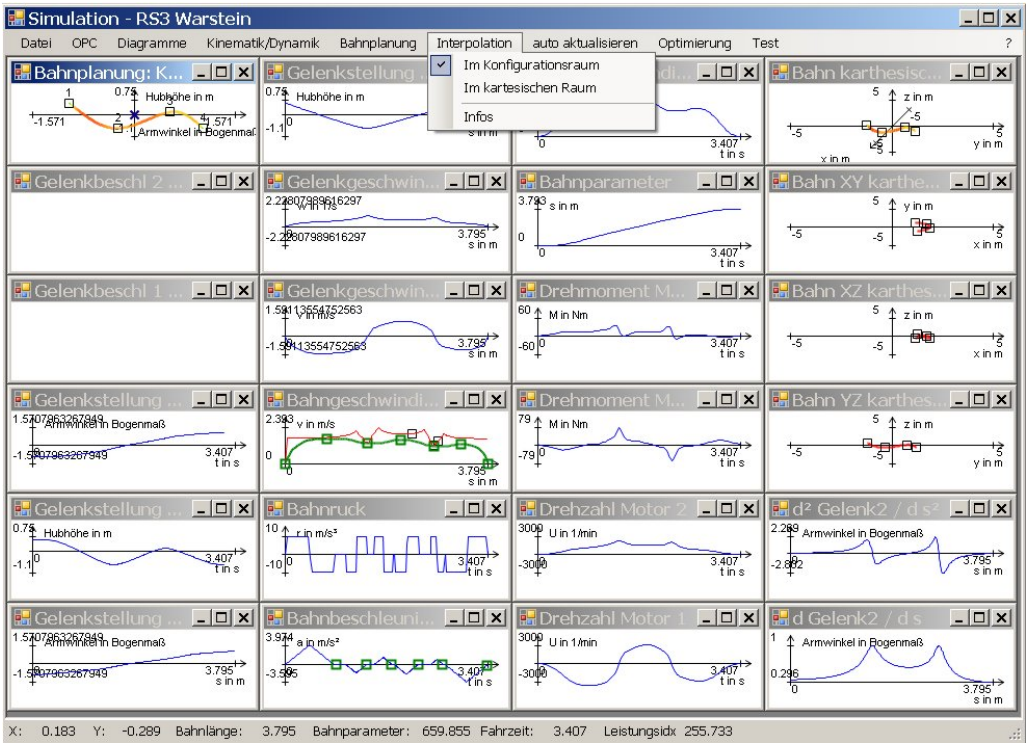


Abbildung E.6: Die Interpolation kann im Gelenkwinkelraum oder im kartesischen Raum ausgeführt werden.

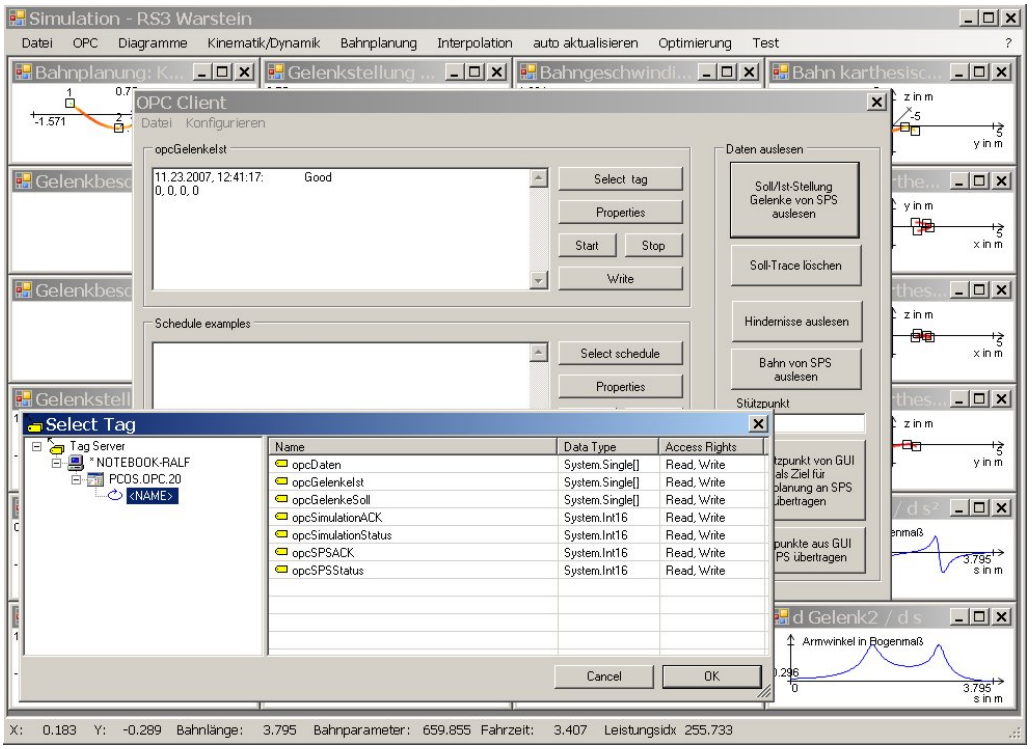


Abbildung E.7: Auswahl der OPC-Variablen zur Visualisierung der SPS-Bahnplanung