Neues Konzept zur Planung, Ausführung und Überwachung von Roboteraufgaben mit hierarchischen Petri-Netzen

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Maschinenbau der

Universität Karlsruhe

eingereichte

Dissertation

von

Dipl.-Ing. Arne Lehmann

geboren am 12. September 1974 in Berlin

Hauptreferent:	Prof. Dr.-Ing. habil. G. Bretthauer
Korreferent:	Prof. Dr.-Ing. R. Dillmann
Tag der Einreichung:	21. Dezember 2007
Tag der mündlichen Prüfung:	06. Mai 2008

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit im Institut für Angewandte Informatik des Forschungszentrums Karlsruhe.

Für die Möglichkeit an einem sehr interessanten Gebiet forschen zu können, möchte ich Professor Georg Bretthauer herzlich danken. Stets war ich mir seiner Unterstützung sicher und fand jederzeit ein offenes Ohr, um Probleme und angestrebte Lösungen konstruktiv zu diskutieren.

Weiterhin danke ich Professor Rüdiger Dillmann für die Übernahme des Korreferats. Mit Interesse hat er den Fortschritt der Arbeit verfolgt und so zu deren Gelingen beigetragen.

Ganz besonderer Dank gilt Herrn Ralf Mikut, der mir seit meinen ersten Tagen im Institut eine hervorragende fachliche Betreuung zukommen ließ.

Für die Mithilfe bei der praktischen Evaluierung des entwickelten Konzepts möchte ich mich bei Herrn Christian Regenstein und bei Herrn Giulio Milighetti bedanken.

Für wertvolle und unterhaltsame Diskussionen gilt mein Dank meinen Zimmerkollegen Herrn Sebastian Beck, Herrn Ole Burmeister und insbesondere Herrn Markus Reischl, sowie allen anderen Mitarbeitern des Instituts, die zum Gelingen dieser Arbeit beigetragen haben.

Weiterhin danke ich allen Studenten, die im Rahmen von Praktika, Studien- oder Diplomarbeiten einen Beitrag zu dieser Arbeit geleistet haben. Insbesondere möchte ich mich bei Herrn Thomas Lotz und Frau Karin Angelbauer für die fruchtbare Zusammenarbeit bedanken.

Schließlich möchte ich bei meinen Eltern, meinem Bruder sowie ganz besonders bei meiner Freundin Johanna für Rückhalt, Geduld und Verständnis bedanken. Weiterer Dank gilt Frau Doris Braun für fleißiges Korrekturlesen.

Karlsruhe, im Mai 2008 Arne Lehmann

Inhaltsverzeichnis

Inhaltsverzeichnis **iii**

Symbolverzeichnis **v**

1 Einleitung **1**
1.1 Bedeutung der Arbeit . . . 1
1.2 Darstellung des Entwicklungsstands . . . 2
1.2.1 Übersicht . . . 2
1.2.2 Steuerungsarchitekturen autonomer Robotersysteme . . . 2
1.2.3 Sicherheitsüberwachung technischer Systeme . . . 9
1.2.4 Humanoide Robotersysteme . . . 13
1.3 Offene Probleme . . . 19
1.4 Zielsetzung der Arbeit . . . 20

2 Neue Verfahren zur Planung, Ausführung und Überwachung von Roboteraufgaben **21**
2.1 Übersicht . . . 21
2.2 Steuerungsarchitektur . . . 22
2.2.1 Modellbasierte Generierung von Aufgabenwissen . . . 25
2.2.2 Petrinetzbasierte Ausführung von Roboteraufgaben . . . 31
2.2.3 Beispiel . . . 34
2.3 Überwachungskonzept . . . 37
2.3.1 Modellbasierte Fehlerdiagnose . . . 39
2.3.2 Petrinetzbasiertes Konzept zur Problemlösung . . . 48
2.3.3 Beispiel . . . 55
2.4 Diskussion . . . 57

3 Implementierung **59**
3.1 Übersicht . . . 59
3.2 Entwurf, Simulation und Analyse von Petri-Netzen . . . 60
3.3 Umsetzung der Roboter-Steuerungsarchitektur . . . 61
3.4 Portierungskonzept . . . 65
3.5 Diskussion . . . 70

4 Neuartige Generierung von Aufgabenwissen für ausgewählte Roboterkomponenten **71**
4.1 Übersicht . . . 71
4.2 Flexibler Mehrfinger-Robotergreifer . . . 71
4.2.1 Modellbildung . . . 71
4.2.2 Modellbasierte Regelung . . . 81
4.2.3 Ergebnisse . . . 88

4.2.4 Bewertung . . . 95
4.3 Anthropomorphe Halseinheit . . . 95
4.3.1 Modellbildung . . . 95
4.3.2 Modellbasierte Regelung . . . 103
4.3.3 Ergebnisse . . . 106
4.4 Diskussion . . . 111

5 Neue modellbasierte Überwachung ausgewählter Roboterkomponenten 113
5.1 Übersicht . . . 113
5.2 Modellbasierte Fehlerdetektion . . . 113
5.2.1 Erweiterte Stabilitätsüberwachung . . . 113
5.2.2 Hardwareüberwachung . . . 120
5.3 Petrinetzbasierte Problemlösung . . . 123
5.4 Prioritätsverwaltung . . . 129
5.5 Diskussion . . . 132

6 Zusammenfassung 133

A Petri-Netze 135

B Denavit-Hartenberg-Notation 137

C Modellparameter 141
C.1 Flexibler Mehrfinger-Robotergreifer . . . 141
C.2 Anthropomorphe Halseinheit . . . 142

D Programmierte Funktionen 143
D.1 Entwicklungsplattform - MATLAB . . . 143
D.2 Zielplattform - MCA . . . 145

Literaturverzeichnis 147

Bildverzeichnis 164

Tabellenverzeichnis 167

Index 168

Symbolverzeichnis

a	diskreter Zustand/Platz eines Petri-Netzes
a_j	Translationsweg entlang x_j-Achse der Halseinheit (DH-Parameter)
a_{DM0}-a_{DM3}	Parameter der Momentenkennlinie des Aktors
a_{F}, a_{N}, a_{M}	Filterkonstanten zur Berechnung von IIR-Filtern
A	Aktion
A_{Aktor}	geometrische Konstante des Aktors
$A_{(i,j)}$	Lagerkraft in den Gelenken
$\boldsymbol{A}(t)$	zeitlicher Verlauf der Platzbelegung eines Petri-Netzes
$A_{x(i,j)}$, $A_{y(i,j)}$	Komponenten der Lagerkraft in x- und y-Richtung
A_{Ac}	Platz im Verwaltungsnetz (Aktionsfolge routinemäßig abarbeiten)
A_{Ca}	Platz im Verwaltungsnetz (Aktionsfolge irreversibel abgebrochen)
A_{Fa}	Platz im Verwaltungsnetz (Aktionsfolge nicht erfolgreich durchgeführt)
A_{Re}	Platz im Verwaltungsnetz (Aktionsfolge bereit)
A_{St}	Platz im Verwaltungsnetz (Aktionsfolge zeitweilig unterbrochen)
A_{Su}	Platz im Verwaltungsnetz (Aktionsfolge erfolgreich durchgeführt)
A1	Platz im Statusnetz (Stand-By)
A2	Platz im Statusnetz Idle Task (Status Bereit)
A3	Platz im Statusnetz (Datenbankabfrage)
A4	Platz im Statusnetz (ausführungsbereite Aktionsfolge)
A5	Platz im Statusnetz (routinemäßige Abarbeitung der Aktionsfolge)
A6	Platz im Statusnetz (zeitweilige unterbrochene Aktionsfolge)
A7	Platz im Statusnetz (irreversible abgebrochene Aktionsfolge)
A8	Platz im Statusnetz (erfolgreich durchgeführte Aktionsfolge)
A9	Platz im Statusnetz (nicht erfolgreich durchgeführte Aktionsfolge)
AF	Aktionsfolge
$\mathcal{A}$	Plätze eines Petri-Netzes
$\mathbf{c}$	Vektor der Coriolis- und Zentripetalkräfte
c	Federkonstante des Aktors
$\mathbf{C}$	Vektor der verallg. Federkonstanten
d	geometrische Konstante des Aktors
d_j	Translationsweg entlang z_{j-1}-Achse der Halseinheit (DH-Parameter)
d_s	Abstandsmaß
d_{sF}	lokales Minimum des Abstandes $(M_{Objekt}, S_{(i,j)})$
d_{sP}	lokales Minimum des Abstandes $(M_{Objekt}, {}^0x_i^{tip})$
d_x	Objektverformung infolge der Kontaktsituation
d_1, d_2, d_3	Stützstellen der Zugehörigkeitsfunktionen des gewichteten Trends
D	Trend
D_{Ad}	Dämpfungsparameter der Admittanzregelung

D_{W}	gewichteter Trend
e	geometrische Konstante des Aktors
$\mathbf{e}$	Vektor der Regeldifferenzen
$e_{(i,j)}$	gewichtete Regeldifferenz des hybriden Kraft-Positionsreglers
$e_{tot_{aus}}$, $e_{tot_{ein}}$	Totzonenbereiche der Regler zur Berechnung der Stellgrößen (Greifer)
e_M	induzierte Spannung des Elektromotors der anthropomorphen Halseinheit
$\boldsymbol{E}$	Selektionsmatrix des hybriden Kraft-Positionsreglers
EA	Elementaraktion
$EA_{(i,u)Ac}$	virtuelle Unterstelle zur routinemäßigen Abarbeitung der Aktionsfolge
$EA_{(i,u)Ca}$	virtuelle Unterstelle zum irreversiblen Abbruch der Aktionsfolge
$EA_{(i,u)St}$	virtuelle Unterstelle zur zeitweiligen Unterbrechung der Aktionsfolge
$EA_{Ko,Aus1}$	Beispiel Elementaraktion zur Ausnahmebehandlung
$EA_{Ko,Aus2}$	Beispiel Elementaraktion zur Ausnahmebehandlung
$EA_{Ko,1}$	Elementaraktion 'Objekt bildbasiert verfolgen' der anthropomorphen Halseinheit
$EA_{Ko,2}$	Elementaraktion 'Objekt suchen' der anthropomorphen Halseinheit
$EA_{1..n_G,1}$	Elementaraktion 'Annähern' des Mehrfinger-Robotergreifers
$EA_{1..n_G,2}$	Elementaraktion 'Warten' des Mehrfinger-Robotergreifers
$EA_{1..n_G,3}$	Elementaraktion 'Zugreifen' des Mehrfinger-Robotergreifers
$EA_{1..n_G,4}$	Elementaraktion 'Warten' des Mehrfinger-Robotergreifers
$EA_{1..n_G,5}$	Elementaraktion 'Entfernen' des Mehrfinger-Robotergreifers
EO	Elementaroperation
E_{pos}	Summe aller positiven Regelabweichungen der Gelenke des Mehrfinger-Robotergreifers
E_{neg}	Summe aller negativen Regelabweichungen der Gelenke des Mehrfinger-Robotergreifers
E_{kin}	kinetische Energie
f	Brennweite der Kamera
$\mathbf{f}$	Vektor der Reibungskräfte
$\boldsymbol{f}_a^R$	Allg. Vektorfunktion zur Beschreibung der Regelung
f_a^{Tg}	Allg. Funktion zur Beschreibung der Trajektoriengenerierung
F	externe Kraft
$\boldsymbol{F}$	Vektor der Fehlerzustände (Aktorik und Sensorik)
F_A	fluidische Kraft resultierend aus Innendruck im Aktor
F_K	resultierende Kontaktkraft
F_{Kx}, F_{Ky}	Komponenten der resultierenden Kontaktkraft F_K in x- und y-Richtung
$F_{Schwell}$	Schwellwert zur Kontakterkennung
$\mathcal{F}$	Kanten eines Petri-Netzes
$\boldsymbol{g}$	rückgeführte Vektorfunktion zur modellbasierten Fehlerdiagnose
$g_{(i,j)}$	Gerade durch Fingerglied (i, j)
$g_{W(i,j)}$	Wichtungsfunktion des hybriden Kraft-Positionsreglers
g_a^R	Allg. definierte Funktion zur Beschreibung der Regelung
$\boldsymbol{G}$	Wichtungsmatrix des hybriden Kraft-Positionsreglers
h	Höhe des zu greifenden Objekts
H	Handlung
$\boldsymbol{H}$	Rückkoppelmatrix zur modellbasierten Fehlerdiagnose
i	Laufvariable der Roboter-Funktionaleinheiten
i_A	Ankerstrom eines Elektromotors der anthropomorphen Halseinheit
$\hat{i}_G$	Übersetzungsfaktor des Harmonic-Drive Getriebes
i_Z	Übersetzungsfaktor eines Zahnriementriebes der anthropomorphen Halseinheit
$\boldsymbol{I}$	Einheitsmatrix des hybriden Kraft-Positionsreglers

IIR	Infinite-Input-Response
j	Laufvariable der Anzahl unabhängiger Freiheitsgrade einer Roboter-Funktionaleinheit i
J_D	Massenträgheitsmoment parallele Schwerpunktsachse
J_{GA}	Massenträgheitsmoment am Getriebeausgang
J_{GE}	Massenträgheitsmoment am Getriebeeingang
J_L	Last-Massenträgheitsmoment des Roboterkopfes
J_{LS}	Massenträgheitsmoment lastseitig
J_M	Rotorträgheitsmoment
J_{Red}	reduziertes Massenträgheitsmoment der anthropomorphen Halseinheit
J_S	Massenträgheitsmoment um Schwerpunktsachse
k	Abtastzeitpunkt
k_M	Drehmomentkonstante des Elektromotors der anthropomorphen Halseinheit
k_O	Federkonstante des Objekts
k_{switch}	Schaltfaktor zur Prioritätenumschaltung des ereignisdiskreten Regelungsalgorithmus
k_u	Auflösung der Kamera in u-Richtung
k_v	Auflösung der Kamera in v-Richtung
K_{Ad}	Steifigkeitsparameter der Admittanzregelung
K_{BP}	Reglerverstärkung des zeitdiskreten P-Algorithmus (bildbasierte Regelung der Halseinheit)
$K_{(i,j)}$	Kontaktsituation des Mehrfinger-Robotergreifers
$\mathbf{K}_K$	Matrix der internen Kameraparameter
K_r	Reglerverstärkung des zeitdiskreten PID-Algorithmus (Positionsregelung Halseinheit)
K_R	Reglerverstärkung des zeitdiskreten PI-Algorithmus (Greifer)
K_{RF}	Reglerverstärkung des zeitdiskreten PI-Algorithmus im positionsgeregelten Modus
K_{RP}	Reglerverstärkung des zeitdiskreten PI-Algorithmus im momentengeregelten Modus
K_V	geometrische Ventilkonstante
$\mathcal{K}$	Kapazitäten eines Petri-Netzes
$l_{(i,j)}$	Hebelarm zur Berechnung der verallg. Gelenkmomente
l_{MG}	Abstand des Gelenkmittelpunkts $M_{F(i,j)}$ von den Koordinaten der Fingerspitze ${}^0x_i^{tip}$
l_{MS}	Abstand des Gelenkmittelpunkts $M_{F(i,j)}$ vom Schnittpunkt $S_{(i,j)}$
$l_{x(i,j)}$, $l_{y(i,j)}$	Komponenten des Hebelarms in x- und y-Richtung
l_1, l_2	Stützstellen der Zugehörigkeitsfunktionen der mittleren Bewertungsfunktion
L	Bewertungsfunktion
L_{EM}	Anschlussinduktivität eines Elektromotors der anthropomorphen Halseinheit
L_{Extern}	Korrektur der Bewertungsfunktion
L_{M}, L_{N}	korrigierte und gefilterte Bewertungsfunktionen
L_{Mittl}	mittlere Bewertungsfunktion
m	Objektmasse
$\mathbf{m}$	Abbildung eines Punktes $\mathbf{X}_0$ auf der Kamerabildebene
m_A	Fluidmasse im Aktor
$\dot{m}_A$	Fluidmassenstrom in den Aktor
m_i	Maximale Anzahl der Freiheitsgrade einer Roboter-Funktionaleinheit i
$\mathbf{m}_{Soll}$	Sollposition eines Punktes $\mathbf{X}_0$ auf der Kamerabildebene
M	resultierendes Antriebsmoment eines Elektromotors der anthropomorphen Halseinheit
$\boldsymbol{M}$	Regelgrößenvektor der Gelenkmomente für den Mehrfinger-Robotergreifer
$\mathbf{M}_{\text{A}}$	Vektor der Aktormomente
M_{EM}	Motormoment des Elektromotors der anthropomorphen Halseinheit
$M_{Fehler(i,j)}$	Regelabweichung des hybriden Kraft-Positionsreglers (Gelenkmoment)
$M_{F(i,j)}$	Gelenkmittelpunkt

m_K	Kopfmasse
MKQ	Methode der Kleinsten Quadrate
M_L	Lastmoment des Elektromotors der anthropomorphen Halseinheit
$M_{L,Red}$	reduziertes Lastmoment der anthropomorphen Halseinheit
M_{Objekt}	Mittelpunkt des zu greifenden Objekts
$\boldsymbol{M_{Soll}}$	Führungsgrößenvektor der Gelenkmomente für den Mehrfinger-Robotergreifer
$\boldsymbol{M_z}$	externer Störmomentenvektor
$\dot{m}_P$	Fördermassenstrom der Pumpe
$\mathbf{M}(\mathbf{q})$	Trägheitsmatrix
m_x	Anzahl der inneren Systemzustände einer Roboter-Funktionaleinheit i
M_0	durch elastische Dehnung erzeugtes Drehmoment
M_1	durch Druck auf die Stirnflächen erzeugtes Drehmoment
M_2	Rückstellmoment
$\mathcal{M}$	Markenbelegung eines Petri-Netzes
$\mathcal{M}_0$	initiale Markenbelegung eines Petri-Netzes
n	Maximale Anzahl von Roboter-Funktionaleinheiten in einem Robotersystem
n_F	Anzahl an Fingern eines Robotergreifers
n_G	an der Ausführung eines Griffs beteiligte Finger
n_y	Anzahl an Regelgrößen in einem Mehrgrößensystem
n_1	Finger mit vorzeitigem Objektkontakt
n_2	Finger mit vorzeitigem Verlust des Objektkontakts
N	linguistischer Term NULL
NEG	linguistischer Term NEGATIV GROSS
O_B	Ursprung des Objektkoordinatensystem
$O_{(i,j)}$	Fingerkoordinatensysteme
O_K	Ursprung des Kamerakoordinatensystems
OKS	Objektkoordinatensystem
O_0	Ursprung des Weltkoordinatensystems
p_A	Druck innerhalb eines Aktors
$p_{A,max}$	maximaler Druck innerhalb eines Fluidaktors
p_{Kpr}	vom Kompressor erzeugter externer Druck
$p_{n_y,1} \ldots p_{n_y,m_i}$	Vektor der Wichtungselemente einer Regelgröße
p_0	Umgebungsdruck
P	Leistung
$\mathbf{P}$	positiv definite Matrix
PG	linguistischer Term POSITIV KLEIN
PK	linguistischer Term POSITIV GROSS
p_{VB}	Ventilbankdruck
PWM	Pulsweitenmodulierung
$\boldsymbol{P}_1 \ldots \boldsymbol{P}_{n_y}$	Matrizen der Wichtungselemente des Roboter-Teilsystems
$\mathbf{q}$	Vektor der verallg. Gelenkwinkel
$\dot{\boldsymbol{q}}$	Vektor der verallg. Gelenkwinkelgeschwindigkeiten
$\ddot{\boldsymbol{q}}$	Vektor der verallg. Gelenkwinkelbeschleunigungen
$\mathbf{q}_1$	Vektor der Antriebspositionen
$\mathbf{q}_2$	Vektor der Fingergliedpositionen
r	Radius des Objekts
$\boldsymbol{r}$	Vektor der modellbasiert geschätzten Störgrößen
R	Anschlusswiderstand des Elektromotors der anthropomorphen Halseinheit

$\mathbf{R}$	Rotation zwischen Welt- und Kamerakoordinatensystem (in homogenen Koordinaten)
S	Stabilitätsgrad zur online Bewertung des Systemverhaltens
$S_{(i,j)}$	Position mit minimalen Abstand Objektmittelpunkt M_{Objekt} Gerade $g_{(i,j)}$
S_{Ist}	Istzustand
S_{Soll}	Sollzustand
t	Zeit
$\mathbf{t}$	Translation zwischen Welt- und Kamerakoordinatensystem (in homogenen Koordinaten)
t_a	Abtastzeit
$t_{Ai,max}$	Maximalzeit einer Marke auf einer Stelle
$\mathcal{T}$	Transitionen eines Petri-Netzes
T_n	Nachstellzeit des zeitdiskreten PID-Algorithmus (Positionsregelung Halseinheit)
T_N	Zeitkonstante des zeitdiskreten PI-Algorithmus (Greifer)
T_{NF}	Zeitkonstante des zeitdiskreten PI-Algorithmus im positionsgeregelten Modus
T_{NP}	Zeitkonstante des zeitdiskreten PI-Algorithmus im momentengeregelten Modus
$\boldsymbol{T}(t)$	zeitlicher Verlauf einer geschalteten Transition eines Petri-Netzes
T_v	Vorhaltezeit des zeitdiskreten PID-Algorithmus (Positionsregelung Halseinheit)
${}^0T_{(i,j)}$	homogene Koordinatentransformation
${}^0\mathbf{T}_K$	Matrix der externen Kameraparameter
$T_{Ko,12}$	Transition 'Objekt nicht lokalisiert'
$T_{Ko,21}$	Transition 'Objekt lokalisiert'
$T_{1..n_G,12}$	Transition 'Objektkontakt einzelner Finger'
$T_{1..n_G,23}$	Transition 'Objektkontakt alle Finger'
$T_{1..n_G,34}$	Transition 'Objektkontakt Verlust einzelner Finger'
$T_{1..n_G,45}$	Transition 'Objektkontakt Verlust alle Finger'
$T_{Ko,1Aus1}$	Beispiel Transition zur Ausnahmebehandlung
$T_{Ko,Aus11}$	Beispiel Transition zur Ausnahmebehandlung
$T_{Ko,2Aus1}$	Beispiel Transition zur Ausnahmebehandlung
$T_{Ko,Aus12}$	Beispiel Transition zur Ausnahmebehandlung
$T_{Ko,1Aus2}$	Beispiel Transition zur Ausnahmebehandlung
$T_{Ko,Aus21}$	Beispiel Transition zur Ausnahmebehandlung
$T_{Ko,2Aus2}$	Beispiel Transition zur Ausnahmebehandlung
$T_{Ko,Aus22}$	Beispiel Transition zur Ausnahmebehandlung
T_{Ac}	Transition im Verwaltungsnetz (routinemäßig abarbeiten)
T_{Be}	Transition im Verwaltungsnetz (Aktionsfolge starten)
T_{Ca}	Transition im Verwaltungsnetz (Abarbeitung irreversibel abbrechen)
T_{Fa}	Transition im Verwaltungsnetz (Abarbeitung nicht erfolgreich durchgeführt)
T_{Su}	Transition im Verwaltungsnetz (Abarbeitung erfolgreich durchgeführt)
T_{St}	Transition im Verwaltungsnetz (Abarbeitung unterbrechen)
T1	Transition im Statusnetz (Power On aktiviert)
T2	Transition im Statusnetz (Stand-By aktiviert)
T3	Transition im Statusnetz (Aufgabe kommandiert)
T4	Transition im Statusnetz (Aufgabe ist bekannt und ausführbar)
T5	Transition im Statusnetz (Aufgabe ist nicht bekannt und/oder nicht ausführbar)
T6	Transition im Statusnetz (Abarbeitung der Aktionsfolge gestartet)
T7	Transition im Statusnetz (Abarbeitung der Aktionsfolge unterbrochen)
T8	Transition im Statusnetz (Abarbeitung der Aktionsfolge routinemäßig fortgesetzt)
T9-T11	Transitionen im Statusnetz (Abarbeitung der Aktionsfolge irreversibel abgebrochen)
T12	Transition im Statusnetz (Abarbeitung der Aktionsfolge erfolgreich durchgeführt)

T13	Transition im Statusnetz (Abarbeitung der Aktionsfolge nicht erfolgreich durchgeführt)
$\mathbb{T}$	Roboteraufgabe (engl.: task)
u	u-Koordinate eines Punktes im Bildebenenkoordinatensystem
$\boldsymbol{u}$	Vektor der Stellgrößen
u_A	Ankerspannung des Elektromotors der anthropomorphen Halseinheit
u_I	I-Anteil des zeitdiskreten PI-Algorithmus
$u_{(i,j)}$	Stellgrößentrajektorie für einen Freiheitsgrad j einer Roboter-Funktionaleinheit i
u_{Soll}	Sollposition der u-Koordinate eines Punktes im Bildebenenkoordinatensystem
$u_{tot_{aus}}$, $u_{tot_{ein}}$	Totzonenbereiche der u-Koordinate der bildbasierten Regelung der Halseinheit
u_0	u-Koordinate des Schnittpunkts zwischen der optischen Achse z_K und der Bildebene
v	v-Koordinate eines Punktes im Bildebenenkoordinatensystem
$\boldsymbol{v}_a$	Parametervektor zur Definition von Führungsgrößentrajektorien und Regelungsalgorithmen
v_{Soll}	Sollposition der v-Koordinate eines Punktes im Bildebenenkoordinatensystem
$v_{tot_{aus}}$, $v_{tot_{ein}}$	Totzonenbereiche der v-Koordinate der bildbasierten Regelung der Halseinheit
v_0	v-Koordinate des Schnittpunkts zwischen der optischen Achse z_K und der Bildebene
V_A	Fluidvolumen in einen Aktor
V_{VB}	Speichervolumen der Ventilbank
$\boldsymbol{w}$	Vektor der Führungsgrößen
$w_{(i,j)}$	Führungsgrößentrajektorie für einen Freiheitsgrad j einer Roboter-Funktionaleinheit i
$\mathbf{w}_{\text{Max}}$, $\mathbf{w}_{\text{Min}}$	Vektoren der Führungsgrößenintervalle
$\mathcal{W}$	Kantengewichte eines Petri-Netzes
WKS	Weltkoordinatensystem
${}^0x_{(i,j)}$	Gelenkpunkt
${}^0x_i^{tip}$	kartesische Koordinaten der Fingerspitze
x_P	Steuerspannung der Pumpe (PWM) des Mehrfinger-Robotergreifers
$\boldsymbol{x}_{(i,j)}^R$	Zustandsvektor der Regelungskomponente einer Roboter-Funktionaleinheit i
$\boldsymbol{x}_i^{Str}$	Zustandsvektor einer Roboter-Funktionaleinheit i
$\mathbf{x_V}$	Vektor der Steuerspannungen der Ventile (binär) des Mehrfinger-Robotergreifers
x	x-Koordinate eines Punktes im Weltkoordinatensystem
$\mathbf{X}_0$	Punkt im Weltkoordinatensystem (in homogenen Koordinaten)
$\boldsymbol{y}$	Vektor der Regelgrößen
$\boldsymbol{y}_i$	Vektor der Regelgrößen einer Roboter-Funktionaleinheit i
y	y-Koordinate eines Punktes im Weltkoordinatensystem
$\boldsymbol{z}$	Vektor der externen Störgrößen
z^{-1}	Verzögerung um einen Abtastzeitpunkt ta
$\hat{\boldsymbol{z}}$	modellbasiert geschätzte Störgröße
$\boldsymbol{z}_i$	Vektor der externen Störgrößen einer Roboter-Funktionaleinheit i
z_K	optische Achse
z	z-Koordinate eines Punktes im Weltkoordinatensystem
α	Wichtungsparameter des kontinuierlich ereignisdiskreten Regelungsalgorithmus
β	Wichtungsparameter des kontinuierlich ereignisdiskreten Regelungsalgorithmus
γ	Orthogonalitätsfaktor der Bildachsen (u, v)
$\Delta\dot{m}$	Massenstrom zum Aktor (pneumatischer Betrieb)
$\Delta M_{(i,j)}$	Differenz der Gelenkmomente
Δp_{VB}	Druckdifferenz an der Pumpe
Δp	Druckdifferenz an einem Ventil
$\Delta\mathbf{w}[k]$	Vektor der Änderungen der Führungsgrößen
$\Delta\mathbf{z}[k]$	Vektor der Änderungen der Störgrößen

$\Delta\varphi_{(i,j)}$	Differenz der Gelenkwinkel
$\boldsymbol{\tau}_{\text{Antrieb}}$	Vektor der verallg. Antriebsmomente
$\boldsymbol{\tau}_{\text{G}}$	Vektor der Gelenkfedermomente
$\boldsymbol{\varphi_{Soll}}$	Führungsgrößenvektor der Gelenkwinkel für den Mehrfinger-Robotergreifer
$\boldsymbol{\varphi}$	Regelgrößenvektor der Gelenkwinkel für den Mehrfinger-Robotergreifer
$\varphi_{Ad(i,j)}$	Führungsgröße der Admittanzregelung
$\varphi_{Fehler(i,j)}$	Regelabweichung des hybriden Kraft-Positionsreglers (Gelenkwinkel)
φ_1	Gelenkwinkel des Kopfdrehens
φ_2	Gelenkwinkel des unteren Nickens
φ_3	Gelenkwinkel des Neigens
φ_4	Gelenkwinkel des oberen Nickens
φ_M	Achsposition des Elektromotors der anthropomorphen Halseinheit
φ_{GE}	Achsposition des Getriebeeingangs
φ_{GA}	Achsposition des Getriebeausgangs
φ_L	Achsposition der Last der anthropomorphen Halseinheit
$\varphi_{Soll(Max(j))}$	Maximale Sollpositionen der Freiheitsgrade der Halseinheit
$\varphi_{Soll(Min(j))}$	Minimale Sollpositionen der Freiheitsgrade der Halseinheit
ω_M	Winkelgeschwindigkeit des Elektromotors der anthropomorphen Halseinheit
$\dot{\omega}_M$	Winkelbeschleunigung des Elektromotors der anthropomorphen Halseinheit

1 Einleitung

1.1 Bedeutung der Arbeit

Als humanoider Roboter wird in dieser Arbeit ein Robotersystem mit möglichst menschenähnlicher Gestalt, menschenähnlichem Verhalten und menschenähnlicher Intelligenz bezeichnet [76]. Der Entwicklung solcher humanoider Roboter liegen historisch bedingt zwei Hauptmotive zugrunde.

Zunächst bestand das Entwicklungsziel in der Konstruktion einer multifunktionalen Arbeitsmaschine. Besonders bei alltäglichen Gebrauchsgegenständen wie Werkzeugen und Geräten, aber auch bei Gebäuden und Verkehrsmitteln wird deutlich, wie stark sich der Lebensraum des Menschen an der menschlichen Physiologie orientiert. Ein durch seine anthropomorphe Gestalt an den Lebensraum des Menschen angepasster Roboter kann dem Menschen in seinem Umfeld als vielseitiger Helfer zur Seite stehen und damit eine Vielzahl von Spezial-Maschinen ersetzen. Ideen für solche 'künstlichen' Helfer gehen bis in das Altertum zurück [233].

Anfang der 50'er Jahre des zwanzigsten Jahrhunderts begannen Forschergruppen sich mit der Erschaffung von künstlicher Intelligenz zu beschäftigen [274]. Diese 'künstliche Intelligenz' sollte in der Lage sein, aktiv am sozialen Leben des Menschen teilzunehmen und durch Beobachtung, Interaktion und Kommunikation zu lernen. Heutzutage ist als Grundlage für die Erschaffung einer menschenähnlichen, künstlichen Intelligenz die Konstruktion eines multifunktionalen Roboters anzusehen. Um seine Akzeptanz unter den Menschen zu erhöhen, soll ein solcher Roboter zudem über eine menschliche Gestalt, Mobilität und menschenähnliche Sensorik verfügen.

Eine der wichtigsten Anforderungen an ein modernes humanoides Robotersystem ist demnach die Fähigkeit, Alltagsaufgaben kooperativ und multimodal interagierend mit dem Menschen zu bewältigen [76]. Diese Forderung geht deutlich über den heutigen Stand der Robotertechnik hinaus [175]. So erfolgt bei Industrierobotern aus sicherheitstechnischen Aspekten derzeit noch eine strikte Trennung zwischen den Arbeitsräumen von Roboter und Mensch [5, 73]. Bei der Anwendung in der Produktion, mit bekanntem Umfeld und vorgegebenen sich wiederholenden Aufgaben, ergeben sich somit weitaus geringere Anforderungen an ein Überwachungssystem als bei humanoiden Robotern, die sich in veränderlichen Umgebungen mit unbekannten Situationen auseinanderzusetzen haben. So ist eine vollständige Offlineplanung der auszuführenden Aufgaben für humanoide Robotersysteme auszuschließen. Dies erfordert ein neues Überwachungskonzept, das in der Lage ist, Gefahrensituationen schnell und flexibel zu erkennen, um gegebenenfalls geeignete Gegenmaßnahmen einzuleiten. Solche Maßnahmen können dabei, neben dem Abbruch der geplanten Aufgabe ('Notaus'), durchaus komplexe Problemlösungsstrategien wie z. B. eine autonome Neuplanung der auszuführenden Aufgaben oder eine Umschaltung in den Benutzerdialog- oder Telepräsenzmodus sein.

Das in dieser Arbeit erstmals vorgestellte, modulare Konzept stellt einen integrierten Lösungsvorschlag zur flexiblen Sicherheitsüberwachung von humanoiden Robotersystemen dar.

1.2 Darstellung des Entwicklungsstands

1.2.1 Übersicht

Aus der Betrachtungsweise in Abschnitt 1.1 lassen sich drei Schwerpunkte für die Darstellung des Entwicklungsstands ableiten:

1. Stand der Entwicklung bei kognitiven Steuerungsarchitekturen, da sie die Grundlage zur Realisierung und Evaluierung von Arbeiten zur Perzeption, Interaktion, Kommunikation (kognitiven Fähigkeiten) und bewegungsmotorischen Steuerung (motorische Fähigkeiten) sowie zur Sicherheitsüberwachung auf humanoiden Robotersystemen bilden (Abschnitt 1.2.2).
2. Stand der Entwicklung bei der Sicherheitsüberwachung von technischen Systemen. Dies beinhaltet allgemein anwendbare Verfahren aus allen technischen Bereichen sowie existierende Sicherheits- und Überwachungskonzepte im Bereich der Robotik (Abschnitt 1.2.3).
3. Überblick über die derzeit fortschrittlichsten humanoiden Robotersysteme aus Industrie und Forschung (Abschnitt 1.2.4).

1.2.2 Steuerungsarchitekturen autonomer Robotersysteme

Allgemein besteht die Aufgabe eines autonomen Robotersystems in der benutzer- und/oder umgebungsabhängigen Umsetzung von abstrakt formulierten Aufgaben in ausführbare Handlungen. In der Literatur werden dabei die unterschiedlichen Möglichkeiten zur Verknüpfung von Sensordaten und gestellter Aufgabe mit einer Handlung des Roboters als Steuerungsarchitekturen bezeichnet.

Dabei entsteht durch die Umsetzung einer abstrakten Aufgabe in eine durch das Robotersystem ausführbare Sequenz von Handlungen, unabhängig von der gewählten Steuerungsarchitektur, ein regelungstechnisches Problem. So hat die Trajektoriengenerierung jeder Steuerungsarchitektur die Aufgabe, die generierte Sequenz von Handlungsabläufen in eine zulässige geometrische Bahn im Konfigurationsraum des Robotersystems umzusetzen. Dabei muss das Robotersystem von einer Anfangs- (Istzustand) in eine Zielstellung (Sollzustand) gebracht werden. Probleme können dabei aus physikalischen (z. B. Gelenkanschläge) und geometrischen Restriktionen (z. B. Hindernisse im Konfigurationsraum) des Systems, die die Zulässigkeit einer geplanten geometrischen Bahn definieren, entstehen. Es existieren zahlreiche verschiedene Methoden zur Trajektoriengenerierung mit [124, 169, 240] und ohne [19, 254] Kollisionsvermeidung. Eine weitere Gruppe sind Verfahren, die auf der Aufzeichnung und Analyse der Bewegung eines Objektes oder einer Person im Raum basieren [178, 238, 285]. Solche Bewegungserfassungssysteme werden in der Robotik z. B. beim Programmieren durch Vormachen und Lernen durch Imitation verwendet. Bei beiden Vorgehensweisen werden Bewegungen von dem Bewegungserfassungssystem aufgezeichnet und daraus Roboterprogramme abgeleitet [22, 79, 83]. Die Grundidee dieser Verfahren ist die Abbildung menschenähnlicher Bewegungen auf das Robotersystem. Dazu werden die von dem Bewegungserfassungssystem aufgezeichneten Bewegungsabläufe skaliert und auf das kinematische Modell des Robotersystems übertragen. Dieses Vorgehen setzt allerdings ”vergleichbare” Kinematiken von Robotersystem und Mensch voraus. Beispiele hierzu sind in [45, 275, 276] zu finden.

Die Regelungskomponente der Steuerungsarchitektur setzt die berechnete geometrische Bahn in eine Bewegung des Robotersystems um. Bei Roboterregelungen handelt es sich im Allgemeinen um nichtlineare, strukturvariable, diskret-kontinuierliche Mehrgrößenregelungen (z. B. Positionen und Kräfte) [170, 239]. Das Regelungsziel besteht darin, ein solches hybrides System mit mehreren Freiheitsgraden einem geeignet spezifizierten Referenzverhalten möglichst gut nachzuführen und zusätzlich externe

Störeinflüsse zu kompensieren. Störungen können dabei sowohl zu Abweichungen in den kontinuierlichen Prozessgrößen als auch zu fehlerhaften diskreten Systemzuständen führen. Die Regelungsalgorithmen eines Roboters müssen deshalb durch eine geeignete Strukturumschaltung an geänderte diskrete Systemzustände angepasst werden. So ist beispielsweise bei freier Bewegung im Raum ein gewünschtes Bewegungsverhalten zu gewährleisten. Bei Interaktion des Roboters mit der Umwelt sind dagegen geforderte Kontaktkräfte einzuregeln. Weiterhin können Roboterregelungen nach der Vorgabe der Führungsgröße unterteilt werden. So wird bei Regelungskonzepten im Gelenkwinkelraum zunächst die Transformation der gewünschten Bewegung vom Konfigurationsraum des Roboters, d. h. von kartesischen Weltkoordinaten, in den Gelenkwinkelraum mit Hilfe der Berechnung der inversen Kinematik durchgeführt. Eingangsgrößen in den Regelkreis sind bei diesen Konzepten Sollgrößen im Gelenkwinkelraum. Nachteilig ist dabei, dass keine direkte Beeinflussung der Führungsgrößen im kartesischen Weltkoordinatensystem möglich ist. Bei Regelungskonzepten im kartesischen Weltkoordinatensystem entfällt die explizite Berechnung der inversen Kinematik. Eingangsgrößen in den Regelkreis sind die Sollgrößen in kartesischen Koordinaten. Dem nominellen Vorteil dieser Konzepte, der direkten Beeinflussung der aufgabenrelevanten Führungsgrößen im Weltkoordinatensystem, steht in den meisten realen Anwendungen jedoch der Nachteil der nicht direkt messbaren Regelgrößen im Weltkoordinatensystem entgegen. Die Umrechnung der Regelgrößen in Weltkoordinaten erfolgt dann mit Hilfe der direkten Kinematik aus den gemessenen Regelgrößen im Gelenkwinkelraum. Bild 1.1 veranschaulicht den Zusammenhang zwischen Transformationen und Bezugsräumen.

Bild 1.1: Direktes und inverses kinematisches Problem

Es wird bislang zwischen vier verschiedenen Kategorien von Steuerungsarchitekturen mit unterschiedlichen Schwerpunkten unterschieden [149], die im Weiteren kurz vorgestellt werden:

- Deliberative, hierarchische planungsdominierte Steuerungsarchitekturen [85, 133],
- Reaktive reflexbasierte Steuerungsarchitekturen [58, 59, 181],
- Verhaltensbasierte Steuerungsarchitekturen [132, 232] und
- Hybride Steuerungsarchitekturen [21, 55, 100, 256].

1.2.2.1 Deliberative Steuerungsarchitekturen (’Sense-Plan-Act’)

Deliberative, planungsdominierte Ansätze stützen sich auf eine aktionsunabhängige symbolische Repräsentation der Umgebung. Dieses Weltmodell bildet die Basis für die Integration von Sensorinformationen (’Sense’) und die Generierung (’Plan’) von auszuführenden Handlungen (’Act’). Die Aufgabe der Planungskomponente besteht dabei darin, aus den vorgegebenen Zielen und einer Menge von möglichen Handlungen die bestmögliche Ausführungsreihenfolge zu finden, um das Weltmodell in den gewünschten Zielzustand zu überführen.

Falls das Weltmodell hinreichend genaue Informationen enthält und dem Roboter ausreichend Zeit zur Planung zur Verfügung steht, ist ein deliberativer Ansatz eine effektive Methode, um eine vorgegebene komplexe Aufgabe erfolgreich abzuarbeiten. Allerdings sind bei humanoiden Robotern, die sich in der

realen, ständig ändernden Welt bewegen, Planungsprozesse häufig sehr rechenintensiv. Dadurch ist mit einer deliberativen Steuerungsarchitektur keine schnelle Reaktion auf dynamisch veränderliche Umweltzustände möglich. Weitere Fehlerursachen, die zu einem unerwünschten Roboterfehlverhalten führen können, sind verrauschte Sensorwerte und eine unvollständige, ungenaue bzw. fehlerhafte Weltmodellierung. Bild 1.2 zeigt die schematische Darstellung einer sequentiellen, deliberativen Steuerungsarchitektur.

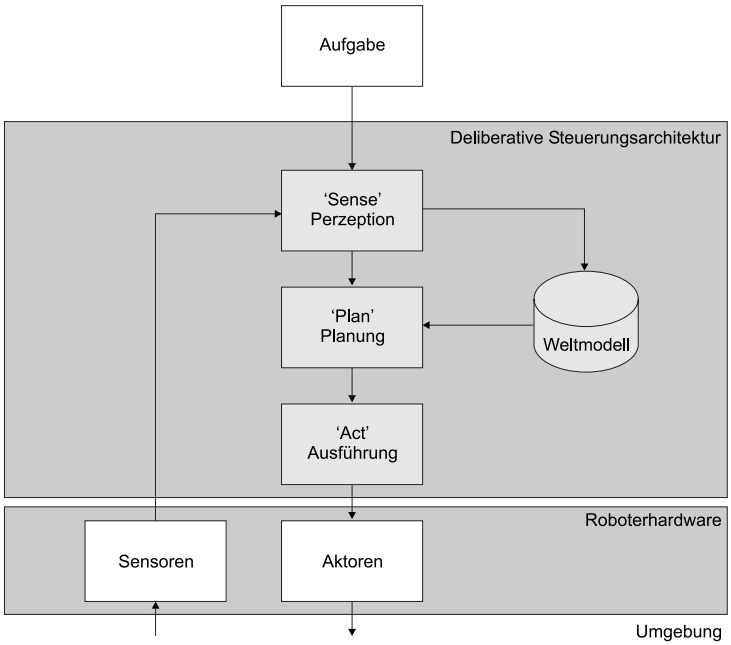


Bild 1.2: Schematische Darstellung einer deliberativen Steuerungsarchitektur nach [149] (modifiziert)

Es haben sich verschiedene Strategien zur autonomen Planung von Handlungen entwickelt. Eine Möglichkeit besteht in der Verfeinerung von Planungsschritten, dem Refinement. Dabei werden einem groben Planskelett weitere Handlungen beigefügt, bis der so entstandene Plan dem Ziel genügt. Demgegenüber werden bei der so genannten Retraction schon existierende Planschritte gelöscht, um das vorgegebene Ziel zu erreichen. Eine Kombination der bereits genannten Ansätze ist der transformationelle Planer. So basiert der transformationelle XFRM-Planer auf dem Prinzip der Planrevision. Für vorgegebene Aufgaben werden Standard-Pläne aus einer Bibliothek bereitgestellt. Diese werden gestartet und ausgeführt [40, 191]. Weitere Beispiele für transformationelle d. h. planrevisionierende Algorithmen sind O-Plan [270] und IxTeT (IndeXed TimE Table) [167]. Darüber hinaus können Planer noch in zwei weitere Kategorien eingeteilt werden. Bei einem generativen Planer werden Pläne von einem leeren Anfangsplan aus entwickelt, d. h. es existieren zunächst keine Handlungen. Fallbasierte Verfahren verfolgen den Ansatz von Planfragmenten bzw. Wissensbasen. Ein generativer verfeinernder Planer ist der PO-Planer (Partical-Order-Planer) [190, 235, 281]. Allen Planern gemeinsam ist die Suche nach einer Verbesserung des aktuellen Plans während der Laufzeit. Wenn eine Verbesserung gefunden wurde, wird die Planausführung gestoppt und der aktuelle Plan durch den verbesserten Plan ausgewechselt. Die Planrepräsentation erfolgt durch Graphen, wobei jeder Knoten für einen bestimmten Weltzustand steht. Jede Kante entspricht einer Handlung, die einen Zustand in einen anderen Zustand überführt.

Die erste deliberative Steuerungsarchitektur ist das bereits 1971 vorgestellte STRIPS [85]. Die Beschreibung von Ist- und Soll-Zuständen, Aufgaben und daraus resultierenden Handlungen erfolgt in STRIPS

symbolisch mit Hilfe von Formeln und Operatoren (STRIPS Notation). Umgesetzt wurde STRIPS beispielsweise auf dem von 1966 bis 1972 am SRI International (USA) entwickelten mobilen Roboter Shakey [206]. Forschungsschwerpunkte dieses frühen Roboterprojekts liegen historisch bedingt im Bereich der künstlichen Intelligenz (KI) und dort vor allem bei der Planung von Handlungsabläufen.

1.2.2.2 Reaktive Steuerungsarchitekturen (’Sense-Act’)

Reaktiven Architekturen steht, im Gegensatz zu deliberativen Steuerungsarchitekturen, kein explizites Modell der Umwelt zur Verfügung. Sie können nur auf die von den Sensoren (’Sense’) gelieferten Informationen zur Durchführung von Handlungen (’Act’) zurückgreifen. Damit orientieren sich diese Architekturen in ihrer Funktionsweise an biologischen Vorbildern (vgl. reflexartige Reaktion des zentralen Nervensystems (ZNS) auf Reize ohne explizite Planungseingriffe durch das Gehirn). Entsprechende Ansätze verwenden dabei häufig verschiedene zielorientierte Regeln (Wenn-Dann-Struktur), die dann auf dem Robotersystem in Form von Modulen parallel ausgeführt werden (vgl. Bild 1.3).

Dadurch sind rein reaktive Steuerungsarchitekturen allerdings nicht in der Lage, komplexe Handlungssequenzen selbstständig zu planen, was ihr Anwendungsgebiet auf einfachere Roboteraufgaben wie z. B. Navigation, Kollisionsvermeidung und Personen- und Objektverfolgung beschränkt. Vorteilhaft ist, dass die langfristige Berechnung und Validierung von Weltmodelldaten entfällt und ein solches System dadurch zu schnellen Reaktionen fähig ist [58, 59]. Bild 1.3 zeigt die schematische Darstellung einer reaktiven Steuerungsarchitektur.

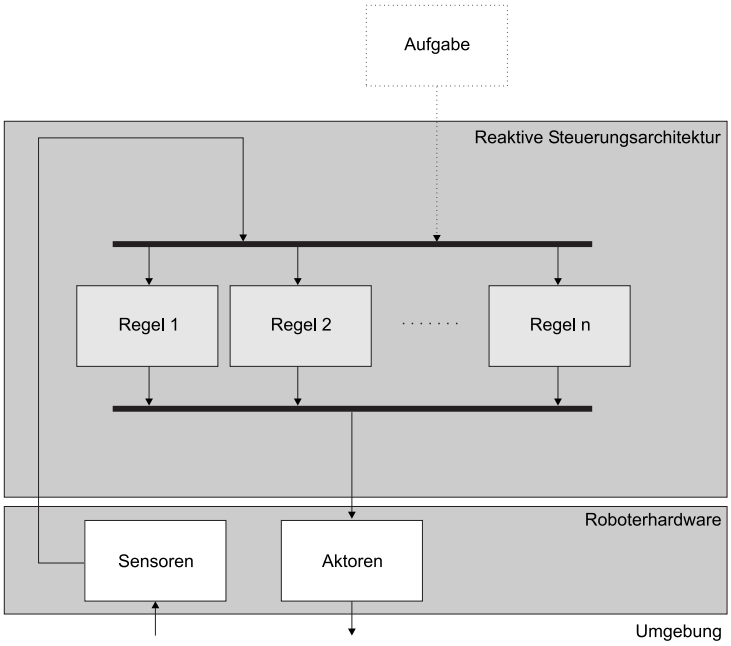


Bild 1.3: Schematische Darstellung einer reaktiven Steuerungsarchitektur nach [149] (modifiziert)

Einer der ersten und bekanntesten Vertreter reaktiver Steuerungsarchitekturen ist die ’Subsumption Architecture’ [58, 59]. Als weiterer reaktiver Ansatz sind die so genannten Verhaltensnetzwerke [181] zu nennen. Beispiele für die Umsetzung von reaktiven Architekturen auf Robotersysteme finden sich in [18] und [122]. Die Forschungen beschränken sich allerdings aufgrund der beschriebenen Nachteile auf Anwendungsgebiete der Navigation, der Kollisionsvermeidung sowie der Personen- und Objektverfolgung.

1.2.2.3 Verhaltensbasierte Architekturen

Verhaltensbasierte Architekturen stellen eine Erweiterung der reaktiven Ansätze dar. Wie die reaktiven Architekturen verzichten sie auf eine explizite symbolische Repräsentation der Umgebung. Grundlegende Roboterverhaltensweisen basieren auf einer engen Kopplung zwischen Wahrnehmung und Handlung. Komplexes Verhalten geht aus der Kooperation und Koordination primitiver, voneinander unabhängiger Verhaltensweisen hervor. Im Gegensatz zu reaktiven Systemen können mehrere übereinstimmende, aber auch widersprüchliche Verhalten parallel aktiviert werden. Dementsprechend müssen die Verhalten koordiniert werden, bevor mit der Ausführung einer Handlung begonnen wird. Die Koordination kann auf zwei unterschiedliche Weisen erfolgen. Entweder wird ein Verhalten ausgewählt und alle anderen ignoriert, oder die verschiedenen Verhalten werden zu einer neuen Handlung kombiniert.

Vorteile gegenüber zentralisierten, hierarchischen Architekturen sind, dass einzelne Verhaltensmuster kein vollständiges Wissen über die Umwelt benötigen und zudem der Ausfall einer Komponente (eines Verhaltensmusters) nicht den Ausfall des gesamten System nach sich zieht. Bild 1.4 zeigt die schematische Darstellung einer allgemeinen verhaltensbasierten Steuerungsarchitektur.

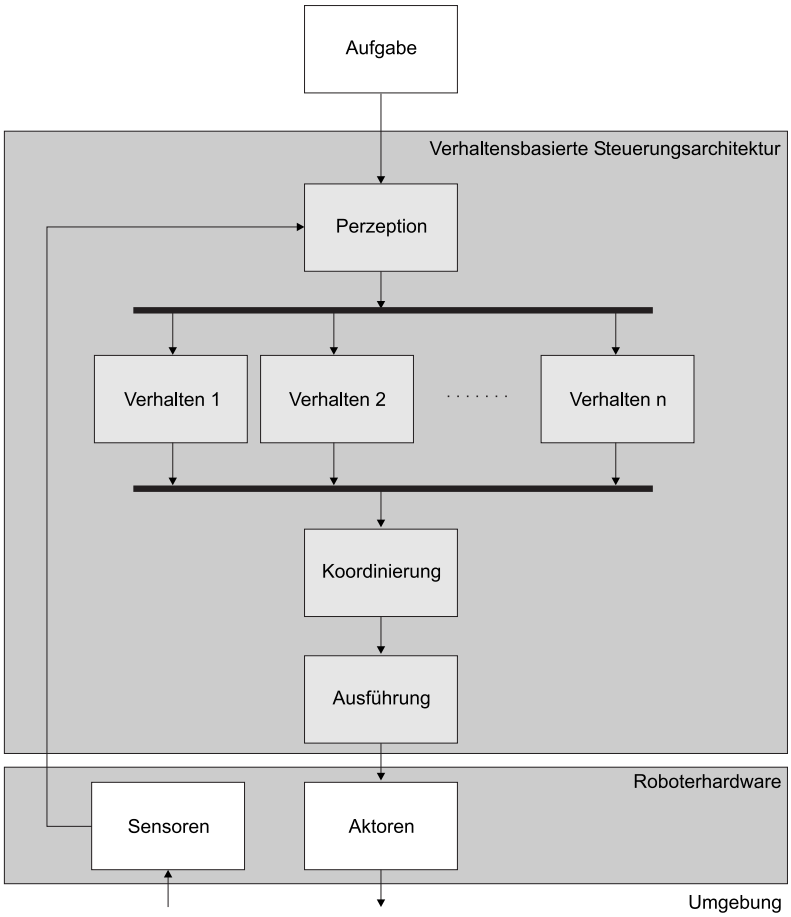


Bild 1.4: Schematische Darstellung einer verhaltensbasierten Steuerungsarchitektur nach [149] (modifiziert)

Ein Beispiel für eine verhaltensbasierte Architektur ist DAMN [232]. Eine weitere verhaltensbasierte Architektur, allerdings ohne Koordinierungskomponente, wird in [132] vorgestellt. Ein guter Überblick

zu verhaltensbasierten mobilen Robotern findet sich in [186]. Verhaltensbasierte Architekturen sind zudem im Bereich der humanoiden Robotik sehr verbreitet. Beispiele sind u. a. die Forschungsroboter DB [26, 48], Cog [60], Hermes [49, 50] und ISAC [210, 217] (vgl. Abschnitt 1.2.4).

1.2.2.4 Hybride Steuerungsarchitekturen

Eine Kombination aus den deliberativen und reaktiven Ansätzen stellt die hybride Steuerungsarchitektur dar. Die reaktive Komponente wird dabei für Standardfunktionen mit schneller Reaktionszeit verwendet. Für komplexere echtzeitunkritische Aufgaben wird dagegen der Planer der deliberativen Komponente benutzt. Die Strukturierung erfolgt bei diesem Ansatz hierarchisch. So hat sich eine Gliederung mit der deliberativen Komponente auf der obersten und der reaktiven Komponente auf der untersten Hierarchieebene als zweckmäßig erwiesen. Eine besondere Herausforderung bei dieser Architektur stellt die Kommunikation zwischen den verschiedenen Komponenten dar. Häufig wird zu diesem Zweck zwischen reaktiven und deliberativen Komponenten eine zusätzliche mittlere Ebene eingefügt. Die Entwicklung einer solchen Koordinierungskomponente stellt gewöhnlich die größte Herausforderung in einem hybriden System dar [187]. Resultierende Architekturen werden in der Literatur als 'Drei-Ebenen-Architekturen' bezeichnet.

Vorteile dieser Struktur sind eine echtzeitfähige Reaktionszeit (rein reflexartige reaktive Reaktion) gekoppelt mit der Fähigkeit zur Lösung komplexer Aufgaben (u. U. aufwändig geplante Sequenz von Handlungsabläufen). Bild 1.5 zeigt die schematische Darstellung einer hybriden Steuerungsarchitektur.

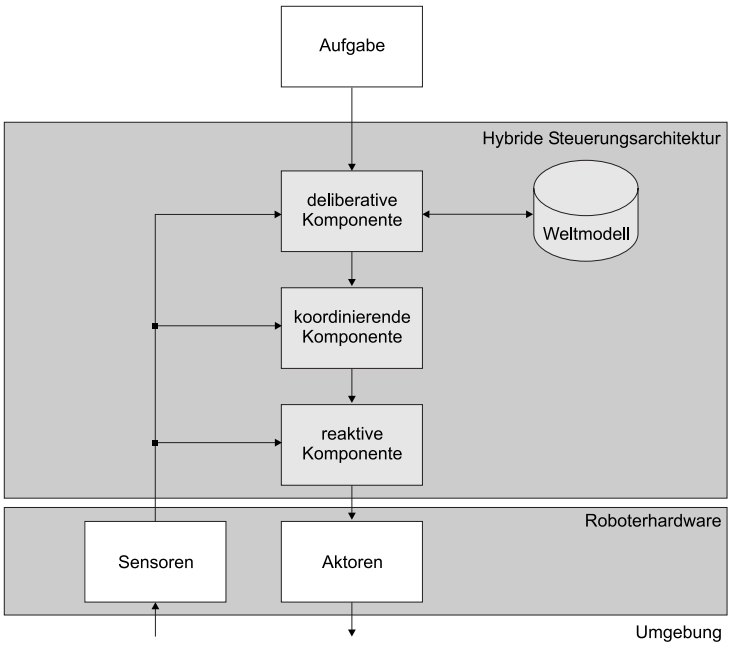


Bild 1.5: Schematische Darstellung einer hybriden Steuerungsarchitektur nach [149] (modifiziert)

Bekannte Beispiele für hybride Steuerungsarchitekturen sind AuRA [21], ATLANTIS [100], TCA [256] und 3T [55]. Wegen der beschriebenen Vorteile bieten sich hybride Steuerungsarchitekturen auch für den Einsatz in der humanoiden Robotik an. So wird beispielsweise derzeit im Sonderforschungsbereich 588 „Lernende und kooperierende multimodale Roboter“ an einer eigenen hybriden Steuerungsarchitektur

gearbeitet [62]. Diese Architektur umfasst neben den Komponenten der Perzeption und der sensomotorischen Steuerung (Aufgabenplanung, -koordination und -ausführung) auch kognitive Komponenten der Kommunikation (Dialog-Manager), des Lernens sowie der Wissensrepräsentation (globale und aktive Modelle). Die aufgabenrelevanten Komponenten der sensomotorischen Steuerung und die Komponenten der Perzeption sind dabei jeweils in der beschriebenen Drei-Ebenen-Struktur ausgeführt [55, 69, 88, 101]. Die Besonderheit dieses Ansatzes ist die Beschreibung der Planungs- und Koordinationskomponenten mit Hilfe von Petri-Netzen [221], anstelle der in der Literatur häufig verwendeten formalen logikbasierten, algebraischen Zustandsrepräsentation [85, 87]. Die Umsetzung dieser Architektur erfolgt mit Hilfe der Steuerungssoftware MCA2 [244] auf dem humanoiden Roboter ARMAR-III. Weiterhin ist davon auszugehen, dass die bekannten kommerziellen japanischen humanoiden Robotersysteme, wie z. B. ASIMO [116, 236], H7 [137, 151, 207] und HRP2 [127, 138, 139], über hybride Steuerungsarchitekturen verfügen (vgl. Abschnitt 1.2.4). Es existieren allerdings keine Veröffentlichungen, die diese Annahme im Detail beweisen.

1.2.2.5 Einordnung

Die Einsetzbarkeit einer Steuerungsarchitektur auf einem Robotersystem hängt stark von der geplanten Anwendung ab. Rein deliberative Architekturen werden zweckmäßiger Weise in gut durchdrungenen ”Spielwelten” eingesetzt, denn nur dort ist die exakte Modellierung der Umwelt und eine detaillierte off-line Planung der Handlung möglich. Typische Einsatzbereiche für deliberative, hierarchische Steuerungsarchitekturen sind beispielsweise kleinere mobile Roboter. Rein reaktive Kontrollsysteme kommen üblicherweise für weniger komplexe Aufgaben, wie z. B. Navigation, Kollisionsvermeidung und Personenverfolgung, in hochdynamischen Umgebungen zum Einsatz. Bei humanoiden Robotern, die sich in veränderlichen, komplexen Umgebungen mit unbekannten Situationen auseinanderzusetzen haben, ist die Verwendung beider Ansätze auszuschließen.

Im Vergleich dazu sind hybride und verhaltensbasierte Systeme weitaus leistungsfähiger (vgl. Bild 1.6). Prinzipiell sind beide Ansätze in ihren Möglichkeiten in etwa miteinander vergleichbar. Beide sind in der Lage, Repräsentationen ihrer Umwelt aufzubauen und Vorhersagen über die Zukunft zu treffen (Prädiktionsfähigkeit), allerdings unter Verwendung verschiedener Mechanismen. Verhaltensbasierte Systeme sind, im Gegensatz zu hybriden Systemen, durch eine aufgabenorientierte Modularisierung des Gesamtsystems charakterisiert. Wegen der genannten Vorteile existieren für beide Ansätze zahlreiche Anwendungen in der humanoiden Robotik. Verhaltensbasierte Architekturen werden dabei hauptsächlich im Bereich der Kognitionsforschung eingesetzt (vgl. Abschnitt 1.2.4). Beispiele für die erfolgreiche Umsetzung von hybriden Architekturen sind u. a. in der Service- und der Unterhaltungs-Robotik zu finden (vgl. Abschnitt 1.2.4).

Im Gegensatz zu den vorgestellten Architekturen autonomer Roboter wird in der Industrierobotik klassischerweise ein anderer Ansatz verfolgt. Da in der Montage teilweise sehr komplexe aber sich ständig wiederholende Aufgaben in bekannten Umgebungen ausgeführt werden müssen, kann auf eine autonome Aufgabenplanung durch den Roboter verzichtet werden. Stattdessen wird die gesamte Aufgabe, z. B. die auszuführende Bewegung im Raum, vom Bediener punkt- oder segmentweise in eine ausführbare Handlung umgesetzt, in der Robotersteuerung abgespeichert (Teach-In) und danach vom Roboter ausgeführt [254]. Sensoren kommen dabei nur zum Ausgleich von auftretenden Abweichungen von der geplanten Bahn zum Einsatz. Diese Vorgehensweise ist wegen der Forderung nach multimodaler Interaktions- und Kommunikationsfähigkeit für humanoide Roboter auszuschließen.

In Bild 1.6 folgt zusammenfassend eine Einordnung der beschriebenen Ansätze in Abhängigkeit von der Reaktionszeit und zu der realisierenden Autonomie sowie der zu verarbeitenden Aufgabenkomplexität bzw. Prädiktionsfähigkeit.

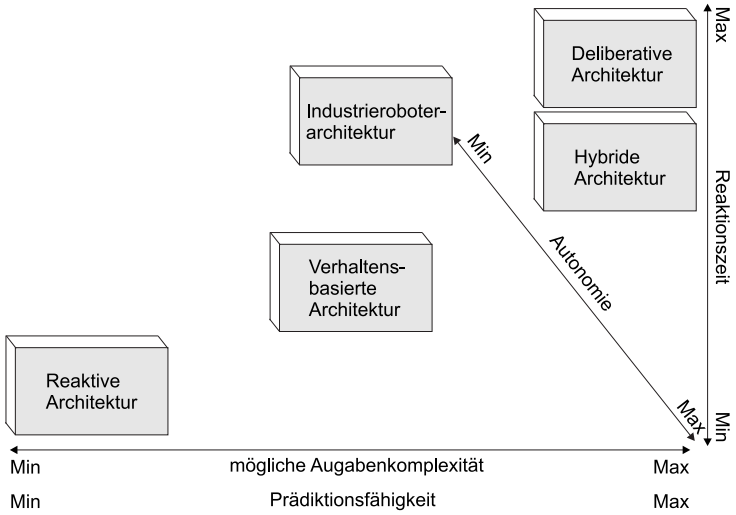


Bild 1.6: Einordnung verschiedener Steuerungsarchitekturen nach [77] (modifiziert)

Zu beachten ist weiterhin, dass bei allen hier vorgestellten Steuerungsarchitekturen keine expliziten Überwachungskomponenten integriert sind. Ein einheitliches und modular anwendbares Konzept hierfür fehlt bislang. Zusammenfassend ergibt sich, dass humanoide Robotersysteme über bestimmte Komponenten innerhalb ihrer Systemarchitektur verfügen müssen, um die an sie gestellten komplexen Anforderungen und Aufgaben erfüllen zu können. Dies sind u. a. Komponenten der Perzeption, der sensomotorischen Steuerung (Aufgabenplanung, -koordination und -ausführung) und der Sicherheitsüberwachung (vgl. Abschnitt 1.2.3) sowie kognitive Komponenten der Kommunikation (Dialog-Manager), des Lernens und der Wissensrepräsentation (globale und aktive Modelle).

1.2.3 Sicherheitsüberwachung technischer Systeme

1.2.3.1 Allgemeine Verfahren

Neben den Sicherheits- und Überwachungskonzepten im Bereich der Robotik existieren auch in anderen Industriebereichen sehr hohe Sicherheitsanforderungen. Als Beispiele sind hier die Kerntechnik [209], der Straßenverkehr [92], die Automobilindustrie und die Automatisierungstechnik genannt. Allgemeine Lehrbücher zur Sicherheit [161, 262], Zuverlässigkeit [41, 42, 71, 227] und Risikoanalyse [243] orientieren sich deshalb häufig an diesen Bereichen. Zusätzlich existieren eine Reihe von Normen [4, 8, 9] und Richtlinien [1], die bei der Entwicklung von technischen Geräten einzuhalten sind.

Prinzipiell lassen sich zwei Strategien unterscheiden. Eine Möglichkeit ist, einen Prozess bereits in der Entwurfsphase möglichst unempfindlich gegenüber Störungen auszulegen (präventive Methode). Dann ist im Idealfall kein zusätzliches aktives Überwachungssystem nötig (fehlertolerante Systeme mit passiver Robustheit). Ist dies nicht ausreichend, werden aktive Überwachungssysteme eingesetzt, die im Falle einer Störung frühzeitig korrigierend in den Prozess eingreifen. Zur aktiven Sicherheitsüberwachung muss ein solches System in der Lage sein, auftretende Fehler zu erkennen (Detektion) und geeignet zu klassifizieren (Isolation). Dieser Prozess wird in der Literatur allgemein als Fehlerdiagnose (engl: Fault Detection and Isolation FDI) bezeichnet. Im Anschluss daran erfolgt die Analyse des klassifizierten Fehlers, um geeignete Maßnahmen zur Problemlösung einzuleiten (Fehleranalyse). In [93] wird dazu ein allgemein anwendbarer dreistufiger Prozess vorgeschlagen:

1. Erkennen von nicht erlaubten Zuständen (Detektion),
2. Lokalisieren der erkannten Fehler (Isolation) sowie
3. Einleiten geeigneter Gegenmaßnahmen (Problemlösung).

Bild 1.7 zeigt schematisch den allgemeinen Prozess der Sicherheitsüberwachung mit Komponenten zur Fehlerdiagnose und Fehleranalyse.

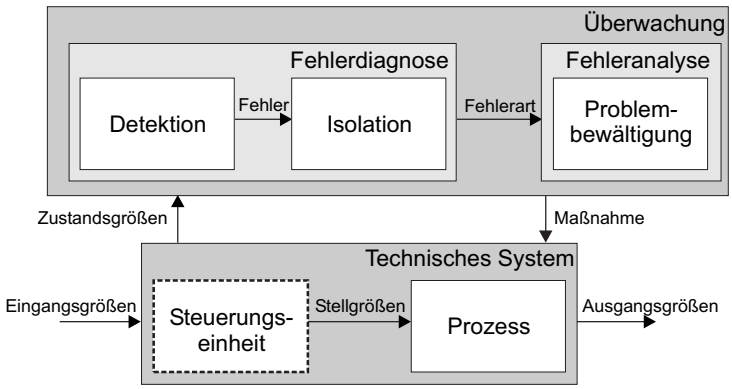


Bild 1.7: Prozess der Sicherheitsüberwachung nach [93] (modifiziert)

Fehler treten in einem technischen System im Anlagenteil (Prozess) und falls vorhanden in der Steuerungseinheit (Bild 1.7 gestrichelt) auf. Beispiele für Fehler sind:

- Materialfehler (Risse, Brüche, Lecks, . . .),
- Antriebsfehler (Lagerschäden, Getriebefehler, Motorfehler, . . .),
- Sensorfehler (Skalierungsfehler, Hysteresen, Drifts, Kontaktfehler, . . .),
- durch äußere Umstände hervorgerufene Funktionsstörungen (Kollisionen mit Fremdgegenständen, Stromausfall, . . .) sowie
- Softwaredefekte (Laufzeitfehler, Syntaxfehler, . . .).

Zur Diagnose von Fehlern gibt es zahlreiche unterschiedliche Verfahren. Grundlegend lassen sich diese zwei verschiedenen Konzepten zuordnen, den signal- und den modellgestützten Diagnoseverfahren. Bei signalgestützten Verfahren werden aus gemessenen Signalen geeignete Merkmale extrahiert und daraus auf die Fehler im Prozess geschlossen [93, 128, 131, 277]. Solche Merkmale sind beispielsweise arithmetische oder quadratische Mittelwerte, Grenzwerte, Trends, Korrelationskoeffizienten oder Kovarianzen. Nachteilig bei dieser Gruppe von Verfahren ist, dass die Möglichkeiten zur Fehlerfrüherkennung und damit zur Ausfallvermeidung eingeschränkt sind.

Leistungsfähiger, aber komplexer sind die modellgestützten Diagnoseverfahren. Bei den modellgestützten Verfahren wird zunächst, mit Hilfe einer möglichst vollständigen Beschreibung des dynamischen Verhaltens des Prozesses, ein mathematisches Modell aufgestellt. Treten hinreichend große Differenzen zwischen gemessenen (reales System) und berechneten Signalen (Modell) auf, wird auf einen Fehler im System geschlossen. Die modellgestützten Verfahren lassen sich weiterhin nach der Form der Modellbeschreibung unterscheiden. So kann der Prozess entweder quantitativ (analytisch) oder qualitativ (heuristisch) beschrieben werden. Beispiele für die Anwendung von quantitativ analytischen Verfahren sind in [80, 81, 94, 283, 284, 286] zu finden. In qualitativen wissensbasierten Modellen ist

das Wissen über den Prozess in Form von Regeln und Fakten hinterlegt. Beispiele für Anwendungen sind [29, 95, 96, 237, 242]. Weitere Möglichkeiten ergeben sich durch Kombination verschiedener Überwachungskonzepte (Redundanz). So wird beispielsweise durch den Abgleich der von den verschiedenen Überwachungssystemen gelieferten Daten eine Fehlerdetektion möglich. Bild 1.8 veranschaulicht den Zusammenhang der beschriebenen Konzepte.

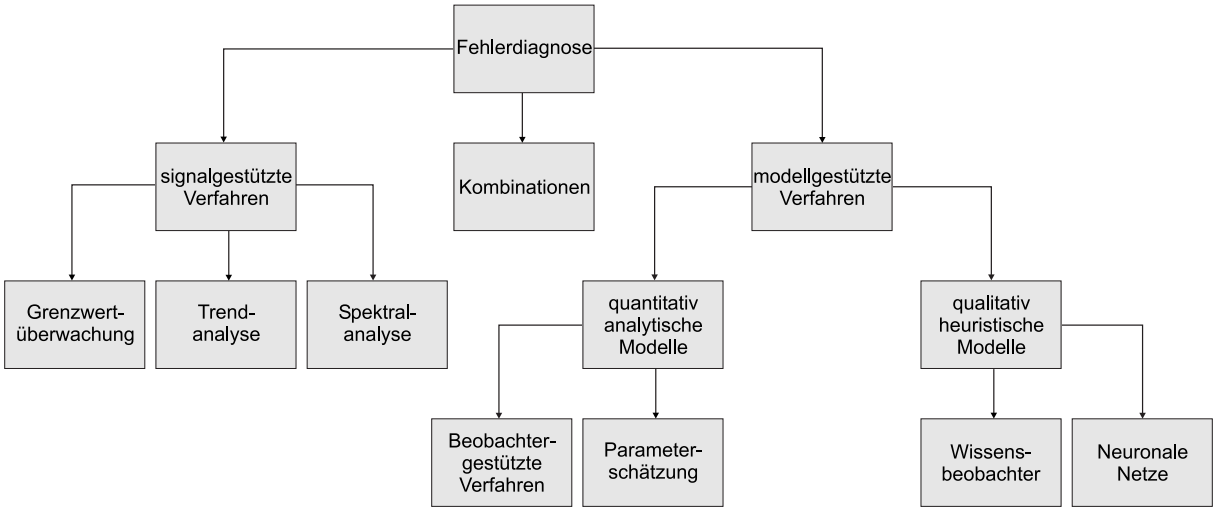


Bild 1.8: Übersicht über Verfahren zur Fehlerdiagnose im Anlagenteil eines Prozesses nach [93]

Ein wichtiges allgemein anwendbares Prinzip zur Fehlerdetektion ist die Ausnutzung von Redundanz. Diese lässt sich entweder physikalisch durch mehrfache Ausführung bestimmter Funktionseinheiten (z. B. mehrere Sensoren messen die gleiche Systemgröße) oder analytisch mit Hilfe eines im Hintergrund mitlaufenden mathematischen Prozessmodells erreichen. Die analytische Redundanz lässt sich je nach verwendetem mathematischen Modell wiederum in quantitative (analytische) oder qualitative (wissensbasierte) Redundanz unterteilen [93].

Weitere klassische Verfahren zur Detektion von Fehlern in technischen Systemen sind die FMEA, Abkürzung für Fehlermöglichkeits- und Einfluss-Analyse und die FTA, Abkürzung für Fehlerbaum-Analyse. Bei der FMEA [2] wird ein Untersuchungsobjekt in einem bestimmten Entstehungs-, Entwicklungs- und/oder Planungsabschnitt auf mögliche Fehler untersucht, deren Risiko abgeschätzt sowie bei Bedarf eine Risikominimierung durchgeführt. Die FMEA ist eine präventive Methode und dient dazu, Fehler im Einsatz eines Produktes zu vermeiden. Es existieren in der Praxis viele verschiedene FMEA-Arten mit unterschiedlichen Zielsetzungen. Die klassischen FMEA-Arten sind System-FMEA, Konstruktions-FMEA und Prozess-FMEA. Die Fehlerbaumanalyse [3] dient zur Abbildung des Funktionssystems und zur Quantifizierung der Systemzuverlässigkeit. Ausgehend vom ’unerwünschten Ereignis’ (Systemausfall) werden ’TOP down’ die Funktionen/Ausfallzustände der Komponenten und Bedienungsmaßnahmen eines Systems ermittelt. Es entsteht das boolesche Modell (der Fehlerbaum), das unter Verwendung der Zuverlässigkeitskenngrößen quantifiziert wird. Ein Überblick über diese Methoden ist in [135] zu finden. In [229]und [278] werden Anwendungen der beschriebenen Verfahren im Bereich der Robotik vorgestellt.

Ist das mathematische Modell der zu regelnden Strecke bekannt, können Verfahren der nichtlinearen Regelungstheorie [91] zur präventiven Stabilitätssicherung eingesetzt werden. Diese sind in Zeitbereichs- und Frequenzbereichsmethoden zu unterteilen. Zu den ersten gehören u. a. die Zustandsraummethode [66, 123], die Ljapunov-Theorie [65, 67] und die Graphentheorie [259]. Zu den Methoden, die den

Frequenzbereich verwenden, sind beispielsweise die Harmonische Balance [53, 54, 148], das Popov-Kriterium [52, 61] und Verfahren über Normabschätzungen [20, 134, 223] zu zählen. Der formale Stabilitätsnachweis vieler Regelkreise ist z. B. wegen schwer modellierbarer Störeinflüsse und/oder kontinuierlich-diskreter Systemeigenschaften für praktisch relevante Aufgaben sehr schwierig. Aufgrund der Komplexität sind Instabilitäten und daraus resultierende Sicherheitsprobleme allerdings nicht auszuschließen. Ein Lösungsansatz ist der Einsatz einer aktiven Fuzzy-Überwachungsebene [192], die auftretende Abweichungen vom Sollverhalten und potenzielle Gefahrensituationen auf der Ebene der Basisregler erkennt und sowohl durch direkte Modifikation der Reglereinstellungen, als auch durch Weiterleiten der detektierten Gefahrensituation an übergeordnete Planungsebenen, geeignete Gegenmaßnahmen veranlasst [170].

Im Rahmen des DFG-Schwerpunktprogramms 1016, 'Analyse und Synthese kontinuierlich-diskreter technischer Systeme' (KONDISK [157, 158]) wurde eine durchgängige Methodik zur Synthese von Koordinationssteuerungen und robusten hybriden Regelungen für diskret-kontinuierliche mechatronische Systeme entwickelt. Das Regelungsziel besteht darin, ein hybrides System mit mehreren Freiheitsgraden einem geeignet spezifizierten Referenzverhalten möglichst gut nachzuführen und zusätzlich externe Störeinflüsse zu kompensieren. Störungen können dabei sowohl zu Abweichungen in den kontinuierlichen Prozessgrößen als auch zu fehlerhaften diskreten Systemzuständen führen. Konkrete Anwendungen für derartige mechatronische Systeme finden sich u. a. in der Fahrzeugtechnik, der Antriebstechnik oder der Robotik (z. B. mehrfingrige Roboterhände) [239].

Die hier beschriebenen Verfahren konzentrieren sich hauptsächlich auf die Fehlerdiagnose (vgl. Bild 1.7). In [93] wird zudem ein Prozessüberwachungssystem vorgestellt, das Komponenten zur Fehlerdiagnose und Fehleranalyse enthält. Dieses System ist in der Lage, Vorschläge für Korrektureingriffe zu geben. Ein automatischer Korrektureingriff findet dort allerdings nicht statt, die letzte Entscheidung bleibt immer beim Menschen. So ist eine bei humanoiden Robotern erforderliche autonome Problemlösung in den allgemein anwendbaren Konzepten derzeit nicht realisiert.

1.2.3.2 Überwachung von Robotern

Der derzeitige Stand der Sicherheitsüberwachung im Bereich der Industrierobotik ist in der europäischen Norm ISO 10218: 'Industrieroboter - Sicherheit' [5] dokumentiert. Gemäß [5] müssen Schutzmaßnahmen für den Bediener bei industriellen Anwendungen durch

- die Anbringung trennender Schutzeinrichtungen,
- die Anwendung von Schutzeinrichtungen mit Annäherungsreaktion sowie
- die Anwendung von sicheren Verfahren

getroffen werden. Die ersten beiden Methoden sind für ein multimodal mit dem Menschen kooperierendes Robotersystem nicht sinnvoll, da die räumliche Trennung von Robotersystem und Bediener in diesem Fall unerwünscht ist. Als sicheres Arbeitsverfahren wird ein Verfahren definiert, das in der Lage ist, die Möglichkeit einer Verletzung während der Ausführung einer zugewiesenen Aufgabe zu vermeiden [5]. Eine genauere Spezifizierung findet jedoch nicht statt, da in der Industrierobotik der Eingriff in den Arbeitsraum des Robotersystems als Ausnahmesituation angesehen wird, z. B. bei der Wartung, der Reinigung, der Reparatur oder der Prozessumstellung. Es gibt jedoch Bestrebungen neue Sicherheits- und Überwachungskonzepte zu entwickeln und einzusetzen, um damit die Flexibilität der Industrie-Robotersysteme zu erhöhen. Beispiele hierfür sind in [258, 265, 266, 272] zu finden. In [258] wird beispielsweise eine Robotersteuerung vorgestellt, die den sicheren Betrieb eines Robotersystems ohne

trennende Schutzeinrichtungen erlaubt. [265] schlägt ein flexibles sensorbasiertes Sicherheitskonzept für einen Industrieroboter in der Produktion vor.

Anders sieht es im Bereich der so genannten Roboterassistenten aus, die den Bediener physisch und/oder informationell bei seiner Arbeit unterstützen sollen [111]. Damit ist die strikte Trennung der Arbeitsbereiche von Robotersystem und Bediener aufgehoben und es ergeben sich deutlich verschärfte Sicherheitsanforderungen. Einen Überblick über den Stand der Technik bei der Sicherheit von mobilen Robotern gibt [12]. Einfache Möglichkeiten, die Sicherheit solcher Systeme zu erhöhen, sind konstruktiv festgelegte passive Nachgiebigkeiten oder aktive Nachgiebigkeiten infolge des verwendeten Regelungskonzepts bzw. Kombinationen von beiden [74, 176, 255]. Ein aufwändigeres Konzept wird in [287] beschrieben. Es basiert auf der sensorischen Überwachung des Arbeitsraums des Robotersystems sowie der Schätzung der geplanten Bewegung des Bedieners zur Vermeidung von Kollisionen. In [50, 51] wird das Sicherheitskonzept des humanoiden Roboters HERMES vorgestellt. Eine Besonderheit bei diesem Konzept ist die Verwendung von 'watch dog timern' auf allen Ebenen der Steuerungsarchitektur (Hauptprozessor (CPU), Digitaler Signalprozessor (DSP) und Mikrocontroller). Damit ist es möglich, Programmablauffehler zu erkennen und den Roboter bei auftretenden Fehlern mit Hilfe von elektronischen Notschaltern auszuschalten. Das Sicherheitskonzept des vom Fraunhofer Institut für Produktionstechnik und Automatisierung (IPA) entwickelten Roboter Assistenzsystems Care-O-bot setzt sich aus einer Beschränkung des Arbeitsraums mit Hilfe von Magnetbändern und einer Kollisionsvermeidung basierend auf taktiler und Infrarot-Sensorik zusammen. Falls die Grenzen des Arbeitsraums erreicht sind oder eine Kollision erkannt wird, wird der Roboter abgeschaltet. Um den Roboter wieder einzuschalten, muss vom Bediener manuell ein Entriegelungsschalter betätigt werden [245].

Zudem verfügen vor allem humanoide Robotersysteme, bei denen die Forschungsschwerpunkte im Bereich der kognitiven Fähigkeiten liegen, über Strategien zum Wissenserwerb (Lernen), zur Selbstreflexion und zur autonomen Problemlösung (vgl. Abschnitt 1.2.4.2). Damit sind sie prinzipiell in der Lage, selbstständig Pläne zur Lösung von Aufgaben zu erstellen, Pläne während der Planausführung zu ändern und/oder, falls notwendig, die Ausführung der Pläne zu unterbrechen oder zu stoppen, um beispielsweise bei erkannten Ausnahmesituationen (erfordert ein integriertes Überwachungskonzept) einen Klärungsdialog zu initiieren. Diese Fähigkeiten werden dabei mit Hilfe von deliberativen bzw. hybriden Steuerungsarchitekturen auf humanoiden Robotersystemen umgesetzt (vgl. Abschnitt 1.2.2). Allerdings sind bei humanoiden Robotern, die sich in der realen, ständig ändernden Welt bewegen, autonome Planungsprozesse häufig zu rechenintensiv. Aus diesen Anforderungen resultiert die Notwendigkeit eines neuen flexiblen Sicherheits- und Überwachungskonzepts, das in der Lage ist, schnell und flexibel auf zum Teil unbekannte Gefahrensituationen zu reagieren. Ein einheitliches modular anwendbares Konzept hierfür fehlt bislang.

1.2.4 Humanoide Robotersysteme

Nach Abschnitt 1.1 lassen sich zwei Forschungsschwerpunkte im Bereich der humanoiden Robotik definieren. Bei der Konstruktion einer multifunktionalen Arbeitsmaschine liegen die Schwerpunkte bei der Entwicklung von Roboterkomponenten (Aktorik und Sensorik) und beim Entwurf geeigneter Steuerungs- und Regelungsalgorithmen, die die Durchführung vielfältiger motorischer Fähigkeiten im menschlichen Umfeld ermöglichen. So sind humanoide Roboter heutzutage zunehmend in der Lage, sich selbstständig, kollisionsfrei und sicher zu bewegen (z. B. laufen (LA), rennen (RE), Treppen steigen (TS), tanzen (TA), vom Boden aufstehen (A) oder mittels einer Plattform fahren (P)), und/oder verschiedene Manipulations- und Handhabungs- (z. B. greifen (GR), manipulieren (MA), transportieren (TR)) bzw. Bewegungsaufgaben (z. B. jonglieren (JO), trommeln (TRO), winken (WI), Bälle werfen (BW),

Surf- (SU) und Tennisbewegungen imitieren (TE) oder Keyboard spielen (KEY)) zu erledigen (vgl. Tabelle 1.1, S. 18).

Bei der Erschaffung künstlicher Intelligenz liegen die Arbeitsschwerpunkte bei der Erforschung kognitiver Fähigkeiten. Solche charakteristischen kognitiven Fähigkeiten und Merkmale sind nach [268] das Wahrnehmen und Speichern von Inhalten, um Ziele zu erreichen (Perzeption (PE) und Problemlösung (PL)), das Interagieren unter Echtzeitbedingungen (Interaktion (I) und Kommunikation (K)), das Erwerben und Repräsentieren von Wissen (Lernen (LE), Bewusstsein (BE) und Gedächtnis) sowie das selbstständige Planen und Ausführen von Aufgaben (PA) (vgl. Tabelle 1.1, S. 18).

Im Weiteren werden zunächst die humanoiden Robotersysteme vorgestellt, bei denen die motorischen Fähigkeiten im Fokus der Entwicklung stehen. Dabei gilt die Realisierung des zweibeinigen Laufens unter realen Umgebungsbedingungen als derzeit schwierigste mechatronische Herausforderung bei der Entwicklung humanoider Roboter. Zunächst werden aus diesem Grund die humanoiden Robotersysteme vorgestellt, die sowohl auf zwei Beinen laufen als auch Gegenstände greifen und manipulieren können. Danach erfolgt ein Überblick über die Robotersysteme, bei denen der Schwerpunkt der Forschung eher bei den kognitiven Fähigkeiten liegt. Zu beachten ist allerdings, dass bei vielen Roboterprojekten keine eindeutige Trennung zwischen den beiden hier formulierten Schwerpunkten vorgenommen werden kann (z. B. ARMAR-III Sonderforschungsbereich 588, Universität Karlsruhe (TH)). So setzt die Erforschung vieler kognitiver Fähigkeiten, wie z. B. Interaktion, Kommunikation oder Aufgabenplanung und -ausführung, vorhandene motorische Fähigkeiten des verwendeten Robotersystems voraus.

1.2.4.1 Forschungsschwerpunkt: Motorische Fähigkeiten

Die Roboter mit den am weitesten fortgeschrittenen motorischen Fähigkeiten stammen derzeit von der Firma Honda (Japan). Dort wurden in den letzten 12 Jahren vier aufeinander aufbauende humanoide Roboter entwickelt. Dies sind der P1, der P2 [115], der P3 [116] und das neueste Modell ASIMO (*Advanced Step in Innovative Mobility*). Alle vier Roboter sind in der Lage, auf zwei Beinen zu laufen (ASIMO max. 3.0 km/h), Treppen auf und ab zu steigen und Objekte zu manipulieren. Für ASIMO entwickelte Honda ein neues echtzeitfähiges prädiktives Regelungskonzept mit dem Namen i-Walk [117]. Damit ist es ASIMO möglich, die nächste Bewegung in Echtzeit im Voraus zu planen, um dann bereits vor der eigentlichen Ausführung dieser Bewegung den Körperschwerpunkt in die richtige Richtung zu verlagern. Als derzeit einziger Roboter kann ASIMO damit seine Schrittlänge (lang oder kurz) und seine Laufgeschwindigkeit (schnell oder langsam) kontinuierlich variieren. Ein Vision-System (VS), bestehend aus Stereokameras, und ein auditives System (AS) aus Mikrofonen ermöglicht die Wahrnehmung der Umwelt und eine eingeschränkte Interaktion mit dem Menschen [236]. Der Fortschritt der Entwicklung zeigt sich im Vergleich der Abmessungen und des Gewichtes der Roboterplattformen. So ist ASIMO der kleinste und leichteste der vier Roboter (siehe Tabelle 1.1). Insgesamt verfügt ASIMO über 26 unabhängige Freiheitsgrade (2 DOF (engl.: degree of freedom) Kopf, 5 DOF pro Arm, 1 DOF pro Greifer und 6 DOF pro Bein [121]).

Auch der von 1996-1998 an der Universität von Tokyo (JSK, Japan) entwickelte, modular aufgebaute Leichtbau-Roboter Saika [151] kann auf zwei Beinen laufen und Gegenstände greifen. Saika besitzt dazu einen Hals mit zwei Freiheitsgraden, zwei Arme mit jeweils fünf Freiheitsgraden, einen Torso und einen Kopf. Zusätzlich wurden mehrere verschiedene Greifertypen entwickelt [152, 153]. Ab 1998 wurden in Zusammenarbeit mit der Aircraft and Mechanical Systems Division von Kawada Industries Inc. (Japan) drei weitere aufeinander aufbauende humanoide Roboterplattformen mit den Namen H5, H6 [137, 207] und H7 entwickelt. Alle drei verfügen über 35 unabhängige Freiheitsgrade (6 DOF pro Bein, 1 DOF pro Fuß, 7 DOF pro Arm, 1 DOF pro Greifer, 2 DOF Hals und 3 DOF Augen). Alle Gelenke werden

mit Gleichstrommotoren und Harmonic Drive Getrieben angesteuert. H6 und H7 sind damit in der Lage zu laufen, Treppen zu steigen (max. Höhe 25 cm), Bälle zu werfen und zu winken. Von der Kawada Industries Inc. (Japan) und dem National Institute of Advanced Industrial Science and Technology (Japan) wurde im Rahmen des Humanoid Robotics Project (HRP) von 1998 bis 2002 der maximal 2.5 km/h schnelle humanoide Roboter HRP-2 entwickelt [127]. HRP-2 besitzt insgesamt 30 unabhängige Freiheitsgrade (6 DOF pro Bein, 6 DOF pro Arm, 1 DOF pro Greifer, 2 DOF Hals [139]). Als Besonderheit verfügt HRP-2 über zwei weitere Freiheitsgrade im Torso, die es ihm ermöglichen, als erster humanoider Roboter selbstständig vom Boden aufzustehen [138]. HRP-2 ist dazu mit 30 Winkelencodern, drei Beschleunigungs-, vier Kraft-, drei Gyrosensoren sowie drei CCD-Kameras ausgestattet [139]. Über das dazu notwendige aufwändige Regelungskonzept ist in der Literatur nichts zu finden.

Die Firma Sony (Japan) präsentierte mit dem Sony Dream Robot-3X (SDR-3X) im Jahr 2000 [166] und seinem Nachfolger dem SDR-4X [165] im Jahr 2002 zwei kleine (siehe Tabelle 1.1) humanoide Unterhaltungsroboter. Beide können gehen, tanzen, einige Worte sprechen und auf bestimmte Sprachkommandos reagieren. Sony entwickelte für SDR-4X ein adaptives Echtzeit-Regelungskonzept, mit dem der Roboter in Lage ist, unebenes Terrain zu durchqueren sowie Abhänge zu meistern. Weiterhin kann SDR-4X die Balance auf einer sich bewegenden Plattform halten und so Surfbewegungen imitieren. Detailinformationen zu diesem Regelungskonzept sind in der Literatur allerdings nicht zu finden. Mit Hilfe seiner Bild- und Spracherkennungs-Software ist er zudem fähig, bekannte Personen zu identifizieren. Angesteuert wird SDR-4X über Servomotoren [98].

Auch das Korea Advanced Institute of Science and Technologie präsentierte Anfang 2005 erstmals einen humanoiden Roboter mit den Namen HUBO. Er besitzt insgesamt 41 Freiheitsgrade, kann auf zwei Beinen mit einer Geschwindigkeit von 1.25 km/h laufen, sprechen und Sprachkommandos verstehen [136].

Weitere humanoide Roboter mit etwa vergleichbaren motorischen Fähigkeiten wurde von der Toyota Motor Company (Japan) im Frühjahr 2004 vorgestellt. Es handelt sich dabei um so genannte ”partner robots”, die den Menschen als persönlichen Assistenten im Haushalt unterstützen sollen. Einer davon kann auf zwei Beinen laufen und ist in der Lage, eine Vielzahl verschiedener allerdings nicht näher spezifizierter Handlungen mit seinen Händen durchzuführen. Von seinen Abmessungen und seinem Gewicht ist er mit Hondas ASIMO vergleichbar.

Ein reiner Laufroboter stammt von der TU München (Deutschland). Innerhalb des DFG-Schwerpunktprogramms „Autonomes Laufen“ beschäftigt sich der Lehrstuhl für Angewandte Mechanik mit der Realisierung der zweibeinigen autonomen Laufmaschine Johnnie. Der Roboter ist in der Lage, geradeaus (max. 2.0 km/h) und um Kurven zu gehen. Primäres Ziel ist das Erreichen eines schnellen, dynamisch stabilen Ganges. Dies stellt hohe Anforderungen an Aktorik, Sensorik und Regelung [177]. Johnnie verfügt über einen menschenähnlichen Aufbau mit 17 angetriebenen Gelenken (6 DOF pro Bein, 1 DOF Oberkörper, 2 DOF pro Arm). Die Arme werden zum dynamischen Drallausgleich eingesetzt. Alle Gelenke werden mit Gleichstrommotoren angetrieben. Neben den Antrieben sind Sensorik, Elektronik und Steuerungs-PC auf der Maschine integriert. Damit erreicht Johnnie ein hohes Maß an Autonomie. Lediglich die Energiezufuhr erfolgt über ein Kabel. Lage (PS) und Geschwindigkeit (DPS) der Gelenke werden mit inkrementellen Winkelencodern gemessen. Die Ermittlung der Bodenreaktionskräfte übernehmen zwei in den Füßen integrierte sechsachsige Kraftsensoren (KS). Ein Orientierungssensorsystem (GS) bestehend aus einem dreiachsigen Beschleunigungsmesser und drei Kreiselsensoren erfasst die räumliche Drehlage des Oberkörpers [103].

Aufgrund der technischen Komplexität humanoider Robotersysteme existieren weiterhin Forschungsprojekte, die sich auf die Entwicklung einzelner Roboterkomponenten konzentrieren. Solche speziellen Komponenten sind beispielsweise Leichtbauarme [16], Mehrfinger-Greifer [63, 99, 180, 253], anthropomorphe Roboterköpfe [113, 147, 257] und zweibeinige Laufmaschinen [155, 156]. Allerdings kann der

angestrebte Leichtbau zum Auftreten von Elastizitäten in den Achsen und/oder Gelenken führen. Dann hat die Regelung die Aufgabe, den Einfluss dieser Elastizitäten zu kompensieren oder gezielt auszunutzen. Beispiele hierfür sind passivitätsbasierte Regelungen, Sliding-Mode-Regelungen und Backstepping [17]. Andere Anwendungsfelder für komplexe Reglerstrukturen wie z. B. Nachgiebigkeits- und hybride Kraft-Positionsregelungen [170, 239, 254] sind Greifbewegungen oder gekoppelte Greif- und Manipulationsvorgänge. Diese erfordern die koordinierte Regelung mehrerer mechatronischer Komponenten (z. B. Greifer, Arme, Kopf und Plattform des Robotersystems). Weiterhin kann bei solchen komplexen Systemen durch den Einsatz optimierter, modellbasierter Regelungskonzepte häufig eine deutliche Verbesserung hinsichtlich der Anforderungen an das Sollverhalten erzielt werden. Diese Konzepte erfordern allerdings eine genaue Kenntnis der internen Systemzustände. Fehlt die dazu notwendige Sensorik am realen System, können die erforderlichen Daten mit Hilfe eines Simulationsmodells geschätzt werden (Beobachter). Zusätzlich kann das Modell für eine verbesserte Sensorüberwachung und als Ersatz für ausgefallene Sensorsignale verwendet werden, wodurch die Zuverlässigkeit und Betriebssicherheit weiter verbessert wird. Regelungstechnische Standardlösungen für solche komplexen Systeme existieren jedoch nicht.

1.2.4.2 Forschungsschwerpunkt: Kognitive Fähigkeiten

Die Japan Science and Technology Corp. arbeitet an einem humanoiden Roboterprojekt mit dem Namen DB (Dynamic Brain). Der Roboter besitzt insgesamt 30 unabhängige Freiheitsgrade (3 DOF Kopf, 7 DOF pro Arm, 3 DOF Torso, 3 DOF pro Bein und 2 DOF pro Auge). Bis auf die Freiheitsgrade der Augen, die nur über Positionssensorik (PS) verfügen, sind alle Freiheitsgrade mit Positions- und Kraftsensorik (KS) versehen. Der Antrieb der Aktoren erfolgt hydraulisch [26]. DB ist in der Lage zu trommeln sowie Tennis-Ausholbewegungen zu imitieren, kann aber nicht selbstständig laufen. Die beiden letzten Fähigkeiten resultieren aus neuen Ansätzen zur Bewegungsplanung, online Trajektorienmodifikation und Lernen durch Imitation vom Menschen [47, 48, 125, 126].

Die ersten humanoiden Roboter stammen von der Waseda Universität (Japan). Dort wurde bereits 1970 mit der Entwicklung des anthropomorphen[1] Roboters WABOT-1 begonnen. WABOT-1 konnte auf zwei Beinen laufen (LA) und mit seinen Händen Gegenstände greifen und transportieren [140, 141]. Von 1980-1984 wurde der Nachfolger WABOT-2 entwickelt. WABOT-2 konnte sprechen (K), Noten lesen, Melodien mittlerer Schwierigkeitsstufe auf dem Keyboard spielen [269]. Die kognitiven Schwerpunkte der Forschungsarbeiten dieser Projekte lagen bei der Interaktion und Kommunikation mit dem Benutzer.

Der Roboter Jack vom Electrotechnical Laboratory der Universität Tsukuba (ETL, Japan) verfügt über insgesamt 46 Freiheitsgrade. Auf dem Kopf befindet sich ein Stereo Vision-System (VS) bestehend aus zwei CCD-Videokameras. Weiterhin ist Jack mit einem auditiven System (AS) bestehend aus Mikrofon-Arrays und Positionssensoren in allen Gelenken ausgerüstet. Kognitive Forschungsschwerpunkte sind lernbasierte Ansätze, sowie die Interaktion und Kommunikation mit dem Benutzer [68, 162].

Am Massachusetts Institute of Technology (MIT, USA) wird die anthropomorphe Roboterplattform Cog entwickelt. Cog besteht aus einem Torso, zwei Armen und zwei Greifern und einem Kopf und verfügt über insgesamt 21 Freiheitsgrade [60]. Er ist mit Sensorsystemen ausgerüstet, die den Sinnen des Menschen nachempfunden sind. So verfügt Cog über Vision- (VS), Berührungs- (KS), Propriozeptions[2]- und Vestibulär (GS)[3]-Sensorik. Zudem verfügt Cog über ein auditives System (AS). Schwerpunkte der Forschung sind die verhaltensbasierte Robotik, soziale Interaktion, Empathie sowie Lernen durch Imitation

[1] anthropomorph: Von menschlicher Gestalt, menschenähnlich, menschlich.
[2] Propriozeption: Wahrnehmung der Lage der beweglichen Teile des Körpers
[3] vestibulär: den Gleichgewichtssinn betreffend

vom Menschen. Zudem beschäftigen sich die Forscher am MIT mit den kognitive Komponenten der Perzeption, der sensomotorischen Steuerung und der Wissensrepräsentation [11, 58, 59].

An der Bundeswehr Universität in München (Deutschland) wurde 1997 der humanoide Serviceroboter HERMES vorgestellt. HERMES besitzt insgesamt 18 Freiheitsgrade (3 DOF omnidirektionale Plattform, 6 DOF pro Manipulationssystem und 2 DOF Kopf). Das Sensorsystem von HERMES besteht aus einem Stereo Vision-System (VS) mit zwei monochromen Videokameras, Positionssensorik (PS) in Form von Winkelencodern sowie Temperatursensoren (TS). Weiterhin verfügt HERMES über eine Mensch-Maschine-Schnittstelle zur Programmierung und Überwachung. Damit ist er in der Lage, unbekannte Umgebungen zu erforschen, Transport- und Manipulationsaufgaben auszuführen und mit unbekannten Bedienern zu kommunizieren und zu interagieren [49, 50].

Im Rahmen des Sonderforschungsbereichs 588 „Lernende und kooperierende multimodale Roboter“ wird in Karlsruhe (Deutschland) an dem humanoiden Robotersystem ARMAR gearbeitet, das in der Lage ist, mit dem Menschen in einer gemeinsamen Umgebung zu kooperieren [76]. Anwendungsgebiete sind dabei sowohl die Arbeitswelt, als auch der private Lebensbereich des Menschen. Weiterhin sollen die Bewegungen und damit das Verhalten des Roboters, möglichst menschenähnlich sein. Aus diesen Anforderungen ergeben sich neue wissenschaftliche Ziele und somit neue Herausforderungen in den Forschungsbereichen, Mechatronik, Simulation und Modellierung [27, 30, 43, 45, 46], Perzeption[4] [32, 33, 104, 119, 263, 264] sowie Kooperation und Lernen [82, 83, 97, 112, 234, 271, 288, 289]. Der Bereich Mechatronik gliedert sich in die Entwicklungsschwerpunkte Konstruktion mechanischer Roboterkomponenten und Regelung. Zum Ersten gehören die Entwicklung des flexiblen Torsos, der Leichtbauarme und des Sensorkopfs [13, 14, 15, 23, 25], die Integration der mobilen Plattform [246] sowie die Konstruktion und Entwicklung der Greifer bzw. Hände des Roboters [250, 251, 252]. Ziel der Regelung ist die Entwicklung und Untersuchung geeigneter Konzepte zur Überwachung und Regelung von humanoiden Robotern in komplexen Szenarien, deren Lösung entscheidend für die Güte und Sicherheit der Bewältigung der geforderten Aufgabe ist [105, 163, 164, 170, 173, 197]. Bild 1.9 zeigt die humanoiden Robotersysteme des Sonderforschungsbereichs 588.

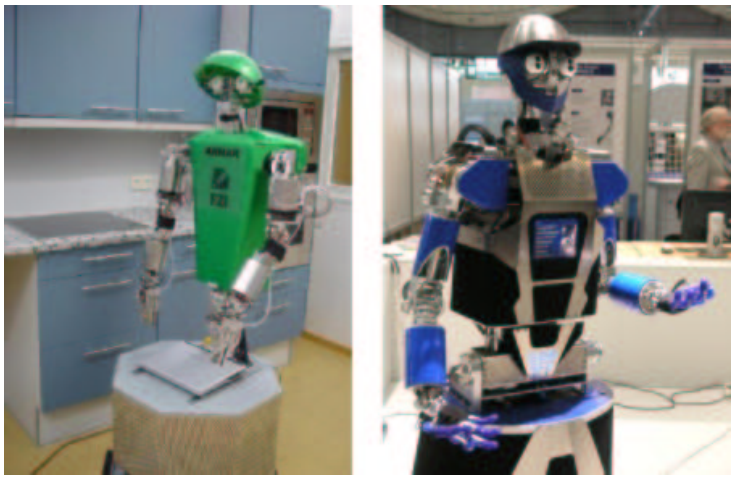


Bild 1.9: Humanoide Robotersysteme des Sonderforschungsbereichs 588 „Lernende und kooperierende multimodale Roboter“: ARMAR-II (links) und ARMAR-III (rechts)

Der humanoide Roboter ISAC vom Intelligent Robotics Lab (IRL) der Vanderbilt University entstand ursprünglich als Forschungsplattform für die Service-Robotik. Er wird mit künstlichen Muskeln [247]

[4] Perzeption: Reizaufnahme durch Sinneszellen oder -organe.

angetrieben und besitzt zwei 6 DOF Arme, zwei anthropomorphe Vier-Finger-Greifer und eine Pan-Tilt-Einheit mit zwei unabhängigen Farbkameras sowie zwei Kraft-Momentensensoren (KS) an den Handgelenken der Arme [142]. Forschungsschwerpunkte dieses Projekts sind hauptsächlich lernbasierte Ansätze [143, 230] sowie die multimodale Roboter-Mensch Kommunikation. Für ISAC wurde zudem eine eigene verhaltensbasierte Architektur namens IMA entwickelt [210, 217].

Zusammenfassend sind in Tabelle 1.1 die wichtigsten in dieser Arbeit vorgestellten humanoiden Roboter aufgeführt und nach Institution, Namen, Anzahl an unabhängigen Freiheitsgraden (DOF), Größe, Gewicht, Fähigkeiten (motorisch und kognitiv) sowie Baujahr unterteilt.

Institution	**Name**	**DOF** [-]	**Größe** [cm]	**Gewicht** [kg]	**Fähigkeiten**	**Jahr**
Honda [116, 236]	P1	-	191	175	LA,TS,GR,MA	1994
	P2	30	182	210	LA,TS,GR,MA	1996
	P3	28	160	130	LA,TS,GR,MA	1997
	ASIMO	26	120	52	LA,TS,GR,MA	1999
Universität Tokyo	Saika	12	-	-	LA,GR	1996
[137, 151, 207]	H5	35	-	-	LA,TS,GR,WI,BW	1998
	H6	35	137	55	LA,TS,GR,WI,BW	2000
	H7	35	137	55	LA,TS,GR,WI,BW	-
Kawada Ind. Inc.	HRP-2	30	154	58	LA,TS,A	1998
[127, 138, 139]					GR,MA	
Sony [98, 165, 166]	SDR-3X	24	50	5	LA,TA	2000
	SDR-4X	38	58	7	LA,TA,SU	2002
Korea Adv. Inst. of	Hubo	41	125	55	LA,GR,MA	2005
Science and Techn. [136]						
TU München [177]	Johnnie	17	180	40	LA	-
Japan Science and	DB	30	185	85	TRO,TE	1996
Technology Corp.					PA,L	
[26, 48, 125]						
Universität Waseda	WABOT-1	-	-	-	LA,GR,TR,K	1970
[114, 140, 141]	WABOT-2	-	-	100	I,K,KEY	1980
Universität Tsukuba	Jack	46	160	60	LE,K,I	2001
[68, 162]						
Massachusetts Inst.	Cog	21	-	-	LE,K,I	1998
of Technology					PE,BE,PA	
[11, 59, 60]						
Bundeswehr Univ.	HERMES	18	-	-	P,TR,MA	1997
München [49, 50]					K,I,(PA)	
SFB-588	ARMAR-II	18	-	45	P,GR,MA	2001
Universität Karlsruhe [76]	ARMAR-III	18	-	150	LE,K,I,PA	2006
Vanderbilt	ISAC	-	-	-	GR,MA	-
University [143, 230]					LE,K,I	

Tabelle 1.1: Vergleichstabelle humanoider Roboter

1.3 Offene Probleme

Zusammenfassend ergeben sich aus den vorherigen Abschnitten für die Regelung und Überwachung von humanoiden Robotersystemen die folgenden offenen Fragestellungen:

1. *Einheitliche Systematik zur Repräsentation von Handlungen:*
   In Abhängigkeit von der verwendeten Roboter-Steuerungsarchitektur existieren unterschiedliche Verfahren zur Repräsentation von Ist- und Soll-Zuständen und Aufgaben sowie daraus resultierenden Handlungen. Möglichkeiten sind beispielsweise die logik-basierte Beschreibung mit Formeln und Operatoren [85, 87] oder die graphenbasierte Beschreibung mit Automaten und Netzen [200]. Eine einheitliche Handlungsrepräsentation für humanoide Robotersysteme muss zudem in der Lage sein, Handlungen mehrerer verschiedener unabhängiger bzw. abhängiger Systemkomponenten (Arme, Greifer, Kopf, Plattform ...) abzubilden und somit parallel und/oder sequentiell zu verarbeiten. Auch dazu existieren bereits einige Arbeiten [22, 24, 279]. Eine durchgängige Entwurfsmethodik hierfür existiert jedoch nicht.

2. *Gewährleistung der Benutzersicherheit bei Interaktion und Kooperation:*
   Für allgemeine technische Systeme existiert eine Vielzahl an theoretisch und praktisch erprobten Sicherheits- und Überwachungskonzepten [93, 135]. Gleichfalls existieren für industrielle Robotersysteme Normen [5], die die Sicherheit von Roboter und Bedienpersonal sicherstellen. Für humanoide Robotersysteme bestehen allerdings zusätzliche Anforderungen an ein Sicherheits- und Überwachungskonzept, da diese Systeme in der Lage sein müssen, in unbekannten, sich ständig ändernden Umgebungen, mit mehreren Bedienern multimodal zu interagieren und zu kooperieren. Weitere Probleme ergeben sich durch parallel vom System zu verarbeitende Aufgaben sowie die Gewährleistung der Kompatibilität zu vorhandenen Systemstrukturen bzw. -architekturen. Aus diesen Anforderungen resultiert die Notwendigkeit eines neuen flexiblen Sicherheits- und Überwachungskonzepts, das in der Lage ist, schnell und flexibel auf zum Teil unbekannte Gefahrensituationen zu reagieren. Ein einheitliches modular anwendbares Konzept hierfür fehlt bislang.

3. *Systematische und einheitliche Ausnahmebehandlungsstrategie:*
   Eine menschenähnliche Behandlung von Ausnahmesituationen verlangt je nach Typ eine einheitliche (z. B. Notaus) oder situationsspezifische Strategie, deren Umsetzung derzeit meist in der Aufgabenplanung erfolgt (z. B. als zusätzliche Stellen in Petri-Netzen mit entsprechenden Transitionen). Dadurch entstehen komplizierte Petri-Netze, die zudem die Koordinierung bei der späteren Fortsetzung von unterbrochenen Aufgaben erschweren. Deswegen ist die vollständige Integration der existierenden Arbeiten zur Erkennung von Ausnahmesituationen (z. B. mit Fuzzy-Regeln und Künstlichen Neuronalen Netzen) in die Steuerungsarchitektur eines humanoiden Roboters ein ungelöstes Problem.

4. *Beherrschung nichtlinearer Systemdynamiken:*
   Bei humanoiden Robotersystemen kommen zunehmend neue innovative Antriebskonzepte und/oder Leichtbaumaterialien zum Einsatz, wodurch eine deutliche Gewichtsreduktion im Vergleich zu Industrierobotern möglich wird [16, 63, 253]. Allerdings kann dieser Leichtbau bei humanoiden Robotersystemen zum Auftreten von Elastizitäten und damit zu Schwingungen in den Achsen und/oder Gelenken führen. In diesem Fall hat die Regelung die Aufgabe, den Einfluss dieser Elastizitäten zu kompensieren oder gezielt auszunutzen und gleichzeitig das geforderte Verhalten zu garantieren. Andere Anwendungsfelder für komplexe Reglerstrukturen bei humanoiden Robotersystemen sind beispielsweise Greifbewegungen oder gekoppelte Greif- und Manipulationsvorgänge. Diese erfordern die koordinierte Regelung mehrerer mechatronischer Komponenten

wie z. B. der Greifer, der Arme, des Kopfes und der Plattform des Robotersystems. Regelungstechnische Standardlösungen für solche komplexen Systeme existieren nicht.

1.4 Zielsetzung der Arbeit

Aus den in Abschnitt 1.3 formulierten offenen Problemen ergeben sich folgende Ziele dieser Arbeit:

1. Entwicklung eines Verfahrens zur systematisierten einheitlichen Beschreibung von Handlungen für humanoide Roboter mit hierarchischen Petri-Netzen als integrierte Teilkomponente einer hybriden Roboter-Steuerungsarchitektur.
2. Bereitstellung und Integration flexibler und modularer modellbasierter Überwachungskomponenten auf allen Ebenen der Roboter-Steuerungsarchitektur.
3. Entwicklung eines neuen Konzepts zur einfachen und systematischen Behandlung von Ausnahmesituationen bei humanoiden Robotern durch eine spezielle Dekomposition von Überwachungsmodulen und deren Kopplung mit geplanten Routinehandlungen(-abläufen).
4. Komponentenspezifische Entwicklung modifizierter modellbasierter Regelungskonzepte zur Beherrschung nichtlinearer Systemdynamiken.

Zu diesem Zweck wird in Kapitel 2 zunächst ein neues ganzheitliches Verfahren zur systematisierten einheitlichen Beschreibung von Handlungen bei humanoiden Robotern vorgestellt. Weiterhin wird ein neues, vollständig in die vorgeschlagene Steuerungsarchitektur integriertes Konzept zur online Überwachung von Bewegungsvorgängen mit Petri-Netzen bereitgestellt und mit einer neuen petrinetzbasierten Methodik zur einfachen und systematischen Behandlung von erkannten Ausnahmesituationen gekoppelt.

Kapitel 3 beschreibt die rechentechnische Umsetzung des in Kapitel 2 vorgestellten Konzepts. Dies beinhaltet einen Überblick über die verwendeten Softwaretools und Modellierungsmethoden. Weiterhin wird ein Verfahren vorgestellt, mit dem sich die entwickelten Algorithmen zur Regelung und Überwachung auf das Zielsystem transferieren lassen.

Im Kapitel 4 erfolgt die Evaluierung des petrinetzbasierten Konzepts zur Generierung von Aufgabenwissen am Beispiel von ausgewählten Roboterkomponenten. Dies beinhaltet die theoretische und experimentelle Modellbildung und den modellbasierten Reglerentwurf für einen fluidisch betriebenen Mehrfinger-Robotergreifer und die anthropomorphe Halseinheit eines humanoiden Roboters.

Kapitel 5 zeigt die Umsetzung des flexiblen, petrinetzbasierten Überwachungskonzepts anhand von Simulationen und Experimenten.

Kapitel 6 fasst die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf zukünftige Entwicklungsmöglichkeiten.

2 Neue Verfahren zur Planung, Ausführung und Überwachung von Roboteraufgaben

2.1 Übersicht

Um mit einem humanoiden Robotersystem benutzer- und/oder umgebungsabhängig abstrakt formulierte Aufgaben in ausführbare Handlungen umzusetzen, ist eine geeignete Roboter-Steuerungsarchitektur erforderlich. In der Literatur existieren dafür zahlreiche Architekturen mit unterschiedlichen Schwerpunkten [149]. Besonders hybride Steuerungsarchitekturen bieten sich wegen der in Abschnitt 1.2.2 genannten Vorteile für den Einsatz in der humanoiden Robotik an. In diesem Kapitel wird gezeigt, wie die aufgabenausführungsrelevanten Komponenten einer hybriden Steuerungsarchitektur (sensomotorische Steuerung) mit Hilfe von hierarchischen Petri-Netzen umgesetzt, integriert (Abschnitt 2.2) und mit Überwachungsmodulen verknüpft werden können (Abschnitt 2.3).

Hierzu wird zunächst ein Konzept zur Generierung von Aufgabenwissen vorgeschlagen, das es erlaubt, 'einfache' Bausteine komplexer Handlungen systematisiert zu entwerfen (Abschnitt 2.2.1). Basierend auf diesem Aufgabenwissen erfolgt danach die Definition einer neuen einheitlichen Systematik zu Beschreibung von komplexen Handlungsabläufen (Pläne zur Lösung von Aufgaben) mit hierarchischen Petri-Netzen. Die Ausführung der auf diese Weise generierten Pläne erfolgt dabei durch die Komponenten der sensomotorischen Steuerung (Abschnitt 2.2.2).

Im Anschluss wird ein neues, vollständig in die vorgeschlagene Steuerungsarchitektur integriertes Konzept zur online Überwachung von Bewegungsvorgängen bei humanoiden Robotern mit Petri-Netzen vorgestellt. Die grundlegende neue Idee besteht dabei darin, die Durchführung der geplanten Aufgabe auf der unteren Ebene der hierarchischen Roboter-Steuerungsarchitektur modellbasiert zu überwachen (Abschnitt 2.3.1) und mit einem neuen petrinetzbasierten Konzept zur einfachen und systematischen Behandlung von erkannten Ausnahmesituationen auf der mittleren Ebene zu koppeln (Abschnitt 2.3.2). Dazu werden zunächst allgemeine und roboterspezifische Anforderungen an ein solches Überwachungskonzept definiert. Danach erfolgt die daraus abgeleitete Umsetzung des neuartigen petrinetzbasierten Lösungskonzepts.

Die gezeigten Beispiele (Abschnitte 2.2.3 und 2.3.3) dienen in diesem Kapitel lediglich zur Veranschaulichung des neuartigen Prinzips. Weitere Beispiele mit detaillierten Ergebnissen von Simulationen und Experimenten werden in den Kapiteln 4 und 5 dargestellt und ausführlich diskutiert. Zum Abschluss des Kapitels werden Vor- und Nachteile des neuen, petrinetzbasierten Konzepts zur systematisierten Planung, koordinierten Ausführung und modellbasierten Sicherheitsüberwachung von Bewegungsvorgängen in humanoiden Robotern diskutiert (Abschnitt 2.4).

2.2 Steuerungsarchitektur

Wegen der in Abschnitt 1.2.2 genannten Anforderungen an ein humanoides Robotersystem wird in dieser Arbeit eine hybride, kognitive Steuerungsarchitektur zugrunde gelegt. Die verwendete Architektur umfasst neben den Komponenten der Perzeption und der sensomotorischen Steuerung auch kognitive Komponenten der Kommunikation (Dialog-Manager), des Lernens sowie der Wissensrepräsentation. Die aufgabenausführungsrelevanten Komponenten der sensomotorische Steuerung (grau hinterlegt) und die Komponenten der Perzeption sind dabei jeweils in der gemäß Literatur weit verbreiteten Drei-Ebenen-Struktur als Planungs-, Koordinierungs- und Ausführungsebenen realisiert [55, 69, 88, 101]. Bild 2.1 zeigt die hybride, kognitive Steuerungsarchitektur des Sonderforschungsbereichs 588 (Karlsruhe) [62].

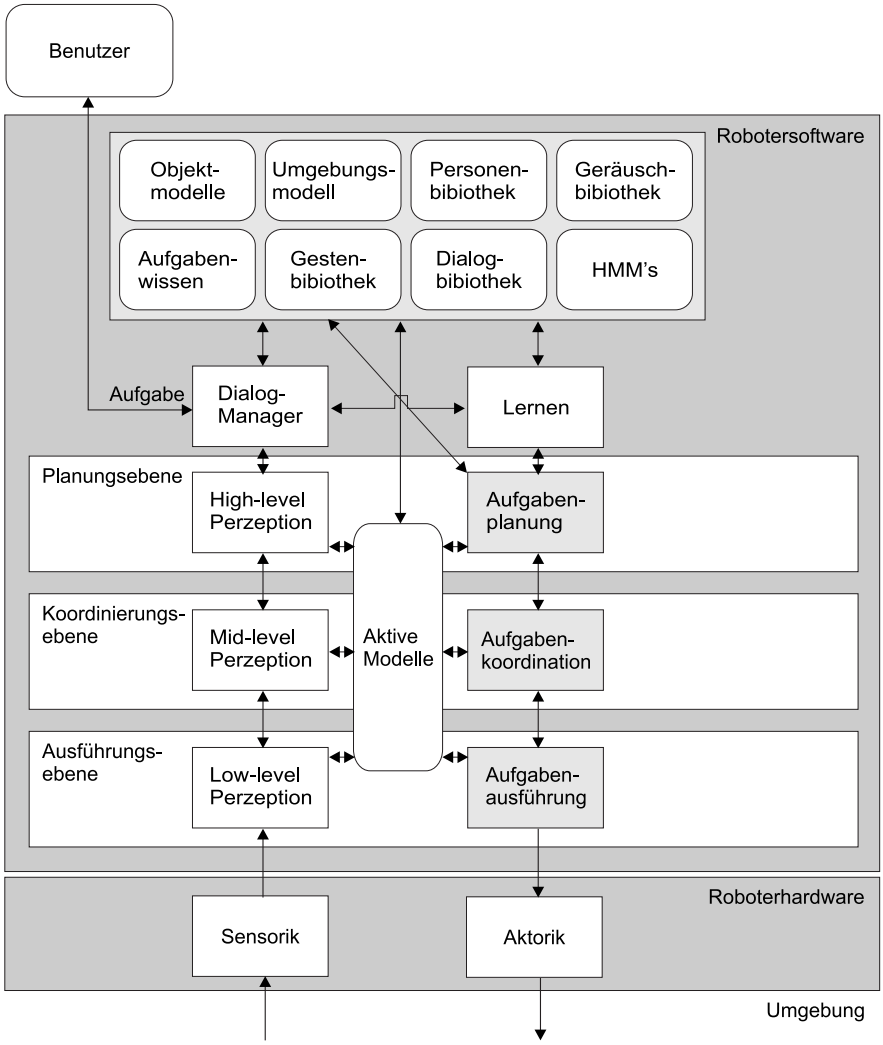


Bild 2.1: Kognitive Steuerungsarchitektur des Sonderforschungsbereichs 588 nach [62]

Da die im Weiteren vorgeschlagene Umsetzung des petrinetzbasierten Konzepts zur systematisierten Planung, koordinierten Ausführung und modellbasierten Sicherheitsüberwachung von Bewegungsvorgängen in humanoiden Robotern in die kognitive Steuerungsarchitektur aus Bild 2.1 zu integrieren ist, erfolgt zunächst eine Beschreibung der grundlegenden Funktionsweise.

Die Repräsentation von Wissen erfolgt einerseits durch eine globale Datenbank. Dort sind Objekte und Bibliotheken ohne Echtzeitanforderung hinterlegt. Beispiele sind Objekt- und Umgebungsmodelle,

Personen-, Gesten-, Geräusch- und Dialogbibliotheken sowie das Aufgabenwissen des Robotersystems. Andererseits benötigen die ausführenden und perzeptiven Komponenten vor allem der mittleren und unteren Ebene der Roboter-Steuerungsarchitektur schnellen Zugriff auf sich während der Ausführung einer Aufgabe ändernde Modellkomponenten. Beispiele sind kinematische und dynamische Modelle von Roboter-Teilsystemen zur modellbasierten Regelung und Überwachung (vgl. Kapitel 4), Szenengraphen der Umgebung [31] oder akustische Umweltkarten [44]. Diese Funktion wird von den so genannten aktiven Modellen übernommen [62]. So werden bei der Ausführung einer Aufgabe die aktiven Modelle zunächst mit dem in der globalen Datenbank hinterlegten Modellwissen initialisiert und dann zur Laufzeit von den perzeptiven Komponenten aktualisiert.

Zur Lösung der vom Benutzer vorgegebenen und vom Dialog-Manager erkannten Aufgabe führt die Aufgabenplanung (engl.: task planning), im gegenwärtigen Entwicklungsstand lediglich eine vom Entwickler in geeigneter Form in der globalen Datenbank hinterlegte, fest zugeordnete Sequenz von Handlungen (Plan) aus (durchgezogene Blöcke Bild 2.2). Verfügt die Aufgabenplanung dagegen über autonome Planungskomponenten (gestrichelte Blöcke Bild 2.2), ist das System in der Lage, selbstständig Pläne zu generieren. Dazu wird der durch die perzeptiven Komponenten der Roboter-Steuerungsarchitektur gegebene aktuelle Istzustand des Systems, unter Verwendung von Optimierungs- und/oder Lernalgorithmen [40, 191], in den durch das Aufgabenziel definierten finalen Sollzustand überführt (vgl. Abschnitt 1.2.2). Probleme ergeben sich, wenn bestimmte Informationen erst zur Zeit der Planausführung verfügbar sind oder bei fehlender sensorischer Erfassung ganz fehlen. Eine mögliche Lösung zur Verbesserung des Plans ist die modellbasierte Schätzung des zukünftigen Systemzustands mit Hilfe der aktiven Modelle. Bild 2.2 zeigt eine Variante der Aufgabenplanung in der Detailansicht. Weitere hier nicht dargestellte Varianten sind durchaus denkbar [62].

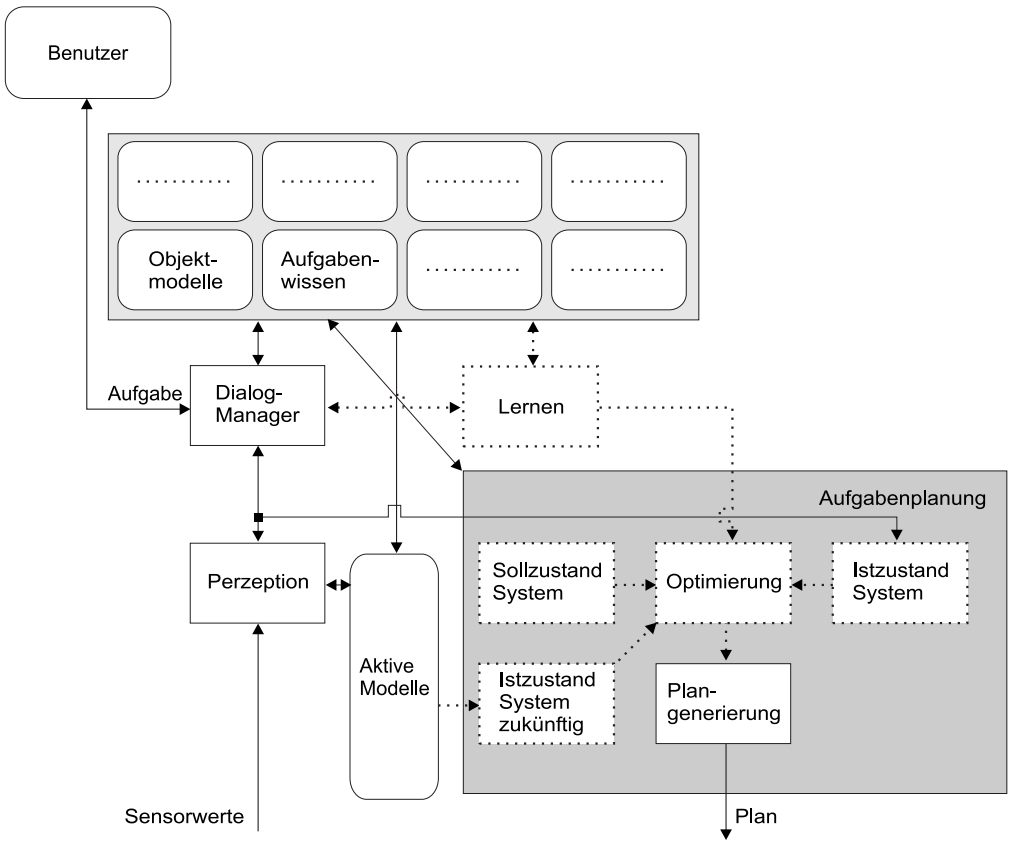


Bild 2.2: Detailansicht einer möglichen Variante der Aufgabenplanung

Die von der Aufgabenplanung generierten Handlungsabläufe werden in den Aufgabenspeicher (engl.: task stack) auf der mittleren Hierarchieebene weitergeleitet. In der Aufgabenkoordination erfolgt dann unter Berücksichtigung von Prioritäten, aufgabenspezifischen Restriktionen und verfügbaren Hardware- und Softwareressourcen die synchronisierte Ausführung der im Aufgabenspeicher abgelegten Handlungen durch die Prioritätsverwaltung. So wird die Ausführung des generierten Plans nur freigegeben, wenn dabei keinerlei Software- bzw. Hardwarekonflikte auftreten. Bild 2.3 zeigt die Detailansicht der Aufgabenkoordination als Ausschnitt aus der gesamten Steuerungsarchitektur.

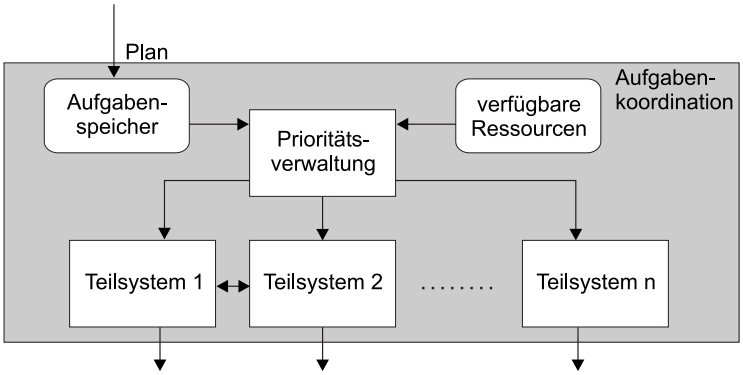


Bild 2.3: Detailansicht der Aufgabenkoordination als Ausschnitt aus der gesamten Steuerungsarchitektur

Auf der unteren Hierarchieebene befindet sich die Aufgabenausführung (engl.: task execution), die alle zugehörigen Funktionen mit teilsystemspezifischen Einstellungen für die Referenztrajektoriengenerierung und die Auswahl bestimmter Reglerstrukturen und -parameter enthält. Mit den ausgewählten und parametrierten Reglern erfolgt dann die Ansteuerung der Aktorik des entsprechenden Roboter-Teilsystems. Durch die Rückführung der mit der integrierten Sensorik gemessenen Regelgrößen und anschließender Verarbeitung durch die perzeptiven Komponenten der Steuerungsarchitektur (vgl. Bild 2.1) wird der Regelkreis geschlossen. Bild 2.4 zeigt die Detailansicht der Aufgabenausführung als Ausschnitt aus der gesamten Steuerungsarchitektur.

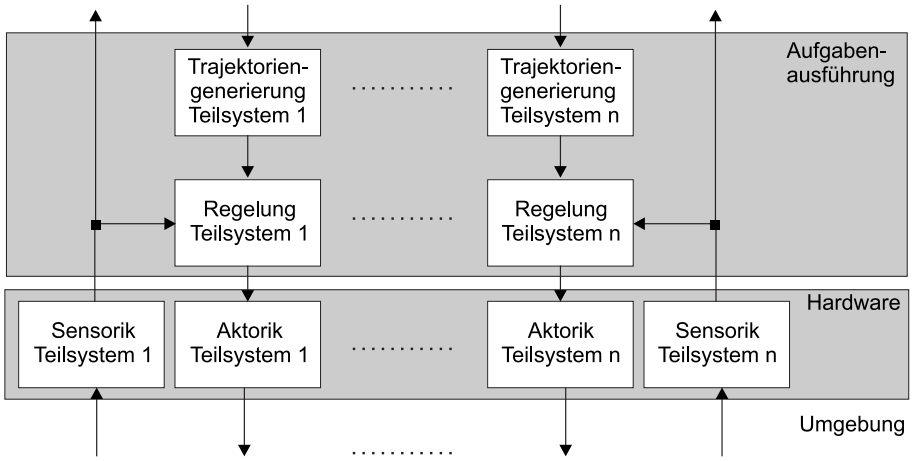


Bild 2.4: Detailansicht der Aufgabenausführung als Ausschnitt aus der gesamten Steuerungsarchitektur

Die Schwerpunkte der Arbeit liegen dabei in der koordinierten Durchführung von Bewegungsvorgängen und der Generierung von dazu benötigtem Aufgabenwissen (mittlere und unteren Ebene der Roboter-Steuerungsarchitektur, Bilder 2.3 und 2.4). Die autonome Umsetzung des generierten Aufgabenwissens zu ausführbaren Plänen (obere Ebene der Roboter-Steuerungsarchitektur, Bild 2.2) ist derzeit Gegenstand der Forschungsarbeiten im Sonderforschungsbereich 588 [120, 219]. Durch die Modularität des hier vorgestellten Konzepts sind diese Algorithmen bei Verfügbarkeit jedoch jederzeit integrierbar.

2.2.1 Modellbasierte Generierung von Aufgabenwissen

Eine Roboteraufgabe ($\mathbb{T}$, engl: task) lässt sich allgemein durch eine geeignete Sequenz von Handlungen (H) beschreiben, die die derzeitige Situation (Istzustand S_{Ist}) des Robotersystems in eine gewünschte Situation (Sollzustand S_{Soll}) überführt:

$$\mathbb{T}: \; S_{\text{Ist}}[k_0] \underset{\text{H}}{\rightarrow} S_{\text{Soll}}[k] \quad \text{mit} \quad k > k_0. \tag{2.1}$$

Handlungsabläufe zur Ausführung von Aufgaben lassen sich entweder manuell durch den Entwickler oder in einer weiteren Entwicklungsstufe autonom durch die Aufgabenplanung des Robotersystems generieren (vgl. Abschnitt 2.2). Bei beiden Vorgehensweisen sind zunächst abstrakt formulierte Aufgaben in konkrete Handlungsanweisungen zu übersetzen. So wird beispielsweise die vom Benutzer an das Robotersystem kommandierte Aufgabe ”Schaue zur Tür” von der Aufgabenplanung unter Berücksichtigung der derzeitigen Blickrichtung (Istzustand S_{Ist}) in eine interne Handlungsanweisung transferiert. Einem humanoiden Robotersystem stehen dabei häufig mehrere Möglichkeiten zur Lösung einer Aufgabe zur Verfügung. So kann ein Robotersystem in dem hier beschriebenen Beispiel entweder nur den Kopf (”Drehe Kopf bis Blick Richtung Tür”) nur die Plattform (”Drehe Plattform bis Blick Richtung Tür”) oder beide Roboterkomponenten (”Drehe Kopf und Plattform bis Blick Richtung Tür”) ansteuern, bis die gewünschte Sollposition (Sollzustand S_{Soll}) erreicht ist. Hat sich die Aufgabenplanung oder der Benutzer auf einen möglichen Lösungsweg festgelegt, muss die entsprechende Handlungsanweisung in regelungstechnisches Aufgabenwissen (z. B. ”Drehe Kopf 30° nach links” mit entsprechenden Führungsgrößentrajektorien und Regelungsalgorithmen inklusive Parametrierung) auf der Roboter-Steuerungsarchitektur umgesetzt und geeignet in der globalen Datenbank des Robotersystems hinterlegt werden. So benötigt die regelungstechnische Umsetzung der Handlungsanweisung ”Drehe Kopf 30° nach links” zur erfolgreichen Durchführung beispielsweise Funktionen zur Generierung von rampen- oder sprungförmigen Führungsgrößentrajektorien und zur Positionsregelung der Freiheitsgrade der Halseinheit.

Im Weiteren wird nun eine diskret-kontinuierliche Systemstruktur vorgestellt (Abschnitt 2.2.1.1), die es ermöglicht, Bausteine komplexer Handlungen (H) modellbasiert zu entwerfen, im Aufgabenwissen des Roboters zu speichern (Abschnitt 2.2.1.2) und auf der Roboter-Steuerungsarchitektur auszuführen (Abschnitt 2.2.2).

2.2.1.1 Diskret-kontinuierliche Systemstruktur

Da die verwendete Roboter-Steuerungsarchitektur eng gekoppelte, ereignisdiskrete (Handlungsabläufe auf der Planungsebene) und kontinuierliche (Führungsgrößentrajektorien und Stellgrößen der Regelung auf der Ausführungsebene) Bestandteile enthält, ist sie den diskret-kontinuierlichen Systemen zuzuordnen [118]. Zur systematisierten Beschreibung solcher diskret-kontinuierlichen Systeme existieren zahlreiche Arbeiten, welche u. A. im Rahmen des KONDISK-Schwerpunktprogramms entstanden sind [157, 158, 239]. In [204] wird beispielsweise ein Netz-Zustands-Modell vorgestellt, das in der Lage ist, die in diskret-kontinuierlichen Systemen auftretenden Umschaltvorgänge abzubilden. Es besteht

aus einem Petri-Netz und einem erweiterten Zustandsraummodell. Bekannte Ansätze zur Modellierung diskret-kontinuierlicher Systeme in der Robotik sind u. A. das Konzept der Aktionsprimitive [86, 198] sowie ein in [24] erstmals vorgestelltes Konzept zur Koordinierung von Roboterkomponenten mit Petri-Netzen. Die dargestellten Ansätze bilden die theoretische Grundlage der in dieser Arbeit verwendeten Systemstruktur. Neu sind dagegen die systematisierte Beschreibung komplexer ereignisdiskreter Roboterhandlungen mit Hilfe von hierarchischen Petri-Netzen (Abschnitt 2.2.2.1), die Verwaltung von Roboterkomponenten durch ein petrinetzbasiertes Ressourcenmanagement (Abschnitt 2.2.2.2) sowie die Kopplung der diskret-kontinuierlichen Systemstruktur mit modellbasierten Überwachungskomponenten zur flexiblen Sicherheitsüberwachung des gesamten Robotersystems (Abschnitt 2.3).

Allen hier beschriebenen Ansätzen gemein ist die Verwendung von Petri-Netzen zur Beschreibung des ereignisdiskreten Teils des betrachteten Systems. Als Petri-Netz wird in dieser Arbeit dabei ganz allgemein eine Methode bezeichnet, um Prozesse mit wertediskreten Zuständen aller Art abstrakt darzustellen. Definiert wird ein Petri-Netz durch ein 6-Tupel ($\mathcal{A}$, $\mathcal{T}$, $\mathcal{F}$, $\mathcal{K}$, $\mathcal{W}$, $\mathcal{M}_0$), dessen Bestandteile Plätze ($\mathcal{A}$) mit Kapazitäten ($\mathcal{K}$) und Markenbelegungen ($\mathcal{M}$) (gegeben anfängliche Markenbelegung $\mathcal{M}_0$), Transitionen ($\mathcal{T}$) und Kanten ($\mathcal{F}$) mit Kantengewichten ($\mathcal{W}$) sind [221]:

$$\mathcal{A} = \{A_1, A_2, \ldots, A_{|\mathcal{A}|}\}, \tag{2.2}$$

$$\mathcal{T} = \{T_1, T_2, \ldots, T_{|\mathcal{T}|}\}, \tag{2.3}$$

$$\mathcal{F} \subseteq (\mathcal{A} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{A}), \tag{2.4}$$

$$\mathcal{K} : \mathcal{A} \rightarrow \mathbb{N}\backslash\{0\}, \tag{2.5}$$

$$\mathcal{W} : \mathcal{F} \rightarrow \mathbb{N}\backslash\{0\}, \tag{2.6}$$

$$\mathcal{M}_0 : \mathcal{A} \rightarrow \mathbb{N}. \tag{2.7}$$

Die Darstellung (Repräsentation) erfolgt mit Hilfe grafischer Symbole (Plätze als Kreise, Transitionen als Rechtecke, Kanten als Pfeile). Um den Zusammenhang zwischen zu beschreibendem Prozess und entsprechender Petri-Netz-Repräsentation herzustellen, werden bei steuerungstechnisch interpretierten Petri-Netzen auftretende Ereignisse als Transitionen, Bedingungen als Plätze und die Verknüpfung von Ereignissen und Bedingungen als Kanten modelliert. Weiterhin werden Regeln benötigt, die den Prozessfortschritt im Netz beschreiben. Dieser Vorgang wird durch das 'Schalten' nachgebildet. Das Schalten verändert die Markenbelegung im Netz. Nur eine einzige Transition kann zeitgleich schalten. Dabei werden so viele Marken von jedem Platz des Vorbereichs wie das Kantengewicht angibt an die Plätze des Nachbereichs übergeben [221]. Weiterführende Grundlagen zu Petri-Netzen sind im Anhang A zu finden. Vorteile von Petri-Netzen sind in der systematischen Entwurfsmethodik, der weit entwickelten Petri-Netz-Theorie sowie den daraus resultierenden vielfältigen Analysemöglichkeiten (Erreichbarkeits-, Deadlock- und Lebendigkeitsanalyse, ...) zu sehen [10, 261].

Zur systematischen Beschreibung von Handlungen (H) werden hier Elementaroperationen EO_a eingeführt. Mit den als diskrete Zustände (Plätze) $a = 1, \ldots, |A|$ in Petri-Netzen ausgeführten Elementaroperationen

$$\mathcal{A} = \{EO_1, EO_2, \ldots, EO_{|A|}\} \tag{2.8}$$

sind kontinuierliche sensomotorische Operationen für *einen* einzelnen Freiheitsgrad $j = 1, \ldots, m_i$ (z. B. einen unterlagerten Regelkreis) einer Roboter-Funktionaleinheit[1] i verknüpft. Dazu sind, entsprechend der gewünschten Funktionalität der ausgewählten Elementaroperation, Funktionen zur Trajektoriengenerierung und Regelung zu entwerfen und zu implementieren (f_a^{Tg}, $\boldsymbol{f}_a^R$ und g_a^R in Bild 2.5). Durch

[1] Eine Roboter-Funktionaleinheit fasst alle unabhängigen Freiheitsgrade (DOF) eines Roboter-Teilsystems zusammen, die funktional gekoppelt sind. Dies beinhaltet sowohl komplette Roboter-Teilsysteme wie z. B. Plattform, Arm oder Hals als auch unterlagerte Sub-Teilsysteme wie einzelne Finger eines Robotergreifers.

Markierung

$$\mathcal{M} = \{(EO_1, \mathcal{M}_1), (EO_2, \mathcal{M}_2), \ldots, (EO_{|A|}, \mathcal{M}_{|A|})\} \quad \text{mit} \quad \mathcal{M}_a \in [0,1] \tag{2.9}$$

einer Elementaroperation (EO_a), d. h. eines diskreten Platzes des die Handlung beschreibenden Petri-Netzes, werden die entsprechenden Funktionen zur Trajektoriengenerierung und Regelung aktiviert und mit dem Vektor $\boldsymbol{v}_{a,(i,j)}[k]$ parametriert. Der Vektor $\boldsymbol{v}_{a,(i,j)}[k]$ enthält dabei die notwendigen Informationen zur vollständigen Definition der Trajektorie (Trajektorienart, z. B. Rampe, Sprung usw., mit dazugehörenden Trajektorienparametern sowie u. U. notwendige kinematische Transformationsvorschriften) und des Reglers (Reglerstruktur z. B. linearer Regler, Kaskadenregler, Impedanzregler usw., sowie dazugehörende Reglerparameter). Die meisten der Parameter sind für die gesamte Ausführungszeit der Elementaroperation (EO_a) konstant $\boldsymbol{v}_{a,(i,j)}[k] = \boldsymbol{v}_{a,(i,j)} = \text{const}$. Um allerdings parameteradaptive Regler (z. B. zur Gravitations- oder Reibungskompensation) oder modellbasiert geschätzte Führungs- und Störgrößenaufschaltungen in die vorgeschlagene Systemstruktur zu integrieren, kann der Vektor $\boldsymbol{v}_{a,(i,j)}[k]$ optional für spezielle Plätze a zur Laufzeit der Elementaroperation (EO_a) von den aktiven Modellen der Steuerungsarchitektur aktualisiert werden (vgl. Bild 2.1). Als resultierende Beschreibung ergibt sich:

$$w_{(i,j)}[k] = f_a^{Tg}(\boldsymbol{v}_{a,(i,j)}[k]), \tag{2.10}$$

$$\boldsymbol{x}^R_{(i,j)}[k+1] = \boldsymbol{f}_a^R(w_{(i,j)}[k], \boldsymbol{x}^R_{(i,j)}[k], \boldsymbol{y}_i[k], \boldsymbol{v}_{a,(i,j)}[k]), \tag{2.11}$$

$$u_{(i,j)}[k] = g_a^R(\boldsymbol{x}^R_{(i,j)}[k], \boldsymbol{v}_{a,(i,j)}[k]). \tag{2.12}$$

Darin bezeichnen $w_{(i,j)}$ die Führungsgrößentrajektorie, $u_{(i,j)}$ die Stellgrößentrajektorie[2], $\boldsymbol{x}^R_{(i,j)}$ den Zustandsvektor der Regelung[3] und $\boldsymbol{y}_i$ die Regelgrößen[4] der entsprechenden Roboter-Funktionaleinheit. Häufig wird nur die skalare Regelgröße $y_{(i,j)}$ zur Regelung verwendet. Sind allerdings in einem Freiheitsgrad mehrere Regelgrößen z. B. Kraft und Position zu regeln, dann wird aus der skalaren Regelgröße der in Gleichung (2.11) dargestellte Regelgrößenvektor $\boldsymbol{y}_i$. Die Aktivierung von Transitionen des die Handlung beschreibenden Petri-Netzes erfolgt entweder direkt auf der untersten Ebene der Roboter-Steuerungsarchitektur durch gemessene bzw. vorverarbeitete und/oder fusionierte Sensorsignale (z. B. Objektkontakt ja/nein) oder durch den Eingriff einer übergeordnete Planungs- und/oder Verwaltungskomponente. Ist die Elementaroperation (EO_a) einer Handlung (H) nicht mit einer Marke belegt, ist der unterlagerte Regelkreis inaktiv und die entsprechenden Funktionen zur Trajektoriengenerierung und Regelung werden nicht ausgeführt. Bild 2.5 zeigt beispielhaft die beschriebenen Zusammenhänge für eine Handlung mit $|A| = 2$ Elementaroperationen[5].

Als problematisch können sich Umschaltvorgänge innerhalb von diskret-kontinuierlichen Systemen erweisen. Das autonome oder von außen gesteuerte Auslösen eines Ereignisses führt dabei zu einer Umschaltung der Dynamik des Systems [56]. So erfordert beispielsweise die Strukturumschaltung des Streckenmodells beim Greifen eines Robotergreifers durch Objektkontakt eine Reglerumschaltung (Umschalten vom positions- in den kraftgeregelten Modus [170]). Weiterhin kann es sinnvoll sein, einen Regler auf Handbetrieb, bzw. vom Handbetrieb wieder zurück auf den automatischen Betrieb umzuschalten (Hand-Automatik-Umschaltung [225]). Sowohl bei Umschaltvorgängen zwischen verschiedenen Reglern, als auch bei der Hand-Automatik-Umschaltung ergibt sich die Forderung nach Stoßfreiheit (engl.: Bumpless Transfer). Vorteile ergeben sich dabei z. B. durch Schonung der Aktorik des Systems.

[2] Die normal gedruckten und mit (i,j) indizierten Variablen bezeichnen Skalare.

[3] Die fett gedruckten und mit (i,j) indizierten Variablen bezeichnen Vektoren der Dimension $(m_x \times 1)$ mit m_x als Anzahl interner Systemzustände.

[4] Die fett gedruckten und mit i indizierten Variablen bezeichnen Vektoren der Dimension $(n_y \cdot m_i \times 1)$ mit n_y als Anzahl der Regelgrößen in einem Mehrgrößensystem.

[5] z^{-1} in Bild 2.5 bezeichnet die Verzögerung um einen Abtastzeitpunkt ta.

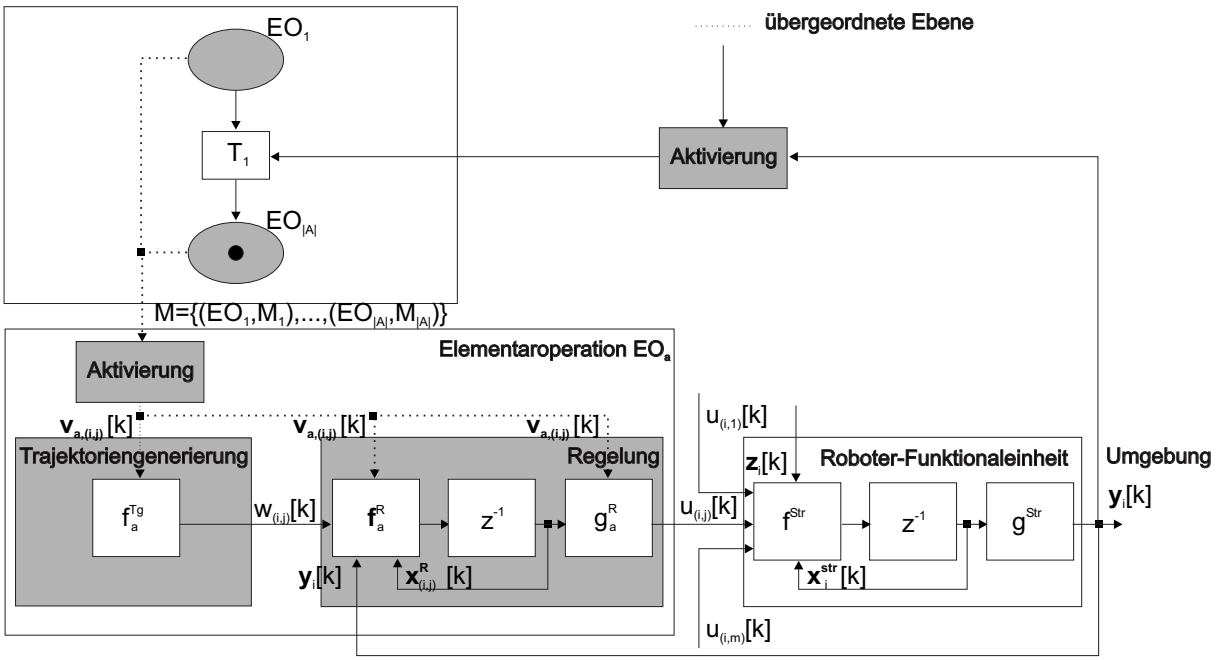


Bild 2.5: Detailansicht der diskret-kontinuierlichen Systemstruktur für Elementaroperationen (EO)

Eine in der Literatur bekannte Maßnahme zur Gewährleistung einer stoßfreien Umschaltung ist die Neuinitialisierung des Integrators zum Umschaltzeitpunkt [225]. Allerdings können auch bei stoßfreien Umschaltvorgängen sprunghafte Änderungen bei Führungsgrößen auftreten (’harte’ Umschaltung). Diese Sprünge können zu Stabilitätsproblemen des gesamten Systems führen. Eine mögliche Lösung ist die Realisierung eines ’weichen’ Überblendvorgangs beispielsweise durch den Einsatz eines überlagerten Fuzzy-Mode-Selectors [105, 163].

2.2.1.2 Elementaraktionen zur Ausführung von Roboteraufgaben

Probleme entstehen, wenn zur Ausführung einer Handlung H dynamisch gekoppelte Roboter-Teilkomponenten mit mehreren (> 1) Freiheitsgraden angesteuert werden sollen. Werden dann die Funktionen zur Trajektoriengenerierung und Regelung (f_a^{Tg}, $\boldsymbol{f}_a^R$ und g_a^R) wie im vorigen Abschnitt vorgeschlagen, unabhängig voneinander entworfen, bleibt die Dynamik des gekoppelten Systems unberücksichtigt, was wiederum zu einem nicht tolerierbaren Systemverhalten führen kann. Somit sind Elementaroperationen nur zur ungekoppelten Ansteuerung eines einzelnen Freiheitsgrades in Form eines unterlagerten Regelkreises sinnvoll einzusetzen. Ein Beispiel für ein solches System ist das Druckregelventil aus Abschnitt 4.2.1 mit integriertem unterlagertem Druckregelkreis. Aus diesem Grund wird das Konzept der Elementaroperationen verallgemeinert. Zunächst werden dazu *alle* einen diskreten Zustand u einer Handlung repräsentierende Elementaroperationen einer Roboter-Funktionaleinheit i mit $j = 1, \ldots, m_i$ Freiheitsgraden zu so genannten Elementaraktionen $EA_{i,u}$ zusammengefasst

$$EA_{i,u} = \{EO_{(i,1)u}, EO_{(i,2)u}, \ldots, EO_{(i,m_i)u}\}. \tag{2.13}$$

Für die Plätze $u = 1, \ldots, |U|$ des die Handlung beschreibenden Petri-Netzes folgt damit:

$$\mathcal{A} = \{EA_{i,1}, EA_{i,2}, \ldots, EA_{i,|U|}\}. \tag{2.14}$$

Bei Markierung

$$\mathcal{M} = \{(EA_{i,1}, \mathcal{M}_{i,1}), (EA_{i,2}, \mathcal{M}_{i,2}), \ldots, (EA_{i,|U|}, \mathcal{M}_{i,|U|})\} \quad \text{mit} \quad \mathcal{M}_{i,u} \in [0,1] \tag{2.15}$$

einer Elementaraktion ($EA_{i,u}$) werden wiederum die entsprechenden Funktionen zur Trajektoriengenerierung und Regelung aktiviert und mit dem Vektor $\boldsymbol{v}_{u,i}[k]$ parametriert. Die resultierende Modellstruktur fasst jetzt allerdings *alle* Freiheitsgrade $j = 1, \ldots, m_i$ einer Roboter-Funktionaleinheit i zusammen, die sich im gleichen diskreten Zustand u befinden:

$$\boldsymbol{w}_i[k] = \boldsymbol{f}_u^{Tg}(\boldsymbol{v}_{u,i}[k]), \tag{2.16}$$

$$\boldsymbol{x}_i^R[k+1] = \boldsymbol{f}_u^R(\boldsymbol{x}_i^R[k], \boldsymbol{w}_i[k], \boldsymbol{y}_i[k], \boldsymbol{v}_{u,i}[k]), \tag{2.17}$$

$$\boldsymbol{u}_i[k] = \boldsymbol{g}_u^R(\boldsymbol{x}_i^R[k], \boldsymbol{v}_{u,i}[k]). \tag{2.18}$$

Neben den Modellen für die Trajektoriengenerierung und die Regelung sind zudem die meisten Roboter-Funktionaleinheiten in Form von nichtlinearen Differenzengleichungen im Zustandsraum zu beschreiben[6] [197]:

$$\boldsymbol{x}_i^{Str}[k+1] = \boldsymbol{f}^{Str}(\boldsymbol{u}_i[k], \boldsymbol{x}_i^{Str}[k], \boldsymbol{z}_i[k]), \tag{2.19}$$

$$\boldsymbol{y}_i[k] = \boldsymbol{g}^{Str}(\boldsymbol{x}_i^{Str}[k], \boldsymbol{z}_i[k]). \tag{2.20}$$

Bild 2.6 zeigt beispielhaft die resultierende diskret-kontinuierliche Systemstruktur für eine Roboter-Funktionaleinheit i mit $j = 2$ Freiheitsgraden und einer Handlung (H) mit $u = 2$ diskreten Zuständen.

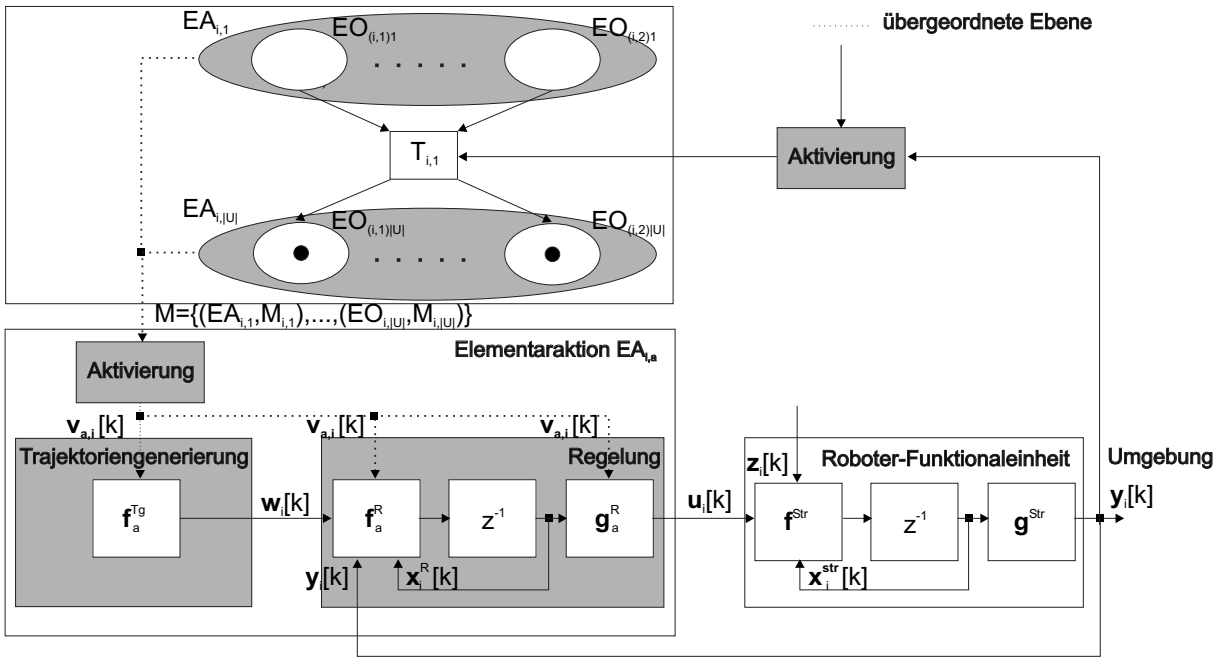


Bild 2.6: Detailansicht der resultierenden diskret-kontinuierlichen Systemstruktur für Elementaraktionen (EA)

[6] Die Zustandsgrößen von Regler $\boldsymbol{x}^R$ und Strecke $\boldsymbol{x}^{Str}$ sind hier Vektoren der Dimension $\sum_{i=1}^{m_i} m_{x,i}$.

Die auf diese Weise definierten Elementaraktionen lassen sich zu Klassen zusammenfassen, die die gleiche Trajektorienart und das gleiche Regelungsprinzip verwenden. Somit müssen den allgemein definierten Funktionen zur Trajektoriengenerierung und Regelung beim Entwurf nur noch die entsprechenden klassenspezifischen Parameter (Vektor $\boldsymbol{v}_{u,i}[k]$) übergeben werden.

Tabelle 2.1 zeigt einen Überblick ausgewählter Elemente als Elementaraktionen mit verwendeter Trajektorienart, benötigtem Regelungsprinzip, angesteuerter Roboter-Funktionaleinheit sowie dazugehörenden Roboteraufgaben.

EA	Trajektorienart	Regelungsprinzip	Funktionaleinheit	Beispiele
Fahren	Rampe, Sprung menschl. Bewegung	Positions-, Nachgiebigkeits-regelung (Folge)	Arme, Greifer, Plattform, Kopf	Winken, Nicken Zeigegesten
Warten	Sollwert konstant	Positions-, Nachgiebigkeits-regelung (Festwert)	Arme, Greifer, Plattform, Kopf	Koordination von Greifbewegungen
Suchen	Rampe alternierend menschl. Bewegung	Positionsregelung (Folge)	Kopf	Aufmerksamkeits-steuerung
Verfolgen	-	Visual-Servoing	Kopf	Aufmerksamkeits-steuerung
Zugreifen	Rampe, Sprung menschl. Bewegung	Kraft-, Nachgiebigkeits-regelung (Folge)	Greifer	Greifbewegungen
Halten	Sollwert konstant	Kraft-, Nachgiebigkeits-regelung (Festwert)	Arme, Greifer	Transport-bewegungen
Führen	Vorgegeben (Benutzer)	Nullkraftregelung	Arme	Kooperation Roboter/Benutzer

Tabelle 2.1: Beispiele für Elementaraktionen (EA) von humanoiden Robotern

Ein weiterer Vorteil des vorgestellten Konzepts ist, dass bereits existierende Algorithmen zur Ausführung von Handlungen (H), die nicht mit dem vorliegenden Schema entworfen worden sind, leicht in die hier beschriebene Form zu überführen sind. Dazu sind lediglich die vorhandenen Funktionen zur Trajektoriengenerierung und Regelung als diskrete Zustände in einem Petri-Netz (d. h. als Elementaraktionen EA) zusammenzufassen. Die auf diese Weise definierten Elementaraktionen werden in der globalen Datenbank der Roboter-Steuerungsarchitektur gespeichert und bilden den auf *Bewegungsvorgänge* fokusierten Teil des Aufgabenwissens des Robotersystems. Neben den hier beschriebenen Elementaraktionen zur planmäßigen Ausführung von Aufgaben können mit dem beschriebenen Verfahren weiterhin Ausnahmebehandlungsstrategien erzeugt und im Aufgabenwissen gespeichert werden. Die auf diese Weise generierten Elementaraktionen werden dann von der petrinetzbasierten Überwachungskomponente der Roboter-Steuerungsarchitektur zur autonomen Problemlösung eingesetzt (vgl. Abschnitt 2.3.2).

Ausnahmen bilden Roboter-Teilsysteme, die aufgrund ihrer Systemeigenschaften, bedingt z. B. durch extremen Leichtbau und daraus resultierenden Elastizitäten (heutzutage wichtiges Entwicklungsziel vor allem in der humanoiden Robotik) [170] oder der Anwendung neuartiger physikalischer Wirkprinzipien (z. B. fluidische Aktoren [249], pneumatische Muskeln [84, 150], ...) eine deutlich komplexere, häufig modellbasierte Regelung erfordern. Bei diesen Komponenten kann beim Entwurf daher nicht auf standardisierte Bibliotheksfunktionen zur Trajektoriengenerierung und zur Regelung zurückgegriffen werden. Stattdessen wird durch Ableiten der Systemgleichungen aus physikalischen Grundgesetzen (theoretische Modellbildung) und Vergleich mit Messungen am realen System (experimentelle Modellbildung) [130, 226, 267] zunächst ein mathematisches Modell der entsprechenden Roboterkomponente aufgestellt, auf dessen Basis dann der Entwurf und die Evaluierung der Regelung erfolgt (vgl. Kapitel 4).

2.2.2 Petrinetzbasierte Ausführung von Roboteraufgaben

Im Weiteren wird nun das in den vorigen Abschnitten generierte Aufgabenwissen in Form von Elementaraktionen zur petrinetzbasierten Planung (Abschnitt 2.2.2.1) und koordinierten Ausführung (Abschnitte 2.2.2.2 und 2.2.2.3) von komplexen Handlungen (H) auf der Roboter-Steuerungsarchitektur verwendet.

2.2.2.1 Aufgabenplanung mit hierarchisch strukturierten Petri-Netzen

Die autonome Planung von Handlungsabläufen zur erfolgreichen Lösung von komplexen Aufgaben gehört zu den wichtigsten Fähigkeiten einer kognitiven Roboter-Steuerungsarchitektur. Probleme ergeben sich, da Robotersysteme nur ein unvollständiges Wissen über ihre Umwelt besitzen. Ein autonomer Roboter kann nur soviel über die Welt wissen, wie er über seine Sensoren wahrnimmt. Die dem Roboter zur Verfügung stehenden Pläne können daher unvollständig sein, wenn bestimmte Informationen erst zur Zeit der Planausführung verfügbar sind oder bei fehlender sensorischer Erfassung ganz fehlen. In der Literatur existieren zahlreiche Konzepte zur Abbildung von Handlungsabläufen. Diese lassen sich prinzipiell in die Bereiche autonome Planerstellung [40, 190, 191, 235, 270, 281] und manuelles Teaching [22, 79, 83] unterteilen. Um autonom Handlungsabläufe zu erstellen, muss das Robotersystem über kognitive Fähigkeiten und damit eine kognitive Architektur (vgl. Abschnitt 1.2.4) verfügen. Derzeitiger Stand der Entwicklung in realen humanoiden Robotersystemen ist jedoch das manuelle (oder bestenfalls halbautonome) Speichern von Handlungsabläufen im Aufgabenwissen des Robotersystems durch den Entwickler (z. B. mit Programmieren durch Vormachen [79, 83]). Die auf diese Weise generierten Pläne werden dann von den aufgabenausführenden Komponenten des Robotersystems abgearbeitet.

Die in dieser Arbeit vorgestellte Aufgabenplanung baut auf dem in den vorigen Abschnitten generierten Aufgabenwissen in Form von Elementaraktionen EA auf. Die Darstellung von u. U. sehr komplexen Handlungen (H) erfordert allerdings eine Erweiterung des vorgestellten Konzepts. Dabei erfolgt die Systematisierung der Beschreibung von abstrakt formulierten, komplexen Handlungen durch Gliederung in übergeordnete Abstraktionseinheiten und Zuordnung zu entsprechend hierarchisch strukturierten Petri-Netzen (Modularisierung). Ausgehend von Elementaroperationen EO, wurden in Abschnitt 2.2.1.2 Elementaraktionen $EA_{i,u}$ (engl.: elementary action) definiert, die für alle $j = 1, \ldots, m_i$ Freiheitsgrade der betreffenden Roboter-Funktionaleinheit i einen diskreten Zustand u umfassen. Elementaraktionen $EA_{i,u}$ entsprechen somit den zusammengefassten Freiheitsgraden einer Roboter-Funktionaleinheit i, die die gleichen Anforderungen an die Trajektoriengenerierung und Regelung haben. Parallel, sequentiell oder vernetzt strukturierte Elementaraktionen $EA_{i,u}$ können zu Aktionen A (engl.: action) zusammengefasst werden. Aktionsfolgen AF können wiederum aus mehreren, beliebig (parallel, sequentiell oder vernetzt) strukturierten Aktionen A zusammengesetzt sein. Aktionsfolgen AF bilden dabei die oberste in dieser Arbeit definierte Abstraktionseinheit. Um mit dem hier vorgestellten Konzept auch komplexere Handlungen, die zusätzliche Hierarchieebenen benötigen, darstellen zu können, ist es zudem zulässig Aktionsfolgen beliebig strukturiert in weitere Aktionsfolgen zu gliedern. Weiterhin kann eine Elementaraktion auch nur aus einer Elementaroperation, eine Aktion nur aus einer Elementaraktion und eine Aktionsfolge nur aus einer Aktion bestehen. Tabelle 2.2 zeigt die neu eingeführten Abstraktionseinheiten zur systematisierten Beschreibung von abstrakt formulierten, komplexen Handlungen (H) sowie mögliche Netzstrukturen.

Ein für die erfolgreiche Umsetzung der vorgeschlagenen Systematik entscheidendes Merkmal ist die Möglichkeit zur Modularisierung von Petri-Netzen [10, 261]. Dazu können transitionsberandete Teilnetze durch Transitionen und platzberandete Teilnetze durch Plätze ersetzt werden (Vergröberung). Alternativ kann ein Platz durch ein platzberandetes Teilnetz und eine Transition durch ein transitionsberandetes Teilnetz ersetzt werden (Verfeinerung). Wird diese Systematik zur Beschreibung von Roboterhandlungen

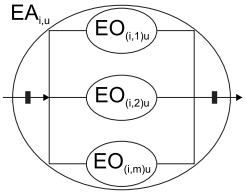
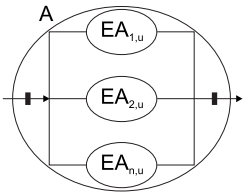
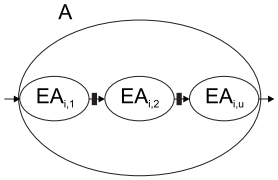
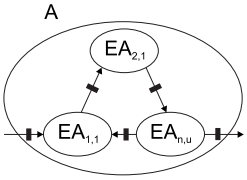
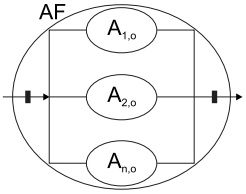
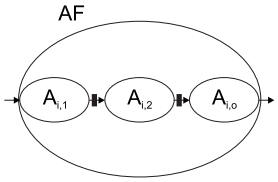
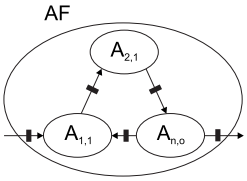
Netzstruktur Abstraktionseinheit	parallel (p)	sequentiell (s)	vernetzt (v)
Elementaraktion ($EA_{i,u}$)		-	-
Aktion (A)			
Aktionsfolge (AF)			

Tabelle 2.2: Abstraktionseinheiten zur systematisierten Beschreibung von abstrakt formulierten, komplexen Handlungen mit Petri-Netzen sowie zulässige Netzstrukturen

auf hierarchische Petri-Netze angewendet, dann entspricht jeder Platz eines übergeordneten Petri-Netzes einer Aktion A bzw. einer Aktionsfolge AF, die wiederum in Form eines eigenen Petri-Netz-Bausteins als platzberandetes Teilnetz (bestehend aus Elementaraktionen $EA_{i,u}$ bzw. Aktionen A) gegeben ist. Dieses Dekompositionssschema setzt sich bis zu den untersten in dieser Arbeit definierten Abstraktionseinheiten den Elementaroperationen EO fort. Bild 2.7 zeigt beispielhaft die vollständige Zuordnung der komplexen Aufgabenbeschreibung zu der vorgeschlagenen modularisierten Petri-Netz-Struktur.

Zur Abbildung von komplexen, parallel und/oder sequenziell zu verarbeitenden Roboteraufgaben (Tabelle 2.2) verfügen Petri-Netze zudem über Mechanismen zur Verzweigung (eine Transition bedient mehrere Nachfolgeplätze), Zusammenführung (die Folgetransition kann erst schalten, wenn alle Zweige fertig sind) und Synchronisation (mehrere Zweige werden durch eine gemeinsame Transition synchronisiert) [221]. Die so entstehenden hierarchisch strukturierten Petri-Netze werden im Weiteren als *Koordinierungsnetze* bezeichnet. In der Literatur existieren bereits Ansätze zur Koordinierung von Roboterkomponenten mit Petri-Netzen. In [22] wird beispielsweise ein Konzept vorgeschlagen, mit dem sich die Kooperation von zwei bzw. drei Teilsystemen eines humanoiden Robotersystems realisieren lässt. Aufgaben, bei denen zwei Teilsysteme gekoppelt gesteuert werden müssen, sind beispielsweise kooperative Zweiarmbewegungen oder einarmige Greifvorgänge. Mehr als drei Teilsysteme sind beispielsweise bei zweiarmigen Transportvorgängen (beteiligte Teilsysteme zwei Arme, zwei Greifer und Plattform) oder visuell überwachten Greifbewegungen (beteiligte Teilsysteme Arm, Greifer, Kopf und Plattform) zu koordinieren. Allerdings sind bereits beim Entwurf solcher komplexer Petri-Netz-Strukturen geeignete Analysepraktiken anzuwenden, um Netzkonflikte, Konfusionen (doppelter Konflikt) und Verklemmungen (Deadlocks) ausschließen zu können (siehe Anhang A).

Mit den beschriebenen Mechanismen zur Modularisierung, Verzweigung, Zusammenführung und Syn-

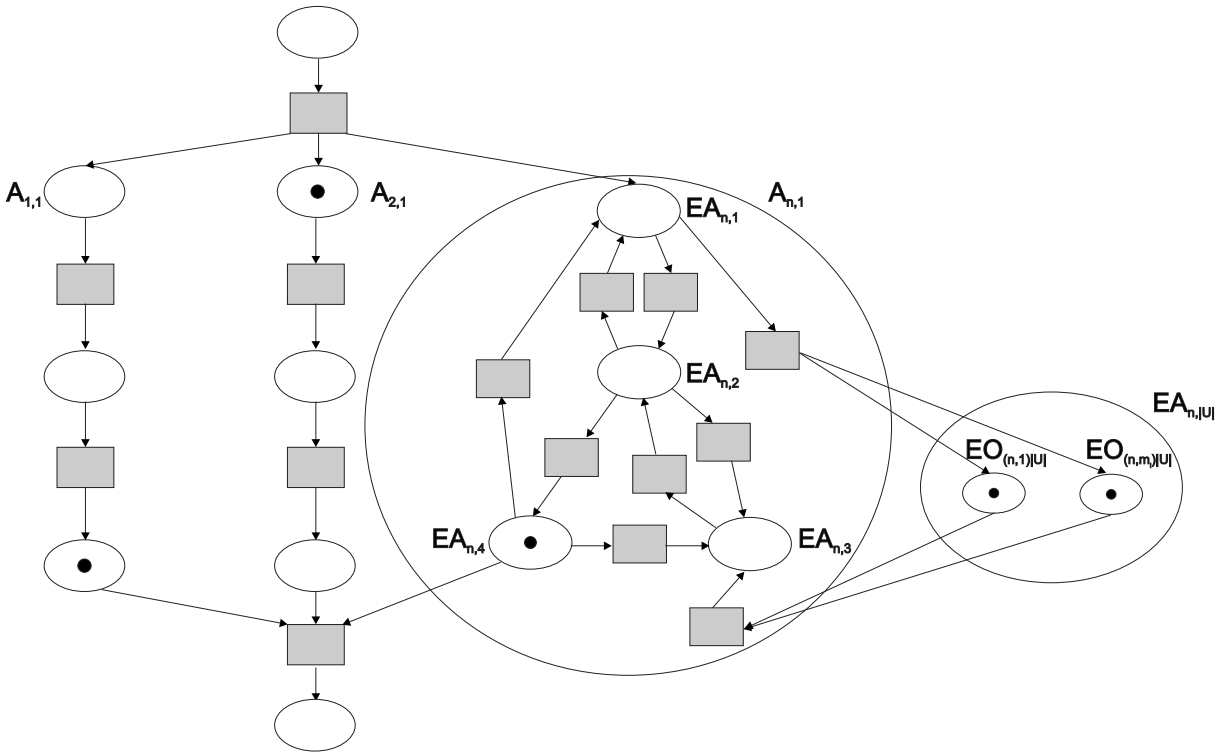


Bild 2.7: Beispiel für die Abbildung von Aktionen A, Elementaraktionen $EA_{i,u}$ und Elementaroperationen EO_a mit modularisierten Petri-Netzen

chronisation können jetzt komplexe Handlungen (H) einfach, ohne Detailwissen über die darunter liegenden Elementaroperationen bzw. Elementaraktionen, zusammengestellt werden. Lediglich die Schnittstellen müssen eindeutig definiert sein (Abschnitt 2.2.1). Zur Schnittstellendefinition werden modularisierte Petri-Netz-Bausteine verwendet (hier ausschließlich platzberandete Teilnetze). Diese Vorgehensweise ist beispielsweise bei der Planung komplexer Koordinierungsnetze auszunutzen, indem verschiedene Entwickler unabhängig voneinander an separaten Teilproblemen (Elementaraktionen als Petri-Netz-Bausteine) arbeiten können. Die Transitionen der Koordinierungsnetze hängen dabei von geplant (z. B. Kontaktzustände oder erreichte Sollpositionen bzw. -kräfte gemessen durch die perzeptiven Komponenten der Roboter-Steuerungsarchitektur) oder ungeplant (z. B. Benutzereingriffe oder modellbasiert detektierte Fehler) eintretenden Ereignissen ab.

Mit der vorgestellten Systematik steht somit ein Methodenapparat zur Verfügung, der es erlaubt, prinzipiell beliebig komplexe Handlungen zusammenzustellen, die auf einfachen im Aufgabenwissen des Robotersystems hinterlegten Elementaraktionen aufbauen. Dies kann dabei manuell durch den Entwickler oder autonom durch die Planungskomponente des Robotersystems erfolgen. Voraussetzung ist allerdings das Vorhandensein von regelungstechnischem Aufgabenwissen in der Datenbank des Robotersystems.

2.2.2.2 Aufgabenkoordination

Die jetzt als Aktionsfolge (AF) in Form eines Koordinierungsnetzes vorliegende Sequenz von Handlungen[7] wird in den Aufgabenspeicher auf der mittleren Hierarchieebene weitergeleitet. In der Auf-

[7] Eine Aktionsfolge (AF) entspricht somit *einer* vollständig parametrierten petrinetzbasierten Realisierung einer konkreten Handlung H.

gabenkoordination erfolgt dann die synchronisierte Ausführung der im Aufgabenspeicher abgelegten Koordinierungsnetze.

Zusätzlich werden die derzeit verfügbaren Hardware- und Softwareressourcen analysiert und mit den Anforderungen der ausführungsbereiten Koordinierungsnetze verglichen. Die Ressourcenverwaltung ist dabei besonders bei humanoiden Robotersystemen eine Herausforderung, da häufig mehrere Roboter-Teilsysteme an der Ausführung der teilweise komplexen Roboteraufgaben beteiligt sind. Als Ressourcen werden im Weiteren Hardware-Komponenten des Robotersystems (Plattform, Kopf mit Kameras und Mikrophonen, Greifer und Arme) mit dazugehörenden Software-Modulen sowie Rechnerkapazitäten bezeichnet. Treten dabei Konflikte auf, z. B. eine für die Ausführung der nächsten Aufgabe benötigte Roboterkomponente ist derzeit nicht bereit, dann wird die Ausführung dieser Aufgabe von der Aufgabenkoordination blockiert (Zugriff auf die Aktorik des entsprechenden Teilsystems verweigert), bis der Konflikt bereinigt ist. Mögliche Ursachen hierfür sind das aktive Ausführen einer anderen Aufgabe mit derselben Teilkomponente oder Hardwaredefekte. So wird z. B. der Roboterkopf zum Visual Servoing, zur Navigation, zur Bedienerinteraktion, zum Erkennen von externen Ereignissen und zur Aufmerksamkeitssteuerung verwendet. Ähnliche Probleme treten auch bei Softwarekomponenten auf (beispielsweise Objekt- und Bedienererkennung) [62]. Bei der Verwaltung von Hardwareressourcen ist zudem nicht nur darauf zu achten, welche Ressourcen *aktiv* an der Ausführung einer Aktion beteiligt sind. So benötigt ein Greifvorgang scheinbar nur Aktionen des Greifers und des Roboterarms, verlangt aber gleichzeitig eine feste Position für die Plattform und den Torso. Diese Ressourcen sind folglich während der Ausführung des Greifvorgangs für andere Aufgaben zu blockieren (*passiv*).

Wenn keine Konflikte vorliegen, startet die Aufgabenkoordination das Koordinierungsnetz mit der höchsten Priorität (Prioritätsverwaltung). Dadurch sind zeitgleich die entsprechenden Ressourcen blockiert, bis die Ausführung dieser Aufgabe beendet ist (mit oder ohne Erfolg). Ist der Konflikt innerhalb eines vorgegebenen Zeitraums nicht aufzulösen, dann initiiert die Aufgabenkoordination durch entsprechende Misserfolgsmeldungen eine Neuplanung der Aufgabe durch die Aufgabenplanung.

2.2.2.3 Aufgabenausführung

Die Aufgabenausführung erfolgt auf der Roboter-Steuerungsarchitektur dann mit der in Abschnitt 2.2.1.1 vorgestellten diskret-kontinuierlichen Systemstruktur. Dazu sind lediglich die entsprechenden Plätze des Koordinierungsnetzes zu markieren und damit die im Aufgabenwissen hinterlegten Funktionen zur Regelung und Trajektoriengenerierung aufgabenspezifisch zu aktivieren.

2.2.3 Beispiel

Als Beispiel wird die Umsetzung eines gekoppelten Greif- und Transportvorgangs mit Hilfe des vorgeschlagenen modularisierten Petri-Netz-Konzepts vorgestellt. Alle folgenden Ausführungen werden anhand der Aktionsfolgen AF aus Tabelle 2.3 erläutert. Die an dieser Aufgabe beteiligten Teilkomponenten des humanoiden Robotersystems sind die bewegliche Plattform und der Torso, ein Arm, ein Greifer und der Kopf mit zwei Stereo-Kameras. Der zweite Arm und der zweite Greifer sind nicht eingebunden und müssen lediglich Minimalanforderungen (keine Kollisionen usw.) erfüllen.

Zur Lösung der Aufgabe werden zunächst die im Aufgabenwissen des Robotersystems hinterlegten Elementaraktionen $EA_{i,u}$ (inkl. Parameterierung) von der Aufgabenplanung oder dem Entwickler zu teilsystemspezifischen Aktionen zusammengesetzt. So enthält die Aktion 'koordiniertes Annähern' des Greifers 1 in Tabelle 2.3 (grau) beispielsweise die Elementaraktionen aus Bild 2.8 (rechts) als platzberandetes Teilnetz. Der einheitlichen Aktion 'Aufmerksamkeitssteuerung' des Kopfes entspricht das

Aktionsfolge AF Teilkomponenten	HINFAHREN +SUCHEN	TASSE GREIFEN	ZURÜCK FAHREN	ÜBERGEBEN
Plattform und Torso	Annähern	Warten	Transport	Warten
Arm 1	Annähern	Warten	Transport	Annähern
Greifer 1	koordiniertes Annähern	Zugreifen	Halten	koordiniertes Loslassen
Kopf	Aufmerksamkeits- steuerung	Aufmerksamkeits- steuerung	Aufmerksamkeits- steuerung	Aufmerksamkeits- steuerung

Tabelle 2.3: Hardware-Ressourcen für das Beispiel 'Greif- und Transportvorgang'

platzberandete Teilnetz aus Bild 2.8 (links) [174]. Die Bedeutung der einzelnen Plätze und Transitionen der platzberandeten Teilnetze sowie der dazu verwendeten Abkürzungen wird im Abschnitt 2.3.3 detailliert beschrieben.

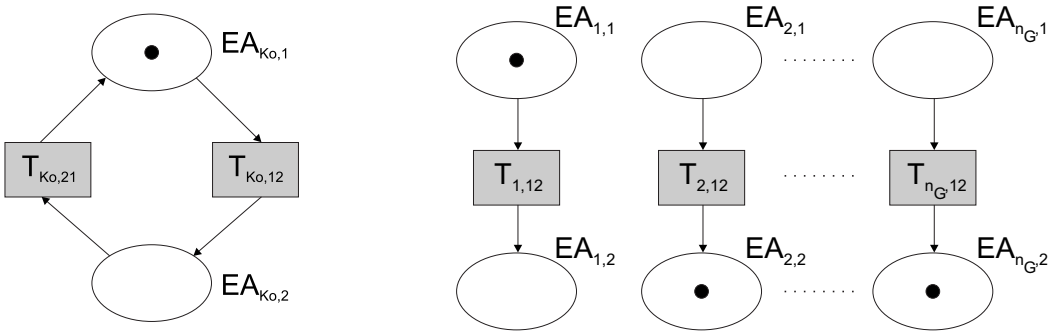


Bild 2.8: Ausschnitte aus den platzberandeten Teilnetzen der Aktionen 'Aufmerksamkeitssteuerung' für die Halseinheit eines humanoiden Roboters (links) und 'koordiniertes Annähern' für einen Mehrfinger-Robotergreifer (rechts)

Ist während der Ausführung der Aktion 'Aufmerksamkeitssteuerung' das zu verfolgende Objekt (z. B. eine vom Roboter zu greifende Tasse) lokalisiert[8], so ist der Platz $EA_{Ko,1}$ markiert (Bild 2.8 (links)) und das Objekt wird vom Robotersystem bildbasiert verfolgt (Elementaraktion 'Verfolgen' aus Tabelle 2.1). Falls das Objekt von der Kamera nicht mehr lokalisiert werden kann, wird die Transition $T_{Ko,12}$ aktiviert. Dies führt zur Markierung des Platzes $EA_{Ko,2}$. Dort startet die Halseinheit einen positionsgeregelten Suchvorgang (Elementaraktion 'Suchen' aus Tabelle 2.1). Während der Suche nach dem zu verfolgenden Objekt findet bei einem eindeutigen Erfolg ein Markenfluss mit $T_{Ko,21}$ zurück auf den Platz $EA_{Ko,1}$ statt. Hat der Robotergreifer die vorausberechnete Greifposition in der für den geplanten Griff notwendigen Greifkonfiguration (Preshape) erreicht, so werden für alle am Griff beteiligten Finger n_G die Plätze $EA_{1..n_G,1}$[9] der Aktion "koordiniertes Annähern" markiert. Ergibt sich für einen oder mehrere Finger vorzeitig Kontakt mit dem zu greifenden Objekt, erfolgt für diese Finger $n_1 \leq n_G$ ein Übergang $T_{n_1,12}$ zum Platz $EA_{n_1,2}$. Dort warten (Elementaraktion "Warten" aus Tabelle 2.1) diese Finger, bis *alle* am

[8] Im Sichtfeld der Kamera.

[9] Es hat sich als zweckmäßig herausgestellt, das Teilsystem Mehrfinger-Robotergreifer in n_F separate Finger mit jeweils m_i Gelenken zu unterteilen, um damit den Greifvorgang in leichter zu koordinierende und zudem anschaulichere Fingerbewegungen zu unterteilen. Dieses Vorgehen erleichtert somit den Entwurf der zur Beschreibung von Greifvorgängen mit einem Mehrfinger-Robotergreifer notwendigen Koordinierungsnetze, vgl. Abschnitt 4.2.1.3 und [174].

Griff beteiligten Finger n_G Kontakt haben $T_{1..n_G,12}$, damit dann ein koordinierter Markenfluss zu den Plätzen $EA_{1..n_G,3}$ erfolgen kann (Elementaraktion ”Zugreifen” aus Tabelle 2.1).

Zur koordinierten Durchführung des komplexen Greif- und Transportvorgangs werden die auf diese Weise definierten Aktionen A zu Aktionsfolgen AF in Form eines Koordinierungsnetzes zusammengesetzt. Bild 2.9 zeigt das resultierende Koordinierungsnetz für das Beispiel ’Greif- und Transportvorgang’. Zu beachten ist dabei, dass das hier dargestellte Netz ’nur’ die routinemäßige Abarbeitung der Aktionsfolgen AF als Petri-Netze ohne gesonderte Plätze für Ausnahmesituationen enthält (vgl. Abschnitt 2.3).

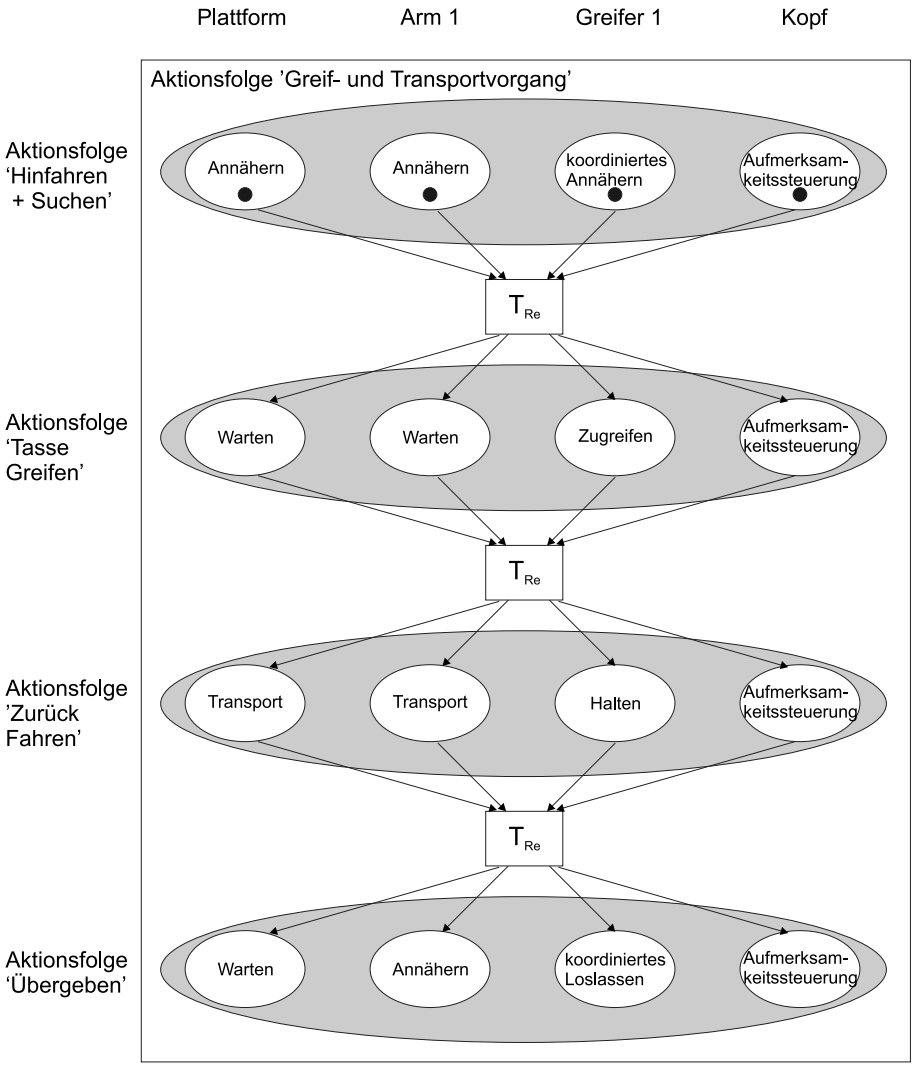


Bild 2.9: Koordinierungsnetz der Aktionsfolge ’Greif- und Transportvorgang’

Nach der Aufgabenplanung wird bei Verfügbarkeit aller notwendigen Ressourcen die erste Aktionsfolge ’Hinfahren + Suchen’ gestartet. Dieser Start setzt im Koordinierungsnetz die Verfügbarkeit der Hardwareressourcen Plattform, erster Arm, erster Greifer und Kopf voraus. Nach Abschnitt 2.2.2.2 ist dabei nicht nur darauf zu achten, welche Ressourcen *aktiv* an einer Aktionsfolge beteiligt sind. So benötigt zwar die Aktionsfolge ’Tasse Greifen’ scheinbar nur die Aktion des Greifers, verlangt aber gleichzeitig eine feste Position für den Arm 1 sowie die Plattform und den Torso. Diese Ressourcen sind folglich zu blockieren. Die entsprechend auszuführenden Aktionen sind ’Warte’ (Halte aktuelle Position). An-

dere Ressourcen, wie hier der zweite Arm und der zweite Greifer, sind im Prinzip frei verfügbar und müssen lediglich auf eine Kollisionsvermeidung achten. Erst wenn alle an der derzeit aktiven Aktionsfolge beteiligten Ressourcen die jeweilige Aktion erfolgreich abgeschlossen haben, erfolgt wiederum nach Überprüfung aller notwendigen Hardwareressourcen der koordinierte Start der nächsten Aktionsfolge (Aktivierung der Transition T_{Re}, Bild 2.9). Erst nach der erfolgreichen Abarbeitung der letzten Aktionsfolge ('Übergeben') werden die blockierten Ressourcen freigegeben.

2.3 Überwachungskonzept

Um die Sicherheit eines technischen Systems zu gewährleisten, muss ein Überwachungskonzept nach Abschnitt 1.2.3 zunächst in der Lage sein, auftretende Fehler zu erkennen (Fehlerdetektion) und geeignet zu klassifizieren (Fehlerisolation). Dieser Prozess wird in der Literatur allgemein als Fehlerdiagnose (engl: Fault Detection and Isolation FDI) bezeichnet. Im Anschluss daran erfolgt die Analyse des klassifizierten Fehlers (Fehleranalyse), um geeignete Maßnahmen zur Problemlösung einzuleiten (Problemlösung).

Spezielle Anforderungen an ein Überwachungskonzept bei humanoiden Robotern resultieren aus der Forderung nach einer multimodalen Kooperation mit dem Menschen und damit der Fähigkeit, sich in veränderlichen Umgebungen mit unbekannten Situationen auseinanderzusetzen. So muss ein solches Robotersystem innerhalb des vorgeschlagenen mehrstufigen Überwachungsprozesses in der Lage sein, die Sicherheit auch bei kooperativen Tätigkeiten im Arbeitsraum des Menschen zu garantieren (Benutzersicherheit). Um dies zu erreichen, muss das Robotersystem zur selbstständigen Bewertung der aktuellen, sich ständig ändernden Situation (Istzustand S_{Ist}) fähig sein (Fehlerdiagnose Bild 2.10 und Abschnitt 2.3.1), um bei erkannten Ausnahmesituationen geeignete Maßnahmen zur erfolgreichen Problemlösung einleiten zu können (Problemlösung Bild 2.10 und Abschnitt 2.3.2).

Allerdings ist nach Abschnitt 1.2.3 eine autonome Problemlösung der detektierten Fehler in den allgemein anwendbaren Konzepten derzeit nicht Stand der Technik. Demgegenüber stehen humanoiden Robotern heutzutage durch ihre motorischen und kognitiven Fähigkeiten (vgl. Abschnitt 1.2.4), Strategien zur Umsetzung dieser Anforderungen zur Verfügung. Solche speziellen Handlungen (H) zur Problemlösung in humanoiden Robotern sind beispielsweise:

- das Unterbrechen der Handlung und Start eines Klärungsdialogs mit dem Benutzer,
- das Unterbrechen der Handlung und Start einer Problembehandlung durch Telemanipulation,
- das verlangsamte Ausführen der Handlung,
- das Ausführen von Ausweichbewegungen zur Kollisionsvermeidung,
- das reflexartige Ausführen von Gegenbewegungen nach bereits erfolgter Kollision,
- das annähernd planmäßige Beenden der Handlung und Ignorieren des Problems,
- der Abbruch der Handlung und der Übergang in einen sicheren Zustand sowie
- die selbstständige Modifikation der kompletten Handlung.

Im Weiteren wird nun eine aus dem allgemeinen mehrstufigen Überwachungsprozess abgeleitete Grundstruktur für ein neues Überwachungskonzept von humanoiden Robotern präsentiert, das zudem in der Lage ist, die beschriebenen Strategien zur autonomen Problemlösung einfach und systematisch in die vorgegebene Roboter-Steuerungsarchitektur zu integrieren. Die grundlegende Idee des neuen Ansatzes besteht dabei in der Zuordnung der Komponenten zur Fehlerdiagnose und Problemlösung zu der unteren

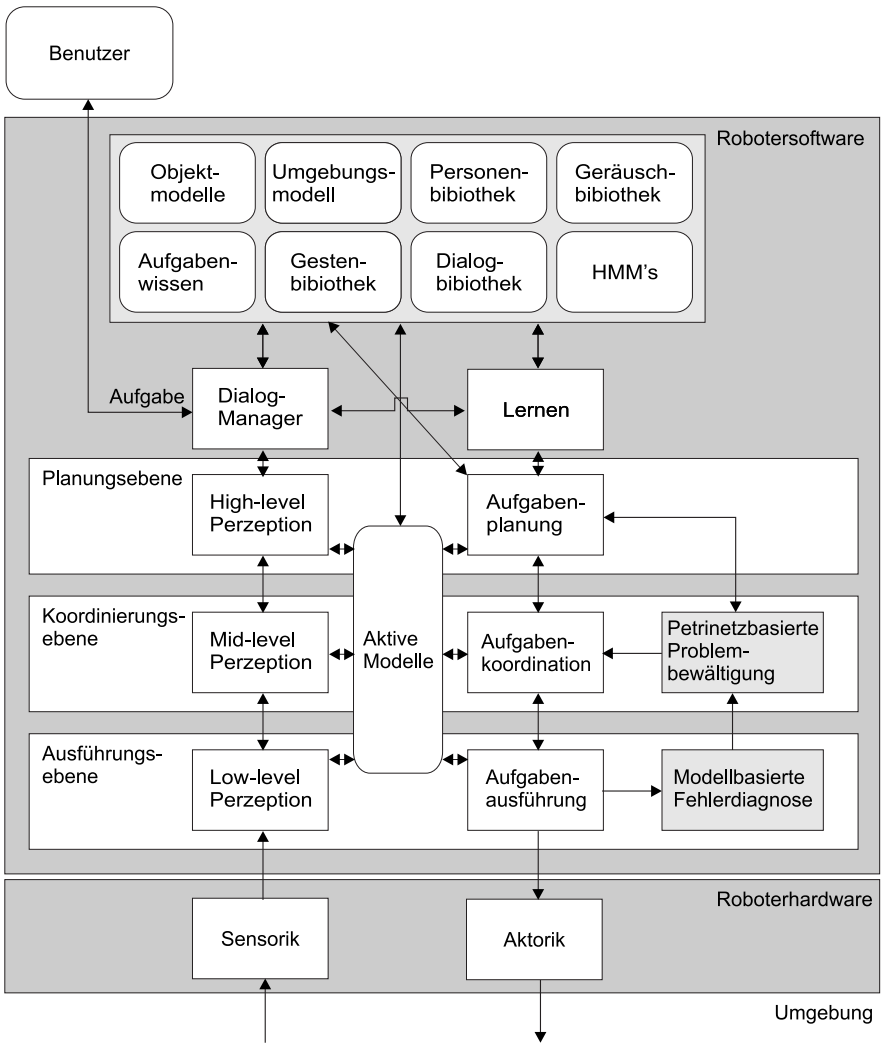


Bild 2.10: Hierarchische Steuerungsarchitektur des Sonderforschungsbereichs 588 mit integrierten Überwachungskomponenten nach [171]

und mittleren Hierarchieebene der Roboter-Steuerungsarchitektur. Bild 2.10 zeigt detailliert den neuen Ansatz, integriert in die kognitive Steuerungsarchitektur des Sonderforschungsbereichs 588.

Auf der unteren Ebene der Steuerungsarchitektur findet die modellbasierte Fehlerdiagnose statt. Dazu werden in der Stabilitätsüberwachung die unterlagerten Regelkreise auf Einhaltung des vorgegebenen Sollverhaltens und in der Hardwareüberwachung die verfügbaren Hardwareressourcen (Aktorik und Sensorik) auf Funktionalität überwacht. Auf der mittleren Ebene der Steuerungsarchitektur erfolgt anschließend die systematische Behandlung der modellbasiert erkannten Ausnahmesituationen sowie zusätzlicher nicht planbarer Ereignisse wie Benutzereingriffe oder unerwünschte Platzmarkierungen durch die petrinetzbasierte Problemlösung des Überwachungskonzepts [171].

Das Ziel der vorgeschlagenen Grundstruktur besteht dabei darin, die mittlere und die untere Hierarchieebene mit einem Höchstmaß an Autonomie zu versehen. Die entsprechenden Komponenten werden in die Lage versetzt, falls nötig, schnell und ohne den direkten Eingriff der Aufgabenplanung auf auftretende Ereignisse (routinemäßig erwartete Situationen und berücksichtigte Ausnahmesituationen) zu

reagieren. Dazu werden die Zustände aller Komponenten der unterlagerten Regelkreise (u.a. Regler, Aktoren, Sensoren) zur Laufzeit überwacht und entsprechend der geforderten Aufgabe bewertet. Auf Basis dieser Informationen entscheidet die übergeordnete Ebene selbstständig über die anzuwendende Ausnahmebehandlungsstrategie. Dies ist ein deutlicher Unterschied zu den klassischen Ansätzen der Aufgabenplanung [40, 190, 191, 235, 281], bei denen Änderungen des ursprünglichen Plans stets von der obersten Ebene der Steuerungsarchitektur überwacht und ausgeführt werden (vgl. Abschnitt 1.2.2 und [75]). Beim Entwurf der Überwachungskomponenten ist dabei darauf zu achten, die Strategien zur Behandlung von Ausnahmesituationen immer mit der entsprechenden Fehlerart und/oder der aktuellen Situation des Robotersystems assoziiert zu sehen.

Im Weiteren erfolgt die Beschreibung der theoretischen Grundlagen der einzelnen Komponenten des neuen Sicherheits- und Überwachungskonzepts für humanoide Robotersysteme. Teilsysteme, die dabei detailliert vorgestellt werden, sind die modellbasierte Fehlerdiagnose mit den Komponenten zur Stabilitäts- und Hardwareüberwachung (Abschnitt 2.3.1) sowie die petrinetzbasierte Problemlösung mit den Komponenten regelbasierter Auswertung und Verwaltung (Abschnitt 2.3.2).

2.3.1 Modellbasierte Fehlerdiagnose

2.3.1.1 Übersicht

Die modellbasierte Fehlerdiagnose hat die Aufgabe, die Zustände aller Komponenten der unterlagerten Regelkreise, bestehend aus Basis-Regelung und Roboterhardware (Aktorik und Sensorik), zur Laufzeit zu überwachen und entsprechend der geforderten Aufgabe zu bewerten. In der Literatur existieren zahlreiche Verfahren zur modellbasierten Fehlerdiagnose [81, 93, 131, 242]. Die grundlegende Struktur der verwendeten modellbasierten Fehlerdiagnose ist in Bild 2.11 dargestellt und besteht aus einem als Ausgangsbeobachtermodell umgesetzten Residuengenerator sowie einem regelbasierten Auswertungsmodul. Bild 2.10 zeigt die Eingliederung der modellbasierten Fehlerdiagnose in das in dieser Arbeit vorgestellte Gesamtkonzept.

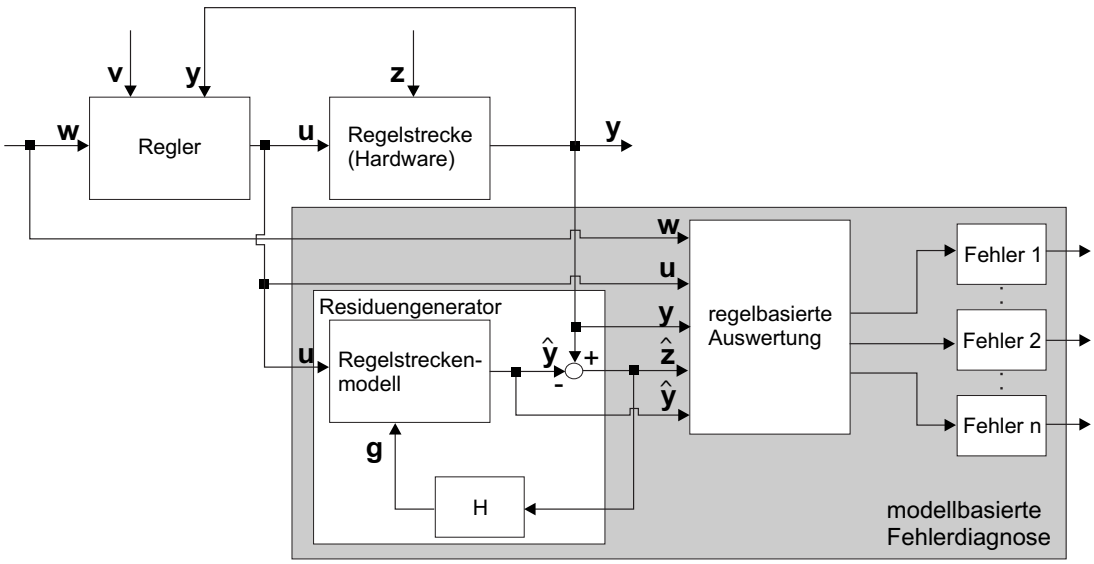


Bild 2.11: Allgemeine Struktur der modellbasierten Fehlerdiagnose

Eingangsgrößen der modellbasierten Fehlerdiagnose sind der Vektor der Führungsgrößen $\boldsymbol{w}$, der Vektor der Stellgrößen $\boldsymbol{u}$ und der Vektor der Regelgrößen $\boldsymbol{y}$. Ausgangsgrößen sind die in Abhängigkeit von der Umsetzung des regelbasierten Auswertungsmoduls berechneten Fehlerzustände. Ist es das Ziel, ein fehlerhaftes Verhalten der Basis-Regelung zu detektieren, dann wird die regelbasierte Auswertung als Stabilitätsüberwachungs-Modul umgesetzt und als Ausgangsgröße ein sogenannter Stabilitätsgrad S berechnet (Abschnitt 2.3.1.3). Wird hingegen die Roboterhardware (Aktorik und Sensorik) modellbasiert überwacht, dann berechnet die als Hardwareüberwachungs-Modul implementierte regelbasierte Auswertung den Fehlerzustandsvektor $\boldsymbol{F}$ (Abschnitt 2.3.1.4). Die berechneten Informationen werden an die übergeordneten Überwachungskomponenten weitergeleitet (Abschnitt 2.3.2) und dort zur Problemlösung verwendet.

Teilmodelle, die im Weiteren einzeln vorgestellt werden, sind der *beobachtergestützte Residuengenerator* (Abschnitt 2.3.1.2) und die in Abhängigkeit von der zu überwachenden Komponente des unterlagerten Regelkreises umgesetzte *regelbasierte Auswertung* (regelbasierte Auswertung zur Stabilitäts- (Abschnitt 2.3.1.3) und zur Hardwareüberwachung (Abschnitt 2.3.1.4)).

2.3.1.2 Residuengenerierung

Ziel des beobachtergestützten Residuengenerators ist es, mit Hilfe eines Regelstreckenmodells die Regel- und Störgrößen des überwachten Systems zu schätzen. Wirken auf das reale System unbekannte externe Störungen $\boldsymbol{z}$, so ergibt sich eine Differenz zwischen berechneten und am realen System gemessenen Regelgrößen:

$$\hat{\boldsymbol{z}}[k] = \boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k]. \tag{2.21}$$

Zu beachten ist dabei, dass es sich bei dem vorgestellten Residuengenerator um einen Ausgangsbeobachter handelt, dessen Ziel im Gegensatz zum Zustandsbeobachter nicht in der Rekonstruktion des gesamten Zustandsvektors liegt.

Die berechnete Differenz $\hat{\boldsymbol{z}}$ (Residuum[10]) aus geschätzten und gemessenen Regelgrößen wird über eine Matrix $\boldsymbol{H}$ auf das Regelstreckenmodell zurückgekoppelt:

$$\begin{aligned} \boldsymbol{g}[k] &= \boldsymbol{H} \cdot \hat{\boldsymbol{z}}[k], \\ \hat{\boldsymbol{y}}[k] &= \boldsymbol{f}(\boldsymbol{u}[k], \boldsymbol{g}[k]). \end{aligned} \tag{2.22}$$

Für die Fehlerdiagnose ist die Matrix $\boldsymbol{H}$ so zu bestimmen, dass gilt:

$$\hat{\boldsymbol{z}}[k] \to 0 \quad \text{für} \quad k \to +\infty. \tag{2.23}$$

Wichtig ist, dass die Güte der Residuengenerierung und damit auch der gesamten modellbasierten Fehlerdiagnose in hohem Maße von der Modellgüte des Regelstreckenmodells abhängig ist. Denn nur mit einem möglichst genauen Streckenmodell und dem daraus abgeleiteten charakteristischen Residuenvektor sind die formulierten Anforderungen an die Überwachung und Fehlerfrüherkennung bei gleichzeitiger Robustheit zu erreichen.

2.3.1.3 Regelbasierte Auswertung zur Stabilitätsüberwachung

Beim Einsatz komplexer Systeme ist die Gewährleistung der Stabilität des Gesamtsystems in allen Betriebssituationen ein wichtiges Ziel. Dies gilt besonders für technische Bereiche mit hohen Sicherheitsanforderungen. Als Stabilität wird hierbei die Fähigkeit bezeichnet, nach Einwirkung von Störungen in einen Gleichgewichtszustand zu gelangen bzw. eine Referenztrajektorie zu erreichen.

[10] in der Literatur häufig auch mit $\boldsymbol{r}$ bezeichnet.

Strukturell handelt es sich bei humanoiden Robotern häufig um

- nichtlineare (z. B. durch Antriebskomponenten wie Gleichstrommotoren, Getriebe und fluidische Aktoren [19, 248] oder Reibung [16], ...),
- diskret-kontinuierliche (durch kontinuierliche Referenztrajektorien und Stellgrößenverläufe und diskrete Petri-Netz-Zustände in der Aufgabenplanung [170, 239] (vgl. Abschnitt 2.2.1.1), ...),
- strukturvariable (durch Strukturumschaltung von Regler und/oder Streckenmodellen z. B. bei Objektkontakt [105, 163, 170] (vgl. Abschnitt 2.2.1.1), ...),
- gekoppelte (lose Kopplung einzelner Roboterkomponenten wie z. B. Greifer oder Arme als offene kinematische Kette oder vollständige Kopplung des Gesamtsystems als eine abgeschlossene kinematische Kette [22, 90], ...)
- Mehrgrößen-Systeme (durch u. U. simultane Regelung mehrerer Systemgrößen wie z. B. Lage, Geschwindigkeit, Beschleunigung sowie Kräfte und Momente [90], ...).

Ein formaler Stabilitätsnachweis für solche Systeme ist wegen der schwer modellierbaren Störeinflüsse und der hybriden kontinuierlich-diskreten Systemeigenschaften für praktisch relevante Aufgaben jedoch nahezu unmöglich [170]. Im Gegensatz dazu ist ein menschlicher Experte in der Lage, mit seinem auf Erfahrung basierendem Prozesswissen das derzeitige Systemverhalten einzuschätzen und bei erkannten Gefahrensituationen Modifikationen der Regelungsstrategie zur Stabilitätssicherung des Prozesses einzuleiten [57]. Üblicherweise wird dazu auch vom Menschen ein mehrstufiger Prozess bestehend aus Fehlerdetektion, Fehlerisolation und Problemlösung angewendet (vgl. Abschnitt 1.2.3).

Um diese Strategie in einem hierarchischen Roboter-Steuerungssystem umzusetzen, wird hier ein Fuzzy-Überwachungskonzept verwendet, das auf verschiedenen bekannten Ansätzen basiert [102, 193, 194, 280]. Für die Anwendung bei häufig nichtlinearen, strukturvariablen, gekoppelten Roboterregelungen ist das bestehende Verfahren jedoch anzupassen. Deshalb erfolgt im Weiteren zunächst eine kurze Darstellung des Grundkonzepts der Fuzzy-Überwachung (Abschnitt Grundkonzept), gefolgt von den für den Einsatz bei humanoiden Robotersystemen erforderlichen Erweiterungen (Abschnitt Erweiterungen).

Grundkonzept Die grundsätzliche Aufgabe der Fuzzy-Überwachung ist die echtzeitfähige Bewertung der Gesamtsituation des überwachten Systems. Ist die Regelung in der Lage, das System nach äußeren Störungen oder Änderungen der Referenztrajektorien wieder in Richtung des Sollverhaltens auszuregeln, stuft die Überwachung die Situation als 'STABIL' ein. Gelingt dies nicht, wird die Gesamtsituation als 'INSTABIL' eingeschätzt. Das Ergebnis dieser Beurteilung ist ein online berechneter Stabilitätsgrad S, mit Zuständen *zwischen* 0 ('INSTABIL') und 1 ('STABIL') [192, 197]. Die für die Beurteilung der Zustände der Basisregler notwendigen Referenztrajektorien werden von der Aufgabenausführung der Steuerungsarchitektur des humanoiden Roboters berechnet (vgl. Abschnitt 2.2). In Bild 2.12 ist die detaillierte Struktur der Fuzzy-Überwachung dargestellt.

Eingangsgrößen der Fuzzy-Überwachung sind der Vektor der Führungsgrößen $\boldsymbol{w}$ und der Vektor der Regelgrößen $\boldsymbol{y}$. Die Ausgangsgröße ist die berechnete Stabilitätsbeurteilung in Form des Stabilitätsgrads S. Teilmodelle, die im Weiteren einzeln vorgestellt werden, sind die *Bewertungsfunktion* L, die *Korrektur der Bewertungsfunktion* L_{Extern}, die *korrigierten und gefilterten Bewertungsfunktionen* L_{M}, L_{N}, die *mittlere Bewertungsfunktion* L_{Mittl}, der *Trend* D, der *gewichtete Trend* D_{W} und die *Fuzzy-Regelbasis* mit dem *Stabilitätsgrad* S.

Ausgehend von einem unscharfen Stabilitätsbegriff nach [28, 192] werden bei diesem Ansatz Ljapunov-ähnliche Funktionen [192] als Bewertungsmaß eingesetzt, die die gewünschte Beurteilung tolerierbarer Ausgangsgrößen erlauben. Dieses Bewertungsmaß soll einen umso kleineren Wert liefern, je näher der

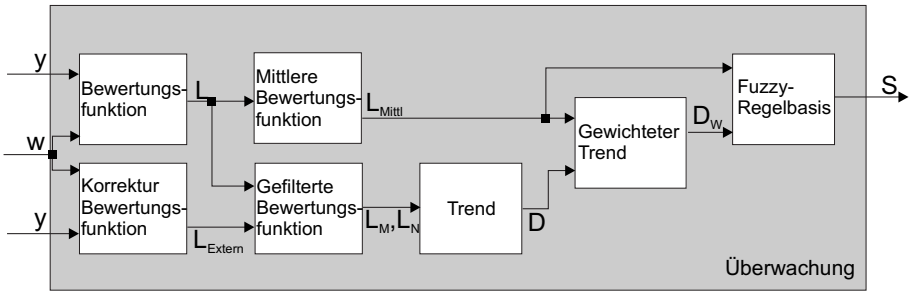


Bild 2.12: Modellstruktur der Fuzzy-Überwachung nach [192]

aktuelle Zustand des Systems dem gewünschten Zustand ist. Allerdings ergeben sich bei der Anwendung von Ljapunov-ähnlichen Funktionen zur Überwachung von Regelkreisen Probleme, von denen die wichtigsten, sowie die dazu gehörenden Lösungsansätze, kurz dargestellt werden.

Nicht alle Prozesszustände sind immer mess- oder beobachtbar. Deshalb wird anstelle einer Ljapunov-Funktion eine *Bewertungsfunktion* L verwendet, die nicht unbedingt alle Prozesszustände, sondern nur die verfügbaren Größen enthält. Ein Beispiel für eine solche Bewertungsfunktion ist

$$L[k] = L(\mathbf{e}[k]) = \mathbf{e}[k]^T \cdot \mathbf{P} \cdot \mathbf{e}[k] = (\mathbf{w}[k] - \mathbf{y}[k])^T \cdot \mathbf{P} \cdot (\mathbf{w}[k] - \mathbf{y}[k]) \tag{2.24}$$

mit der positiv definiten Matrix $\mathbf{P}$ und dem Vektor der Regeldifferenzen $\mathbf{e}[k]$. Auf Stabilität eines autonomen Systems kann geschlossen werden, wenn L alle Zustände erfasst und

$$L[k+1] = \begin{cases} < L[k] & \text{für} \quad L[k] > 0 \\ 0 & \text{für} \quad L[k] = 0 \end{cases} \tag{2.25}$$

gilt.

Freie und erzwungene Signale sind zu separieren, da sonst Änderungen der Führungs- bzw. Störgrößen $\Delta\mathbf{w}[k]$ bzw. $\Delta\mathbf{z}[k]$ zu einer schlechteren Stabilitätsbeurteilung durch die Bewertungsfunktion L führen. Deshalb werden diese Änderungen in der Bewertungsfunktion kompensiert, was im Weiteren zu einer *Korrektur der Bewertungsfunktion* L_{Extern} führt:

$$L_{\text{Extern}}[k+1] = L(\mathbf{w}[k+1], \mathbf{y}[k+1]) - L(\mathbf{w}[k+1] - \Delta\mathbf{w}[k], \mathbf{y}[k+1] - \Delta\mathbf{z}[k]). \tag{2.26}$$

Der zweite Term in L_{Extern} stellt die Bewertung durch die Überwachungsebene zum Zeitpunkt $k+1$ dar, falls die Führungs- und Störgrößen gleich geblieben sind ($L(\mathbf{w}[k], \mathbf{y}[k])$).

Da Gleichung (2.26) nur eine Beurteilung der Zeitpunkte k und $k+1$ ermöglicht, ist die Erweiterung um eine Rückwärtskorrektur erforderlich. Die sich dadurch ergebende Funktion wird als *korrigierte Bewertungsfunktion* bezeichnet. Bei der Bewertung ist weiterhin nicht eine kurzzeitige Verschlechterung des Bewertungsmaßes, sondern die längerfristige Entwicklung entscheidend. Deshalb erfolgt die Berechnung eines *Trends* durch *Filterung* der *korrigierten Bewertungsfunktion* anstelle der zeitlichen Ableitung

$$L_{\text{F}}[k+1] = a_{\text{F}}(L_{\text{F}}[k] + L_{\text{Extern}}[k+1]) + (1 - a_{\text{F}})L[k+1], \quad 0 < a_F < 1 \tag{2.27}$$

anhand von zwei Zeitfenstern M ($a_{\text{F}} = a_{\text{M}}$) und N ($a_{\text{F}} = a_{\text{N}} < a_{\text{M}}$). Ist der Mittelwert des Zeitfensters N kleiner als der Mittelwert des Zeitfensters M, so ist der Trend der Funktion negativ. Die Funktion D wird als *Trend* der *korrigierten und gefilterten Bewertungsfunktion* L_F bezeichnet:

$$D[k] = \frac{L_{\text{N}}[k] - L_{\text{M}}[k]}{T_{\text{N}} - T_{\text{M}}} \quad \text{mit} \quad T_{\text{N}} - T_{\text{M}} = T_{\text{A}} \frac{(a_{\text{M}} - a_{\text{N}})}{(1 - a_{\text{N}})(1 - a_{\text{M}})}. \tag{2.28}$$

Zusätzlich ergibt die Filterung der unkorrigierten *Bewertungsfunktion* L an, wie weit das System durchschnittlich von seinem Sollzustand entfernt war. Diese Größe wird als *mittlere Bewertungsfunktion* L_{Mittl} bezeichnet:

$$L_{\text{Mittl}}[k+1] = a_{\text{F}} L_{\text{Mittl}}[k] + (1 - a_{\text{F}}) L[k+1], \quad 0 < a_F < 1. \tag{2.29}$$

Weiterhin wird ein *gewichteter Trend* D_{W} eingeführt, der den *Trend* D in Bezug zur *mittleren Bewertungsfunktion* L_{Mittl} setzt. Dies ermöglicht eine Einschätzung des Systemverhaltens unabhängig von der Größenordnung der Bewertungsfunktion:

$$D_W[k+1] = \frac{D[k+1]}{L_{\text{Mittl}}[k]}. \tag{2.30}$$

Die *mittlere Bewertungsfunktion* L_{Mittl} und der *gewichtete Trend* D_{W} werden von einem Fuzzy-System linguistisch interpretiert. Stützstellen der Zugehörigkeitsfunktionen sind NEG (NEGATIV), N (NULL), PK (POSITIV KLEIN), PG (POSITIV GROSS) (vgl. Bild 2.13).

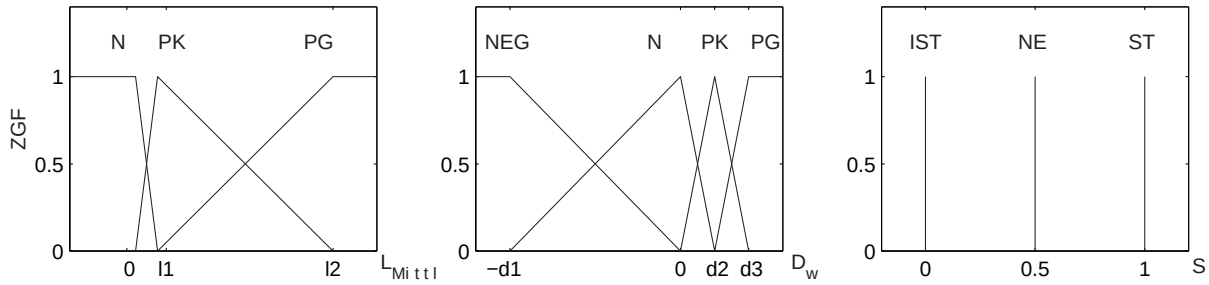


Bild 2.13: Beispiele für Zugehörigkeitsfunktionen zur Beurteilung des Stabilitätsgrads mit den Parametern l_1, l_2, d_1, d_2, d_3

Das Ergebnis ist eine qualitative Stabilitätsbeurteilung mit den linguistischen Werten STABIL (ST), INSTABIL (IST) und NICHT ENTSCHEIDBAR (NE), die nach einer Defuzzifizierung mit der Center of Singleton Methode auf einen Stabilitätsgrad zwischen 0 und 1 führt.

L_{Mittl} / D_{W}	N	PK	PG
NEG	STABIL	STABIL	STABIL
N	STABIL	NICHT ENTSCHEIDBAR	NICHT ENTSCHEIDBAR
PK	STABIL	NICHT ENTSCHEIDBAR	INSTABIL
PG	STABIL	INSTABIL	INSTABIL

Tabelle 2.4: Fuzzy-Regelbasis zur Beurteilung des Stabilitätsgrads nach [192]

Erweiterungen Aufgrund der beschriebenen strukturellen Komplexität von Regelkreisen bei humanoiden Robotersystemen ist das bestehende Verfahren anzupassen und zu verallgemeinern. So ist bei Roboterregelungen die Störgrößenkompensation nach [192] aufgrund von schwer modellierbaren (unbekannte sich ständig ändernde Umwelt) und meist (mit den Regelgrößen des Systems) korrelierten Störeinflüssen nicht anwendbar (vgl. Gleichungen (2.26) und (2.33)).

Weiterhin existieren derzeit lediglich Vorüberlegungen zur Umsetzung des Konzeptes bei gekoppelten Mehrgrößen-Systemen (im Weiteren MIMO (Multiple Input Multiple Output)- Systeme) [192]. Zudem müssen die bei Roboterregelungen häufig auftretenden Strukturvariabilitäten, z. B. durch Umschaltvorgänge bei Objektkontakten, in der zusammengefassten Stabilitätsbeurteilung Berücksichtigung finden, da andernfalls keine sinnvolle und einheitliche Bewertung der Gesamtsituation zu gewährleisten ist. Die Lösung dieser Probleme erfolgt durch

1. die Erweiterung des Fuzzy-Überwachungskonzepts zur systematisierten Anwendung bei gekoppelten Mehrgrößen-Systemen sowie
2. die Definition von Führungsgrößenintervallen zur aufgabenspezifischen Priorisierung der zusammengefassten Stabilitätsbeurteilung.

Grundsätzlich lässt sich der Begriff der Kopplung bei Robotersystemen auf zwei verschiedene Arten deuten. Aus der Sicht der koordinierten Bewegungssteuerung ist die Kopplung der einzelnen Roboterkomponenten (Plattform, Torso, Greifer, Arme und Kopf) bei der gemeinsamen Ausführung einer Aufgabe gemeint [22]. Bei der so genannten losen Kopplung werden die Roboter-Teilsysteme als getrennte, kinematisch offene Ketten mit unabhängiger Aufgabenplanung und -ausführung betrachtet. Demgegenüber werden bei der vollständigen Kopplung alle an der Aufgabe beteiligten Roboterkomponenten zu einer einzigen kinematischen Kette mit gemeinsamer koordinierter Aufgabenplanung und -ausführung zusammengefasst. Solche aufgabenspezifischen Ketten können entweder offen (z. B. Plattform, Torso, ein Arm und ein Greifer für Greif- und Transportbewegungen) oder geschlossen (z. B. Plattform, Torso, zwei Arme, zwei Greifer und zu greifendes Objekt) sein. Zur Ausführung von koordinierten Bewegungen hat sich dabei die Beschreibung als vollständig gekoppeltes System als zweckmäßig erwiesen (vgl. Abschnitt 2.2).

Im Gegensatz dazu ist für die hier vorgestellte Stabilitätsbeurteilung die regelungstechnische Kopplung der Regel- und Stellgrößen innerhalb einer Roboter-Teilkomponente entscheidend. Im Weiteren wird also von der separaten Betrachtung des geschlossenen Regelkreises *einer* mechanischen Komponente des humanoiden Robotersystems ausgegangen (nach Abschnitt 2.2.2 eine Aktionsfolge AF mit zugehörenden Elementaraktionen EA sowie entsprechenden Funktionen zur Trajektoriengenerierung und Regelung). Werden zur Laufzeit mehrere Roboterkomponenten parallel überwacht, so erfolgt für jeden Teilregelkreis eine eigene Stabilitätsbeurteilung für alle Regelgrößen des Teilsystems mit Hilfe einer separaten Bewertungsfunktion innerhalb eines eigenen Verwaltungsmoduls (vgl. Abschnitt 3.3).

Da bei der beschriebenen Art von Roboter-Systemklasse die aus der klassischen linearen Regelungstheorie bekannten Methoden zur Entkoppelung des Mehrfachregelkreises wie z. B. die Reihenentkopplung oder die Entkopplung einer V-Struktur nicht anwendbar sind [90] und zudem die Forderung nach Eigenautonomie[11] nicht zu realisieren ist (Beispiele hierfür sind u. a. Robotergreifer und -füße im Kontaktfall (Veränderung der Gelenkposition bedeutet automatisch Beeinflussung der Kontaktkraft) oder Roboterarme unter Nullkraftregelung), muss nach der in [192] vorgenommenen Einteilung eine gemeinsame Stabilitätsbewertung für alle Regelgrößen des Roboter-Teilsystems mit Hilfe einer gemeinsamen Bewertungsfunktion erfolgen. Allerdings ist hierbei zu beachten, dass bei unterschiedlicher Größenordnung der verschiedenen Regelgrößen, Wichtungsmatrizen $(\mathbf{P_1} \dots \mathbf{P_{n_y}})$ mit Wichtungselementen $(p_{n_y,1} \dots p_{n_y,m_i})$ eingeführt werden müssen, um diese Differenzen für eine korrekte Stabilitätsbeurteilung auszugleichen.

Für ein Mehrgrößensystem ergibt sich damit für die Vektoren der Führungs- und Regelgrößen

$$\mathbf{w} = (w_{1,1} \dots w_{1,m_i}, \dots, w_{n_y,1} \dots w_{n_y,m_i})^T$$
$$\mathbf{y} = (y_{1,1} \dots y_{1,m_i}, \dots, y_{n_y,1} \dots y_{n_y,m_i})^T$$

[11] Eigenautonomie: Änderung einer Regelgröße des Mehrgrößenregelsystems bleibt ohne Auswirkung auf die anderen Regelgrößen

Für die positiv definite Matrix $\mathbf{P}$ gilt weiterhin im einfachsten Fall[12]:

$$\mathbf{P} = \begin{pmatrix} \mathbf{P_1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{P_{n_y}} \end{pmatrix} \quad \mathbf{P_n} = \begin{pmatrix} p_{n_y,1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & p_{n_y,m_i} \end{pmatrix}$$

n_y	=	Anzahl Regelgrößen, $n_y \in \mathbb{N}$
m_i	=	Anzahl Elemente/Regelgröße, $m_i \in \mathbb{N}$
$(p_{n_y,1} \ldots p_{n_y,m_i})^T$	=	Vektor der Wichtungselemente einer Regelgröße
$(\mathbf{P_1} \ldots \mathbf{P_{n_y}})$	=	Matrizen der Wichtungselemente des Roboter-Teilsystems

Für die Führungs- und Regelgrößen von strukturvariablen Mehrgrößensystemen existieren, je nachdem in welchen Zustand sich das System gerade befindet, regelungs- und planungsspezifische Anforderungen (vgl. Abschnitt 2.2), die in der zusammengefassten Stabilitätsbeurteilung berücksichtigt werden müssen. Deshalb werden in dieser Arbeit Führungsgrößenintervalle definiert (Korridorkonzept), mit denen die Wichtung einer Regelgröße bei der Stabilitätsbeurteilung aufgaben- und damit strukturspezifisch angepasst werden kann. Ein Beispiel hierfür ist die Strukturumschaltung von Regler und zu regelndem Streckenmodell während eines Greifvorgangs bei Erreichen des Objektkontakts. So wird im Zustand der *freien Bewegung*, d. h. während der Annäherung an das zu greifende Objekt, durch ein sehr schmales Führungsgrößenintervall für die Gelenkwinkel und ein breites Führungsgrößenintervall für die Gelenkmomente ein geringes Abweichen von den Sollpositionen stark gewichtet (Bild 2.14 a). Dies ist sinnvoll, da der Regelungsalgorithmus in dieser Phase die Aufgabe hat, die von der Aufgabenplanung vorgegebenen Sollpositionen zu erreichen. Im *Kontaktfall* besteht das Ziel in der Einregelung vorgegebener Gelenkmomente zum Erreichen eines stabilen Griffs. Deshalb werden durch ein schmales Führungsgrößenintervall für die Gelenkmomente und ein breites Führungsgrößenintervall für die Gelenkwinkel bereits geringe Abweichungen der Gelenkmomente von den Sollmomenten als kritisch bewertet (Bild 2.14 b). Damit ist die Überwachung in der Lage, aufgabenspezifische Ziele und Schwerpunkte mit in die Stabilitätsbeurteilung einzubeziehen.

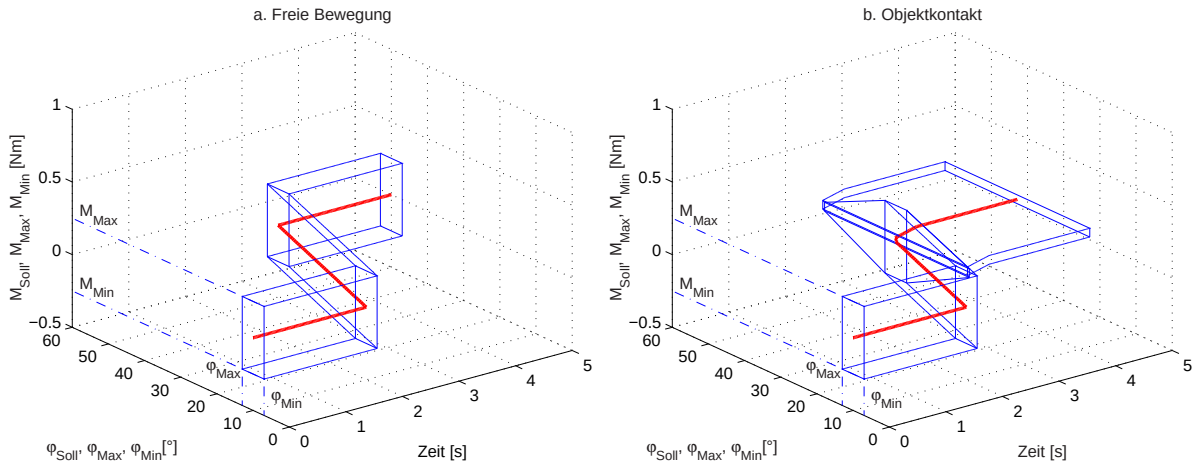


Bild 2.14: Umsetzung des neuen Korridorkonzepts am Beispiel eines Greifvorgangs zur aufgabenspezifischen Priorisierung der zusammengefassten Stabilitätsbeurteilung

[12] Positiv definite Matrizen mit besetzten Nebendiagonalelementen sind prinzipiell zulässig, werden hier aber nicht verwendet.

Es folgt für die von der Intervallgenerierung zu bestimmenden Vektoren der maximalen und minimalen Führungsgrößen:

$$\mathbf{w}_{\text{Max}}=(w_{Max_{1,1}} \dots w_{Max_{1,m_i}}, \dots, w_{Max_{n_y,1}} \dots w_{Max_{n_y,m_i}})^T$$
$$\mathbf{w}_{\text{Min}}=(w_{Min_{1,1}} \dots w_{Min_{1,m_i}}, \dots, w_{Min_{n_y,1}} \dots w_{Min_{n_y,m_i}})^T$$

Der Vektor $\mathbf{e}[k]$ berechnet sich jetzt in Abhängigkeit von $\mathbf{y}[k]$ und $\mathbf{w}_{\text{Max}}[k]$ bzw. $\mathbf{w}_{\text{Min}}[k]$:

$$\mathbf{e}[k] = \begin{cases} \mathbf{y}[k] - \mathbf{w}_{\text{Max}}[k] & \text{für} \quad \mathbf{y}[k] > \mathbf{w}_{\text{Max}}[k] \\ \mathbf{y}[k] - \mathbf{w}_{\text{Min}}[k] & \text{für} \quad \mathbf{y}[k] < \mathbf{w}_{\text{Min}}[k] \\ 0 & \text{sonst.} \end{cases} \tag{2.31}$$

Für die *Bewertungsfunktion* L ergibt sich

$$L[k] = L(\mathbf{e}[k]) = L(\mathbf{y}[k], \mathbf{w}_{\text{Max}}[k], \mathbf{w}_{\text{Min}}[k]) = \mathbf{e}[k]^T \cdot \mathbf{P} \cdot \mathbf{e}[k] \tag{2.32}$$

und für die *Korrektur der Bewertungsfunktion* L_{Extern} gilt

$$L_{\text{Extern}}[k+1] = L(\mathbf{w}[k+1], \mathbf{y}[k+1]) - L(\mathbf{w}[k+1] - \Delta\mathbf{w}[k], \mathbf{y}[k+1], \mathbf{w}_{\text{Max}}[k], \mathbf{w}_{\text{Min}}[k]). \tag{2.33}$$

Die restlichen Teilmodelle berechnen sich weiterhin mit den im Abschnitt Grundkonzept vorgestellten Gleichungen. Im Unterschied zum bisherigen Verfahren besteht das Ziel der Stabilitätsüberwachung nun jedoch darin, die Regelgrößen während der Laufzeit in vorgegebenen Führungsgrößenintervallen zu überwachen. Wenn sich die Regelgrößentrajektorien $\mathbf{y}[k]$ zu allen Zeitpunkten innerhalb der Führungsgrößenintervalle $\mathbf{w}_{\text{Max}}[k]$, $\mathbf{w}_{\text{Min}}[k]$ befinden, stuft die Überwachung die Situation als 'STABIL' ein. Bei Verlassen der Intervalle werden die Ursachen (z. B. veränderte Solltrajektorien) mit in die Beurteilung einbezogen. Der Regelkreis ist nach äußeren Störungen oder Änderungen der Intervallgrenzen wieder in Richtung des Sollverhaltens auszuregeln. Gelingt dies nicht, wird die Gesamtsituation als 'INSTABIL' eingeschätzt. Das Ergebnis der Beurteilung ist weiterhin ein Stabilitätsgrad S, mit Zuständen *zwischen* 0 ('INSTABIL') und 1 ('STABIL') [192, 197]. Bild 2.15 zeigt die detaillierte Struktur der regelbasierten Auswertung zur Stabilitätsüberwachung.

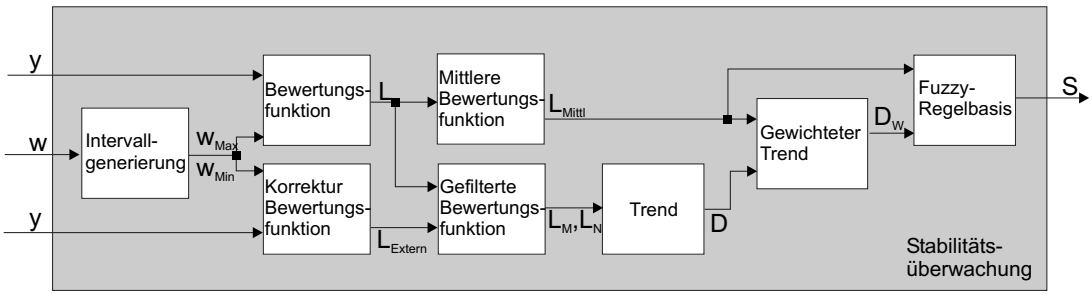


Bild 2.15: Modellstruktur der regelbasierten Auswertung zur Stabilitätsüberwachung

Eingangsgrößen der Stabilitätsüberwachung sind wiederum die Referenztrajektorien $\mathbf{w}$ aus der Aufgabenausführung (vgl. Abschnitt 2.2) sowie die Regelgrößentrajektorien $\mathbf{y}$. Ausgangsgröße bleibt der jetzt teilkomponenten- und aufgabenspezifische Stabilitätsgrad S.

2.3.1.4 Regelbasierte Auswertung zur Hardwareüberwachung

Ein ganzheitliches Überwachungssystem für humanoide Roboter sollte nach Abschnitt 2.3, neben der bereits vorgestellten Stabilitätsüberwachung, über Komponenten zur Überwachung der verfügbaren Hardwareressourcen verfügen. Damit ist ein solches System in der Lage, auftretende Hardwaredefekte frühzeitig zu erkennen, um weitere Schäden an der oftmals teuren Roboterhardware zu verhindern. Das Ziel der regelbasierten Auswertung zur Hardwareüberwachung ist damit, modellbasiert Aussagen über den aktuellen Zustand der mechatronischen Komponenten des Robotersystems zu treffen. Dies beinhaltet sowohl die Aktorik als auch die Sensorik des überwachten Systems.

Während des Betriebs treten in den mechatronischen Komponenten von humanoiden Robotersystemen die unterschiedlichsten Arten von Fehlern auf. Solche Fehler können z. B.

- durch extremen Leichtbau verursachte Verschleißerscheinungen, wie Risse und Brüche in den mechanischen Komponenten,
- durch äußere Hindernisse (Kollisionen) verursachte Risse, Brüche und Lecks in den mechanischen Komponenten,
- Antriebsfehler wie Pumpendefekte bei fluidischen Systemen, defekte Elektromotoren bei elektrisch angetriebenen Systemen oder Lagerschäden sowie
- Sensorfehler, verursacht durch Kabelbruch, Skalierungs- und Kalibrierungsfehler oder Hysteresen

sein.

Da der Fokus dieser Arbeit auf der Bereitstellung und Integration eines neuen ganzheitlichen Überwachungskonzepts für humanoide Roboter liegt, wird hier für die modellbasierte Hardwareüberwachung ein einfaches und leicht zu implementierendes Konzept vorgeschlagen. Die integrierte Hardwareüberwachungskomponente vervollständigt somit die vorgeschlagenen Grundstruktur des in Abschnitt 2.3 vorgestellten Gesamtkonzepts. Das hier verwendete quantitativ modellgestützte Verfahren ist jederzeit durch komplexere analytische [80, 94, 283, 286] und/oder heuristische [29, 95, 96, 105, 164] Verfahren zu erweitern oder zu ersetzen.

Der Umsetzung des regelbasierten Auswertungsmoduls stehen in dieser Arbeit zur Beurteilung des Hardwarezustands prinzipiell verschiedene Modell- und Sensorinformationen zur Verfügung. Dazu gehören z. B.:

- die Auswertung der in Abschnitt 2.3.1.2 berechneten Residuen,
- die Ausnutzung redundanter Sensorinformationen sowie
- Schlussfolgerungen aus gänzlich fehlenden Sensorinformationen.

Die in der Regelbasis hinterlegten Regeln sind von der Form:

$$\begin{array}{lllllll} \text{WENN} & \hat{\mathbf{y}}_1[k] > 0 & \text{UND/ODER} & \hat{\mathbf{z}}_1[k] > 0 & \text{DANN} & \textit{Fehler} = 1 \\ \text{WENN} & \hat{\mathbf{y}}_2[k] = 0 & \text{UND/ODER} & \hat{\mathbf{z}}_2[k] \neq 0 & \text{DANN} & \textit{Fehler} = 2 \\ \text{WENN} & \vdots & \text{UND/ODER} & \vdots & \text{DANN} & \vdots \end{array}$$

Allerdings müssen diese Regeln hardwarespezifisch, d. h. für jede zu überwachende Roboterkomponente einzeln, implementiert werden. Beispiele für die Umsetzung folgen in Kapitel 5.2.2.

2.3.2 Petrinetzbasiertes Konzept zur Problemlösung

2.3.2.1 Übersicht

Die im Abschnitt 2.2 skizzierte Vorgehensweise bei der Planung und Ausführung einer Aufgabe behandelt bisher hauptsächlich den Routineablauf. Die gezielte Integration von Problemlösungsstrategien zur erfolgreichen Behandlung von Ausnahmesituationen erfordert allerdings zusätzliche Plätze und Transitionen in den als Petri-Netze ausgeführten Roboteraufgaben [172]. Die entstehenden Petri-Netze werden somit wesentlich komplizierter (Bild 2.16).

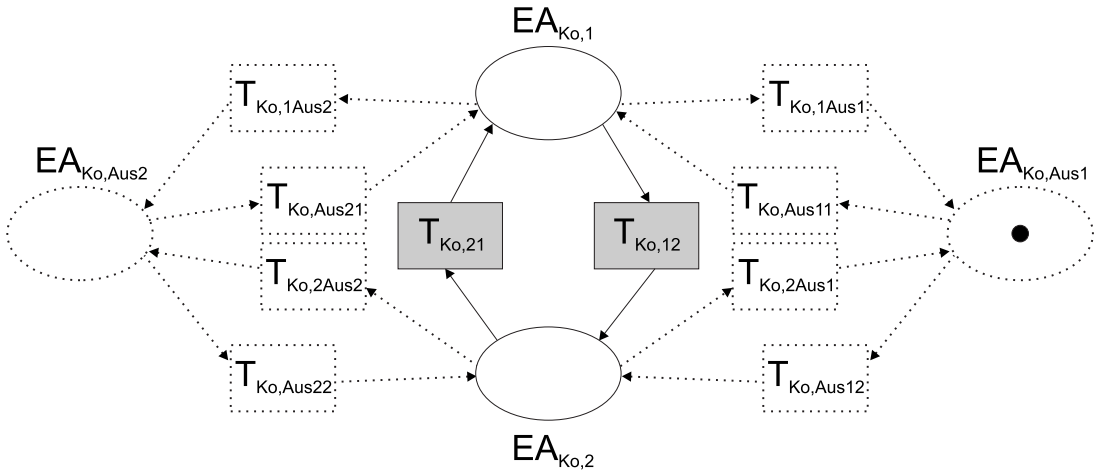


Bild 2.16: Ausschnitt aus dem platzberandeten Teilnetz der Aktion 'Aufmerksamkeitssteuerung' für die Halseinheit eines humanoiden Roboters (durchgezogen) mit zusätzlichen Plätzen und Transitionen zur Ausnahmesituationsbehandlung (gestrichelt, vgl. Beispiel Abschnitt 2.2.3, Bild 2.8 links)

In dem hier dargestellten einfachen Beispielnetz mit lediglich zwei Plätzen und zwei Transitionen führt bereits die Implementierung von zwei zusätzlichen Problemlösungsstrategien (Plätze $EA_{Ko,Aus1}$ und $EA_{Ko,Aus2}$ in Bild 2.16) zu einer Verdoppelung der Platzanzahl und zu einer Vervierfachung der Transitionenanzahl. Außerdem sind bei dieser Vorgehensweise nicht im Voraus zu planende Eingriffe des Benutzers in die Aufgabenausführung gar nicht oder nur schwer umzusetzen.

Aus diesen Gründen skizziert der vorliegende Abschnitt ein neues Konzept zur Ausnahmesituationsbehandlung mit Petri-Netzen, das diese Probleme löst. Die neue einheitliche Überwachungsstrategie besteht darin,

- die Problemlösung für eine Roboteraufgabe durch je ein Modul zur *Verwaltung* und zur *Auswertung* zu organisieren sowie
- alle Plätze von Elementaraktionen ($EA_{i,u}$) mit zusätzlichen Funktionen für eine platzspezifische Ausnahmebehandlung zum Unterbrechen und Abbrechen von Aufgaben auszustatten.

Im Weiteren erfolgt nun die detaillierte Beschreibung der *Verwaltung* (Abschnitt 2.3.2.2) und der *regelbasierten Auswertung* (Abschnitt 2.3.2.3) sowie deren Zusammenspiel beim Auftreten von Ausnahmesituationen.

2.3.2.2 Verwaltung

Die Verwaltung der Aufgabenausführung erfolgt auf der mittleren Ebene der Roboter-Steuerungsarchitektur durch ein weiteres Petri-Netz, das als Verwaltungsnetz bezeichnet wird. Bild 2.17 zeigt die Umsetzung des Verwaltungsnetzes mit den zugehörenden Plätzen, Verbindungen und Transitionen.

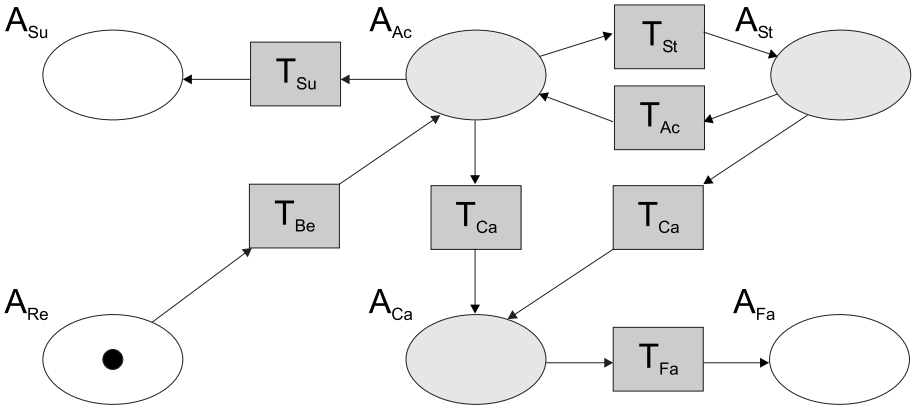


Bild 2.17: Verwaltungsnetz mit Plätzen Ready A_{Re}, Active A_{Ac}, Stopped A_{St}, Success A_{Su}, Cancelled A_{Ca} und Failed A_{Fa} (hellgrau hinterlegte Plätze entsprechen einer aktivierten Aktionsfolge AF)

Jedem die Aufgabe beschreibenden Koordinierungsnetz, d. h. jeder Aktionsfolge AF, wird zur Ausnahmebehandlung ein eigenes Verwaltungsnetz zugeteilt. Die Plätze eines Verwaltungsnetzes stehen für eine ausführungsbereite (Ready A_{Re}), eine routinemäßig abzuarbeitende (Active A_{Ac}), eine zeitweilige zu unterbrechende (Stopped A_{St}), eine irreversible abgebrochene aber noch aktivierte (Cancelled A_{Ca}), eine erfolgreich durchgeführte und beendete (Success A_{Su}) oder eine nicht erfolgreich durchgeführte und beendete (Failure A_{Fa}) Aktionsfolge AF. Der Vorteil dieser Struktur besteht darin, dass ein Verwaltungsnetz keine Detailinformationen über die auszuführenden Aufgaben benötigt. Das Verwaltungsnetz hat somit immer die gleiche Netzstruktur, die nicht von den zu verwaltenden Aktionsfolgen AF in Form von Koordinierungsnetzen und deren aufgabenspezifischen Netztopologie abhängt. In Tabelle 2.5 sind die Bedeutungen der Plätze und Transitionen des Verwaltungsnetzes nochmals zusammengefasst.

Die Ausführung einer Aktionsfolge AF wird von der Aufgabenkoordination initiiert. Notwendige Voraussetzungen sind eine ausführungsbereite Aktionsfolge AF (A_{Re} im Verwaltungsnetz der Aufgabe ist markiert) und die Verfügbarkeit der benötigten Hardwareressourcen (vgl. Abschnitt 2.2.2.2). Zusätzlich kann die Aufgabenplanung weitere initiale Bedingungen prüfen. Diese Bedingungen ergeben zusammen die Transition T_{Be} im Verwaltungsnetz. Das Verwaltungsnetz setzt bei Durchlaufen dieser Transition die im Aufgabenwissen hinterlegte Startmarkierung (initiale Markenbelegung $\mathcal{M}_0$) im zugehörigen Koordinierungsnetz der Aktionsfolge AF.

Erkannte Ausnahmesituationen aktivieren je nach Schweregrad die Transitionen T_{St} bzw. T_{Ca}. Sie bewirken somit für das jeweilige Verwaltungsnetz einen Markenfluss auf den Stopped- (A_{St}) bzw. Cancelled-Platz (A_{Ca}). Die Ereignisse zum Schalten der Transitionen in den Verwaltungsnetzen stammen aus den Konklusionen der Regeln im Modul *Auswertung*. Beispiele für solche Ereignisse sind nicht zu planende Benutzereingriffe, Änderungen des auszuführenden Plans durch die autonome Planungskomponente der Roboter-Steuerungsarchitektur oder modellbasiert erkannte Fehler (Abschnitt 2.3.2.3). Die Markenbelegung des aufgabenspezifischen Verwaltungsnetzes wird der zugehören-

Platz	Bedeutung
A_{Re}	ausführungsbereite Aktionsfolge (Ready)
A_{Ac}	routinemäßige Abarbeitung der Aktionsfolge (Active)
A_{St}	zeitweilige unterbrochene Aktionsfolge (Stopped)
A_{Ca}	irreversible abgebrochene aber noch aktivierte Aktionsfolge (Cancelled)
A_{Su}	erfolgreich durchgeführte Aktionsfolge (Success)
A_{Fa}	nicht erfolgreich durchgeführte Aktionsfolge (Failure)
Transition	**Bedeutung**
T_{Be}	Abarbeitung der Aktionsfolge gestartet
T_{Ac}	Abarbeitung der Aktionsfolge erfolgt routinemäßig
T_{St}	Abarbeitung der Aktionsfolge unterbrochen
T_{Ca}	Abarbeitung der Aktionsfolge irreversibel abgebrochen
T_{Su}	Abarbeitung der Aktionsfolge erfolgreich durchgeführt
T_{Fa}	Abarbeitung der Aktionsfolge nicht erfolgreich durchgeführt

Tabelle 2.5: Bedeutung der Plätze und Transitionen des Verwaltungsnetzes

den Aktionsfolge AF mitgeteilt, die auf der unterlagerten Ebene der Elementaraktionen $EA_{i,u}$ virtuell Funktionen für

- eine routinemäßige Abarbeitung der Aufgabe (Markierung im Verwaltungsnetz auf A_{Ac}, Ziel: planmäßige Abarbeitung),
- eine zeitweilige Unterbrechung der Aufgabe (Markierung im Verwaltungsnetz auf A_{St}, Ziel: sichere Unterbrechung, um eine spätere Fortsetzung möglichst an der gleichen Markierung zu ermöglichen) oder
- einen Abbruch der Aufgabe (Markierung im Verwaltungsnetz auf A_{Ca}, Ziel: sicheres irreversibles Abbrechen)

zugewiesen bekommt.

Der grundlegende Unterschied zwischen dem Cancelled-Platz A_{Ca} und dem Failure-Platz A_{Fa} besteht dabei darin, dass eine abgebrochene Aktionsfolge (A_{Ca} belegt) weiterhin ein aktiviertes Koordinierungsnetz besitzt und somit in der Lage ist, passiv auf äußere Ereignisse (durch Schalten entsprechender Transitionen im Koordinierungsnetz) zu reagieren. Im Gegensatz dazu wird bei Belegung des Platzes A_{Fa} das zugehörige Koordinierungsnetz gelöscht und die Weiterführung der Aktionsfolge beendet.

Das Zusammenwirken von Koordinierungs- und Verwaltungsnetz ergibt ein deutlich komplexeres Petri-Netz, das allerdings nicht *explizit* implementiert werden muss. Jede Elementaraktion $EA_{i,u}$ wird nun virtuell durch eine Aktion A mit drei Unterstellen $EA_{(i,u)Ac}$, $EA_{(i,u)St}$ und $EA_{(i,u)Ca}$ und dazugehörenden Transitionen ersetzt. Dabei entspricht die Unterstelle $EA_{(i,u)Ac}$ der ursprünglichen, die Aufgabe im Routinebetrieb beschreibenden, Elementaraktion $EA_{i,u}$. Die zusätzlichen Unterstellen $EA_{(i,u)St}$ und $EA_{(i,u)Ca}$ dienen zur zeitweilige Unterbrechung und zum irreversiblen Abbruch der der Aufgabe.

Bild 2.18 zeigt beispielhaft die virtuelle Verknüpfung der drei Unterstellen des Verwaltungsnetzes mit der als platzberandetes Teilnetz innerhalb einer Aktionsfolge AF umgesetzten Aktion 'Aufmerksamkeitssteuerung' der Halseinheit des Roboters (vgl. Bild 2.8, links).

Die hier dargestellte Situation zeigt den Fall einer initialen Markenbelegung der Aktion von

$$\mathcal{M}_0 = \{(\mathbf{EA}_{(\mathbf{Ko},\mathbf{1})\mathbf{Ac}}, 1), (\mathbf{EA}_{(\mathbf{Ko},\mathbf{2})\mathbf{Ac}}, 0)\}. \tag{2.34}$$

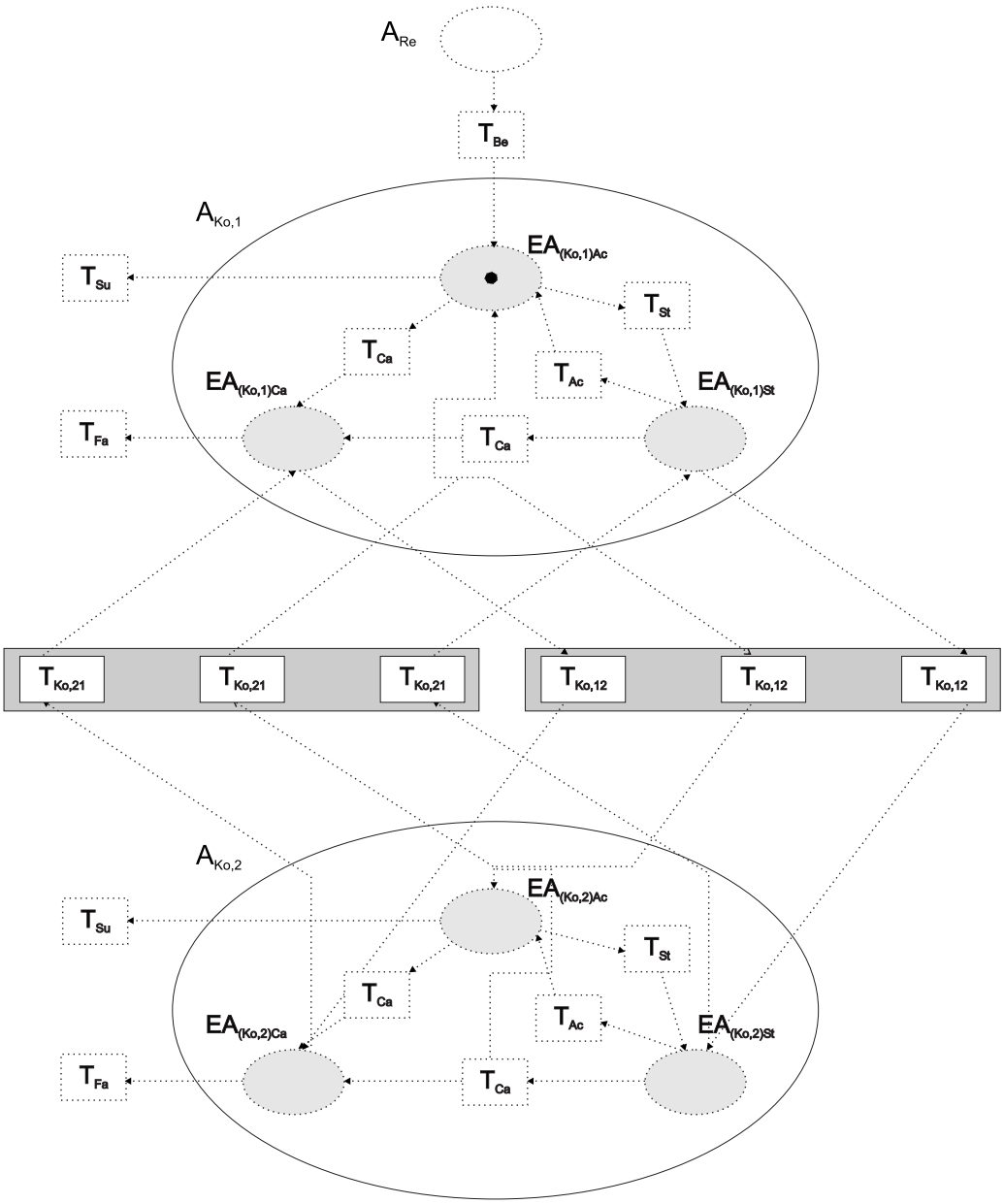


Bild 2.18: Verknüpfung der drei Unterstellen des Verwaltungsnetzes $EA_{(i,u)Ac}$, $EA_{(i,u)St}$ und $EA_{(i,u)Ca}$ (gestrichelt) mit der Aktion 'Aufmerksamkeitssteuerung' des Kopfes (vgl. Bild 2.8, links)

Dadurch wird beim Durchlaufen der Start-Transition T_{Be} der Platz A_{Re} des Verwaltungsnetzes mit dem Platz $EA_{(Ko,1)Ac}$ der Aktion 'Aufmerksamkeitssteuerung' verbunden. Bei einer initialen Markenbelegung von

$$\mathcal{M}_0 = \{(\mathbf{EA_{(Ko,1)AC}}, 0), (\mathbf{EA_{(Ko,2)Ac}}, 1)\}, \tag{2.35}$$

ergibt sich stattdessen beim Durchlaufen der Start-Transition T_{Be} eine Verbindung der Plätze A_{Re} und $EA_{(Ko,2)Ac}$. Die initiale Verbindung der Start-Transition T_{Be} erfolgt dabei ausschließlich mit der jeweiligen Unterstellen $EA_{(i,u)Ac}$ der entsprechenden Aktion.

Durch das virtuelle Hinzufügen der drei Unterstellen zur Ausnahmebehandlung werden aus den ursprünglichen Elementaraktionen $EA_{Ko,1}$ ('Objekt bildbasiert verfolgen') bzw. $EA_{Ko,2}$ ('Objekt suchen') die "virtuellen" Aktionen $A_{Ko,1}$ ('Objekt bildbasiert verfolgen mit Ausnahmebehandlung') bzw. $A_{Ko,2}$ ('Objekt suchen mit Ausnahmebehandlung'). Die ursprüngliche Aktion 'Aufmerksamkeitssteuerung' wird zur "virtuellen" Aktionsfolge 'Aufmerksamkeitssteuerung mit Ausnahmebehandlung'.

Ist die Ausführung einer Aktionsfolge AF unterbrochen (Stopped-Platz $EA_{(i,u)St}$ markiert), sollte ein Robotersystem in der Lage sein, seine derzeitige Situation, autonom oder mit Hilfe des Benutzers bzw. des Teleoperators, zu analysieren und zu bewerten. Aus diesem Grund enthalten die meisten Elementaraktionen $EA_{i,u}$ bei Unterbrechung (d. h. $EA_{(i,u)St}$) zunächst einfache Wartefunktionen (Elementaraktion 'Warte' aus Abschnitt 2.2.1.2). Allerdings ist bei bestimmten Elementaraktionen aus Sicherheitsgründen eine andere Funktion sinnvoll (z. B. halte gegriffenes Objekt gerade, setzte Objekt definiert ab usw.).

Im Anschluss daran erfolgt dann die Auswahl der entsprechenden Problemlösungsstrategie, wiederum durch die Planungskomponente des Robotersystems, den Benutzer oder den Teleoperator. Mögliche Strategien sind nach Abschnitt 2.3 z. B. der Start eines Klärungsdialogs mit dem Benutzer, der Start einer Problembehandlung durch Telemanipulation, das verlangsamte Ausführen der Aktionsfolge, das annähernd planmäßiges Beenden der Aktionsfolge und Ignorieren des Problems oder der Übergang in einen sicheren Zustand. Mit der Auswahl einer geeigneten Problemlösungsstrategie wird die Transition T_{Ac} im Verwaltungsnetz aktiviert und die Ausführung der Aktionsfolge AF fortgesetzt (Active-Platz $EA_{(i,u)Ac}$ markiert).

Neben diesen bewusst einzusetzenden Strategien ist es zudem sinnvoll, biologisch motivierte Problemlösungsstrategien (Reflexe) in das hier erstmals vorgeschlagene Überwachungskonzept zu integrieren. Physiologisch betrachtet sind Reflexe *automatische*, weitgehend *stereotype* motorische Reaktionen eines Organismus auf äußere Reize [241]. Demzufolge sind Reflexe nicht in den routinemäßigen Entwurfsprozess der Aufgabenplanung zu integrieren, sondern als nicht planbare Ausnahmesituationen zu behandeln. Beim Eintreten solcher nicht zu planender Ereignisse, wie z. B. Kollisionen oder Verluste von gegriffenen Objekten, werden bei der aktiven Ausführung von Elementaraktionen, wie 'Fahren', 'Verfolgen' oder 'Zugreifen', anstelle der ursprünglich geplanten Funktionen, reflexartige und von der Planungskomponente des Robotersystems nicht bewusst zu steuernde, der ursprünglichen Bewegung entgegen gerichtete Ausweichbewegungen ausgeführt. Führt das System dagegen die Elementaraktion 'Warten' aus, dann muss bei einer Kollision, eine der Kraftrichtung entgegen gerichtete Ausweichbewegung initiiert werden. Dies bedeutet einen fundamentalen Unterschied zu den bisher beschriebenen Problemlösungsstrategien, bei denen eine eingehende Situationsbewertung (auf dem Stopped-Platz $EA_{(i,u)St}$) durch die Aufgabenplanung oder den Benutzer der letztendlichen Problemlösung vorangeht. Die Entscheidung, ob nun eine unbewusste Reflexbewegung oder eine bewusste Problemlösungsstrategie angewendet wird, hängt dabei nur von der Art der eintretenden Ausnahmesituationen ab.

Beide Arten von Problemlösungsstrategien lassen sich allerdings in Form von Elementaraktionen ($EA_{i,u}$) im Aufgabenwissen des Robotersystems speichern und sind somit einfach in das neue Überwachungskonzept zu integrieren. Tabelle 2.6 gibt einen Überblick über Beispiele solcher Elementaraktionen, gegliedert in zu implementierende Trajektorienart, benötigtem Regelungsprinzip, angesteuerter Roboter-Teilkomponente sowie dazugehörenden Roboteraufgaben.

Zusätzlich können Plätze fest programmierte Informationen für eine aufgabenpezifische regelbasierte Auswertung enthalten. Eine Aktivierung der Transition T_{Ac} nach Beseitigung des Problems erfolgt derzeit hauptsächlich extern durch Befehl des Benutzers oder des Teleoperators (siehe vorheriger Abschnitt). In einfachen Fällen (z. B. erkannte Tasse nach einem Dialog) kann diese Transition auch durch den Roboter selbst mittels der Planungskomponente der Aufgabenplanung aktiviert werden. Allerdings

Elementar-aktion	Trajektorienart	Regelungsprinzip	Roboter-Teilkomponente	Beispiele
Ausweichen	Rampe, Sprung menschl. Bewegung	Positionsregelung (Folge)	Arme, Greifer, Plattform, Kopf	Winken, Nicken Zeigegesten
Stoppen	Sollwert konstant	Positionsregelung (Festwert)	Arme, Greifer, Plattform, Kopf	alle Roboter Bewegungen
Fahren verlangsamen	Rampe, Sprung menschl. Bewegung	Positionsregelung (Folge)	Arme, Greifer, Plattform, Kopf	alle Roboter Bewegungen
Ignorieren	-	Positions- und Kraftregelung	Arme, Greifer Plattform, Kopf	alle Roboter Bewegungen
Kräften ausweichen	Rampe, Sprung menschl. Bewegung	Nullkraftregelung	Arme	Zwei-Arm-Manipulation
Kontakt ausweichen	Rampe, Sprung menschl. Bewegung	Kraftregelung	Arme, Greifer	alle Roboter Bewegungen

Tabelle 2.6: Beispiele für bewusste und unbewusste Problemlösungsstrategien von humanoiden Robotern in Form von Elementaraktionen [289]

können zwischenzeitliche Markenflüsse durch äußere Einflüsse (aktivierte Transitionen in einem Koordinierungsnetz) auftreten (beispielsweise bei Kollisionen, Verlust von gegriffenen Objekten usw.). Damit reagiert eine Aktionsfolge auch bei unterbrochenen oder abgebrochenen Aufgaben auf veränderte äußere Situationen, solange sich das zugehörige Verwaltungsnetz noch nicht in A_{Fa} befindet. Die finale Markenbelegung im Koordinierungsnetz der Aktionsfolge, die für die erfolgreiche Abarbeitung der Aufgabe steht, aktiviert die Transition T_{Su} im Verwaltungsnetz. Die Aufgabenkoordination gibt nach der erfolgreichen Abarbeitung T_{Su} oder nach einem abgeschlossenen Abbruch T_{Fa} die Ressourcen wieder zurück. Bei einer Markierung des Stopped-Platzes T_{St} werden demgegenüber die Ressourcen *nicht* zurückgegeben, da die entsprechende Roboterkomponente, trotz Unterbrechung der planmäßigen Ausführung, weiterhin aktiv bzw. passiv angesteuert wird (vgl. Abschnitt 2.2.2.2).

Grundsätzlich können die übergeordnete Aufgabenplanung sowie der Benutzer alle Transitionen im Verwaltungsnetz extern aktivieren. Damit wird insbesondere bei T_{Su}, T_{Fa} die volle Verantwortung für den weiteren Ablauf übernommen, während bei T_{St}, T_{Ac}, T_{Ca} die Aktionsfolge AF noch aktiv eingreift. Die Transitionen für diese internen Übergänge entsprechen den Transitionen des Verwaltungsnetzes. Das Verlassen eines platzberandeten Teilnetzes einer Aktionsfolge AF geschieht durch die ursprünglich definierte ausgangsseitige Transition[13]. Ausgangsseitige Transitionen können auch koordinierend sein (Bild 2.18), hier existierenden dann drei paarweise Transitionen mit gleicher Schaltbedingung. Zusätzlich kann jeder Platz einer "virtuellen" Aktion aus der Unterstelle $EA_{(i,u)Ac}$ bzw. $EA_{(i,u)Ca}$ verlassen werden, wenn im Verwaltungsnetz die Transition T_{Su} bzw. T_{Fa} aktiviert wird (Bild 2.18).

2.3.2.3 Regelbasierte Auswertung zur Problemlösung

Bei der Abarbeitung einer Aktionsfolge AF können verschiedene Situationen eintreten, welche den routinemäßigen Ablauf der Ausführung stören:

- unerwünschte Ereignisse (z.B. Kollisionen in der Annäherungsphase an ein Objekt),
- Nichterreichen von gewünschten Ereignissen (z. B. ausbleibender Objektkontakt, Nichterreichen gewünschter Positionen, ...),

[13] Hier sind prinzipiell auch mehrere Transitionen möglich.

- Stabilitätsprobleme von unterlagerten Regelkreisen,
- Ausfall von Hardwarekomponenten (z. B. Antriebe oder Sensoren),
- nicht realisierbare Pläne sowie
- Änderung von Plänen durch den menschlichen Benutzer während der Abarbeitung einer Aktionsfolge.

Die regelbasierte Auswertung verarbeitet Informationen aus den einzelnen Plätzen der Aktionsfolge AF und den Komponenten der modellbasierten Fehlerdetektion. Die hierfür in der Regelbasis hinterlegten Regeln sind von der Form

WENN *Prämisse* DANN *Transition(en) aktivieren (=1) bzw. blockieren (-1).*

Die Konklusion beeinflusst somit nur Transitionen im Verwaltungsnetz. Diese Form erzwingt eine strenge Dekomposition von Fehlerdetektion und -isolation (modellbasierte Fehlerdiagnose Abschnitt 2.3.1) und Problemlösung (auszuführende Funktionen im Koordinierungsnetz der Aktionsfolge AF).

In der Prämisse befinden sich dabei Informationen wie:

- die Zeitüberschreitung für den Aufenthalt einer Marke auf einem Platz $t > t_{Ai,max}$,
- erkannte Stabilitätsprobleme für mindestens eine aktive Regelung (Algorithmen siehe Abschnitt 2.3.1.3) [197],
- regelbasiert erkannte Hardwarefehler (Algorithmen siehe Abschnitt 2.3.1.4) [105],
- unerwartete Kollisionen,
- erkannte unzulässige Wettläufe in einem Petri-Netz mit Verzweigungen (gleichzeitiges Aktivieren mehrerer Transitionen an der Ausgangsseite eines markierten Platzes im Koordinierungsnetz),
- Markenbelegungen bei (fest programmierten) Plätzen, die Ausnahmesituationen kodieren,
- die finale Markenbelegungen, die für eine sicher abbrechbare Aufgabenausführung stehen sowie
- die finale Markenbelegung, die für die erfolgreiche Abarbeitung der Aufgabe steht.

Damit ergeben sich Regeln wie:

WENN	*Stabilitätsgrad* $S < 0.5$	DANN	$T_{St} = 1$ und $T_{Ca} = -1$ und $T_{Su} = -1$
WENN	*Fehlerzustand* $\boldsymbol{F}[k] \neq 0$	DANN	$T_{Ca} = 1$ und $T_{St} = -1$ und $T_{Su} = -1$
WENN	*Markenbelegung final*	DANN	$T_{Su} = 1$ und $T_{St} = -1$ und $T_{Ca} = -1$
WENN	$\vdots$	DANN	$\vdots$

Bei Widersprüchen von Konklusionen haben von der Aufgabenplanung, d. h. von der übergeordneten Ebene der Roboter-Steuerungsarchitektur, übermittelte Aktivierungen bzw. Blockierungen von Transitionen Vorrang. Tabelle 2.7 zeigt die sich aus dieser Forderung ergebenden Vorrangregeln.

Durch diese Einteilung kann die übergeordnete Aufgabenplanung falls nötig, autonom bzw. durch den Benutzer mit Hilfe des Dialog-Managers (vgl. Bild 2.1), auch bei Widersprüchen, alle Transitionen im Verwaltungsnetz extern aktivieren. Damit wird die volle Verantwortung für den weiteren Ablauf der Aufgabenausführung an die übergeordneten Ebenen der Roboter-Steuerungsarchitektur delegiert. Weiterhin ist eine zusätzliche Abstufung der Priorisierung denkbar. So ist es möglich, neben den in diesem Abschnitt definierten Vorrangregeln für Aufgabenplanung und modellbasierter Fehlerdiagnose weitere Regeln vorzusehen, die Widersprüche zwischen autonomer Aufgabenplanung und externen Benutzerkommandos regeln. Allerdings liegt dabei die höchste Prioritätsstufe eindeutig beim menschlichen Benutzer.

	Aufgabenplanung			
modellbasierte Fehlerdiagnose	0	1	-1	1 UND -1
0	0	1	0	0
1	1	1	0	0
-1	0	1	0	0
1 UND -1	0	1	0	0

Tabelle 2.7: Vorrangregeln für Transitionen (0: nicht aktiviert, 1: aktiviert) im Verwaltungsnetz bei regelbasierter Aktivierung (1), Blockierung (-1), Widersprüchen (1 und -1: mindestens eine aktivierende und mindestens eine blockierende Regel) oder fehlenden Aussagen (0) von Transitionen durch die Aufgabenplanung

2.3.3 Beispiel

Das Funktionsprinzip des neuen Überwachungskonzepts wird wiederum beispielhaft anhand der beiden Aktionen 'Aufmerksamkeitssteuerung' und 'koordiniertes Annähern' des in Abschnitt 2.2.3 vorgestellten 'Greif- und Transportvorgangs' gezeigt (Tabelle 2.3 grau hinterlegt).

Durch das virtuelle Hinzufügen von drei Unterstellen zur Ausnahmebehandlung, werden aus den ursprünglichen Elementaraktionen $EA_{Ko,1}$ ('Objekt bildbasiert verfolgen') bzw. $EA_{Ko,2}$ ('Objekt suchen') in Bild 2.8 (links), die "virtuellen" Aktionen $A_{Ko,1}$ ('Objekt bildbasiert verfolgen mit Ausnahmebehandlung') bzw. $A_{Ko,2}$ ('Objekt suchen mit Ausnahmebehandlung'). Die ursprüngliche Aktion 'Aufmerksamkeitssteuerung' wird zur "virtuellen" Aktionsfolge 'Aufmerksamkeitssteuerung mit Ausnahmebehandlung' in Bild 2.18. In Tabelle 2.8 sind die Aktionen der sich ergebenden Aktionsfolge als Plätze mit möglichen unterlagerten Callback-Funktionen ("virtuelle" Unterstellen $EA_{(i,u)Ac}$, $EA_{(i,u)St}$ und $EA_{(i,u)Ca}$) sowie die entsprechenden Transitionen erläutert.

	Callback-Funktionen		
Platz	Active ($EA_{(i,u)Ac}$)	Stopped ($EA_{(i,u)St}$)	Cancelled ($EA_{(i,u)Ca}$)
$A_{Ko,1}$	Objekt bildbasiert verfolgen	Ignorieren (+ Dialog)	Stopp (+ Dialog)
$A_{Ko,2}$	Objekt suchen	Ignorieren (+ Dialog)	Stopp (+ Dialog)

Transition	Bedeutung
$T_{Ko,12}$	Objekt nicht lokalisiert
$T_{Ko,21}$	Objekt lokalisiert

Tabelle 2.8: Plätze und Transitionen der Aktionsfolge 'Aufmerksamkeitssteuerung mit Ausnahmebehandlung' mit beispielhaft zugeordneten Callback-Funktionen zur Ausnahmebehandlung

Analog werden aus den ursprünglichen Elementaraktionen $EA_{1..n_G,1}$ bzw. $EA_{1..n_G,2}$ der Aktion 'koordiniertes Annähern' durch das virtuelle Hinzufügen von drei Unterstellen zur Ausnahmebehandlung, die "virtuellen" Aktionen $A_{(1..n_G,1}$ ('Annähern mit Ausnahmebehandlung') bzw. $A_{1..n_G,2}$ ('Warten mit Ausnahmebehandlung'). Die ursprüngliche Aktion 'koordiniertes Annähern' wird zur "virtuellen" Aktionsfolge 'koordiniertes Annähern mit Ausnahmebehandlung'. Tabelle 2.9 zeigt wiederum die Aktionen der sich auf diese Weise ergebenden Aktionsfolge als Plätze mit möglichen Callback-Funktionen sowie die entsprechenden Transitionen.

	Callback-Funktionen		
Platz	Active ($EA_{(i,u)Ac}$)	Stopped ($EA_{(i,u)St}$)	Cancelled ($EA_{(i,u)Ca}$)
$A_{1..n_G,1}$	Annähern	Warten (+Dialoganforderung)	Stopp (+ Dialog)
$A_{1..n_G,2}$	Warten	Ignorieren	Kraft ausweichen

Transition	Bedeutung
$T_{n_1,12}$	Objektkontakt Finger n_1

Tabelle 2.9: Plätze und Transitionen der Aktionsfolge 'koordiniertes Annähern mit Ausnahmebehandlung' mit beispielhaft zugeordneten Callback-Funktionen zur Ausnahmebehandlung

Die regelbasierte Auswertung in der Aufgabenverwaltung enthält für dieses Beispiel unter anderem die folgenden Regeln:

R_1 :	WENN	$t > t_{Ai,max}$	DANN	$T_{St} = 1$	und	$T_{Ca} = -1$	und	$T_{Su} = -1$
R_2 :	WENN	$S < 0.5$	DANN	$T_{St} = 1$	und	$T_{Ca} = -1$	und	$T_{Su} = -1$
R_3 :	WENN	$\boldsymbol{F}[k] \neq 0$	DANN	$T_{Ca} = 1$	und	$T_{St} = -1$	und	$T_{Ac} = -1$
R_4 :	WENN	externer Abbruch	DANN	$T_{Ca} = 1$	und	$T_{St} = -1$	und	$T_{Su} = -1$
R_5 :	WENN	externer Stopp	DANN	$T_{St} = 1$	und	$T_{Ca} = -1$	und	$T_{Su} = -1$
R_6 :	WENN	Regelziel nicht erreichbar	DANN	$T_{Ca} = 1$	und	$T_{St} = -1$	und	$T_{Ac} = -1$

Die erste Regel erkennt ein unerwünschtes Verharren in bestimmten Plätzen durch eine Zeitüberschreitung. Wenn z. B. der Suchvorgang ($A_{Ko,2}$ markiert) länger als $t_{Ai,max} = 3$ s dauert, wird die Transition T_{St} markiert und ein Dialog initiiert (Tabelle 2.8). Ist der Klärungsdialog erfolgreich, kann die Aktionsfolge mit der ausgewählten Problemlösungsstrategie fortgesetzt werden (T_{Ac} aktiv). Weiterhin sollte z. B. $A_{1..n_G,1}$ nicht länger als $t_{Ai,max} = 3 \ldots 4$ s dauern (d. h. alle am Griff beteiligten Finger haben geplanten Kontaktzustand erreicht). Bleibt der erwartete Markenfluss aus, wird das Verwaltungsnetz gestoppt (T_{St} aktiv) und eine geeignete Problemlösungsstrategie kann ausgewählt werden.

Falls die Basis-Regler der aktiven Elementaraktionen nicht in der Lage sind, die Vorgaben der Trajektoriengenerierung umzusetzen, gibt die jeweilige Stabilitätsüberwachung Stabilitätsgrade $S \leq 0.5$ aus. Dies wertet die regelbasierte Auswertung als Ausnahmesituation und die Transition T_{St} wird aktiviert. Mögliche Problemlösungsstrategien sind dabei der Übergang in einen sicheren Zustand oder das verlangsamte Fortsetzen der geplanten Aktionsfolge. Liefert die jeweilige Hardwareüberwachung Fehlerzustandsvektoren $\boldsymbol{F}[k] \neq 0$ zurück, bedeutet dies einen modellbasiert diagnostizierten Hardwaredefekt. Für den Kopf können das z. B. defekte Elektromotoren oder Winkelencoder und für den Robotergreifer z. B. eine defekte Antriebseinheit (Pumpe) oder Leckageverluste in den Leitungen sein. Das Auftreten dieser u. U. schwerwiegenden Fehler verursacht sicherheitshalber einen gezielten Abbruch der Aktionsfolge (T_{Ca} aktiv).

Die vierte und die fünfte Regel beenden bzw. unterbrechen die Ausführung der Aktionsfolge z. B. nach einem Benutzereingriff (externes Abbruch- bzw. Stoppkommando). Die sechste Regel entspricht einem finalen Abbruch aufgrund eines Misserfolgs. Regeln für ein erfolgreiches Beenden (Konklusion T_{Su}) existieren erst für die finale Markenbelegung des Koordinierungsnetzes nach erfolgreichem Abschluss der kompletten Aktionsfolge. Weitere Dialoge (z.B. der Roboter schafft es nicht, das Objekt anzuheben usw.) können nun regelbasiert bei jedem erkannten spezifischen Stopped-Zustand initiiert werden und müssen nicht extra geplant werden. Je nach Ergebnis des Dialogs wird die Elementaraktion fortgesetzt oder definiert abgebrochen.

2.4 Diskussion

In diesem Kapitel wurde erstmals gezeigt, wie die aufgabenausführungsrelevanten Komponenten einer hybriden Steuerungsarchitektur mit Hilfe von hierarchischen Petri-Netzen umgesetzt werden können. Hierzu wurde zunächst ein Konzept vorgestellt, um Bausteine komplexer Handlungen H mit Hilfe von elementaren Basisoperationen (Elementaraktionen $EA_{i,u}$) im Aufgabenwissen der Robotersystems zu speichern. Im Anschluss erfolgte die Erweiterung des Konzepts zur systematisierten Beschreibung von Handlungsabläufen zur Lösung komplexer Aufgaben mit hierarchisch strukturierten Petri-Netzen. Mit der in diesem Kapitel erstmals präsentierten Systematik steht somit ein Methodenapparat zur Verfügung, um komplexe Handlungsabläufe strukturiert abzubilden. Dies erfolgt derzeit allerdings noch manuell durch den Entwickler. Zukünftig wird dieser Schritt jedoch, mit Hilfe modellbasierter Optimierungs- und Lernalgorithmen, autonom von der Planungskomponente der Roboter-Steuerungsarchitektur ausgeführt. Ein weiterer wichtiger Aspekt des vorgeschlagenen flexiblen Konzepts ist, dass dieselbe Roboteraufgabe entweder als eine zusammenhängende Aktionsfolge AF (dabei entsteht *ein* komplexes Koordinierungsnetz) oder als nacheinander auszuführende, separate Aktionsfolgen AF (hierbei entstehen mehrere sequentiell und/oder parallel auszuführende Koordinierungsnetze) implementiert werden können. Vorteile der zweiten Variante liegen bei der größeren Flexibilität in Hinblick auf die Änderung der ursprünglich geplanten Handlungsabläufe innerhalb einer Aktionsfolge AF während der Laufzeit, beispielsweise durch Auftreten von Ausnahmesituationen (vgl. Kapitel 5) und der Möglichkeit zur verteilten Entwicklung. Nachteilig ist, dass im Vergleich zu Variante 1, immer eine größere Anzahl an Schnittstellen zu definieren sind, was wiederum das Fehlerpotential erhöht.

Weiterhin wurde ein neues, vollständig in die vorgeschlagene Steuerungsarchitektur integriertes Konzept zur online Überwachung von Bewegungsvorgängen bei humanoiden Robotern mit Petri-Netzen vorgestellt. Die grundlegende neue Idee ist dabei, die Durchführung der geplanten Handlung auf der unteren Ebene der hierarchischen Roboter-Steuerungsarchitektur modellbasiert zu überwachen und mit einem neuen petrinetzbasierten Konzept zur einfachen und systematischen Behandlung von erkannten Ausnahmesituationen auf der mittleren Ebene zu koppeln. Dazu wurden zunächst allgemeine und roboterspezifische Anforderungen an ein solches Überwachungskonzept definiert und ein Lösungskonzept präsentiert. Das entworfene Lösungskonzept beinhaltet Komponenten zur modellbasierten Fehlerdetektion auf der untersten Ebene der Steuerungsarchitektur. Dazu werden in der Hardwareüberwachung die verfügbaren Hardwareressourcen (Aktorik und Sensorik) und in der Stabilitätsüberwachung die unterlagerten Regelkreise auf Einhaltung des vorgegebenen Sollverhaltens überwacht. Auf der mittleren Ebene der Steuerungsarchitektur erfolgt anschließend die systematische Behandlung der erkannten Ausnahmesituationen durch die Auswertungs- und Verwaltungskomponente des Überwachungskonzepts. Vorteile der vorgeschlagenen Grundstruktur bestehen dabei darin, die mittlere und die untere Hierarchieebene mit einem Höchstmaß an Autonomie zu versehen. Die entsprechenden Komponenten werden in die Lage versetzt, möglichst schnell ohne den direkten Eingriff der Aufgabenplanung auf auftretende Ereignisse (routinemäßig erwartete Situationen und berücksichtigte Ausnahmesituationen) zu reagieren. Dazu werden die Zustände aller Komponenten der unterlagerten Regelkreise (u. a. Regler, Aktoren und Sensoren) zur Laufzeit überwacht und entsprechend der geforderten Aufgabe bewertet. Auf Basis dieser Informationen entscheiden die übergeordneten Ebenen selbstständig über die Fortsetzung, die zeitweilige Unterbrechung oder den Abbruch der geplanten Handlung. Dies ist ein deutlicher Unterschied zu den klassischen Ansätzen der Aufgabenplanung [40, 190, 191, 235, 281], bei denen Änderungen des ursprünglichen Plans stets von der obersten Ebene der Steuerungsarchitektur überwacht und ausgeführt werden.

Die erfolgreiche Umsetzung des vorgestellten Konzepts setzt die Verfügbarkeit von mathematischen Modellen der Roboter-Teilkomponenten auf der Steuerungsarchitektur (aktive Modelle, Bild 2.1) vor-

aus. Auf Basis dieser Modelle erfolgt beispielsweise die Entwicklung von modellbasierten Regelungsalgorithmen und Funktionen zur Referenztrajektoriengenerierung. Weiterhin können die mathematischen Modelle zur Bestimmung von Ereignissen dienen, die zur Aktivierung von Transitionen in Petri-Netzen führen. Im Rahmen des vorgeschlagenen neuen Überwachungskonzepts werden die Modelle zudem zur modellbasierten Fehlerdetektion verwendet (Abschnitt 2.3.1). Weitere Argumente für den Einsatz von mathematischen Modellen sind die Möglichkeit zur umfassenden simulativen Erprobung des Konzepts, ohne Gefährdung der realen Hardware, sowie die Möglichkeit die Simulationsmodelle als Basis für die Optimierung von Auslegungsvarianten (z. B. Sensorik) zu verwenden. Im Kapitel 4 erfolgt deshalb die Evaluierung des petrinetzbasierten Konzepts zur Generierung von Aufgabenwissen am Beispiel von ausgewählten Roboterkomponenten. Dies beinhaltet die theoretische und experimentelle Modellbildung, den modellbasierten Reglerentwurf sowie die Definition geeigneter Aktionsfolgen für einen fluidisch betriebenen Mehrfinger-Robotergreifer und die anthropomorphe Halseinheit eines humanoiden Roboters. Kapitel 5 zeigt die Umsetzung des flexiblen, petrinetzbasierten Überwachungskonzepts anhand von Simulationen und Experimenten.

3 Implementierung

3.1 Übersicht

Grundsätzliches Ziel der Implementierung ist die Bereitstellung von lauffähigen Algorithmen auf dem jeweiligen Zielsystem. Dabei sind zwei prinzipielle Vorgehensweisen zu unterscheiden:

1. die Entwicklung und Erprobung der Algorithmen erfolgt direkt auf dem späteren Zielsystem, damit ist keine weitere Portierung nötig, oder
2. die Entwicklung und Erprobung erfolgt mit speziellen Entwicklungswerkzeugen, die den Entwicklungsprozess vereinfachen. Allerdings ist dann eine Portierung von der Entwicklungsplattform auf das Zielsystem unumgänglich.

In dieser Arbeit erfolgt die Entwicklung und Erprobung von Algorithmen auf der Entwicklungsplattform MATLAB/SIMULINK® [188]. Zielplattform ist das C/C++ Framework Modular Controller Architecture (MCA) [244] zur Ansteuerung der Roboterhardware. Bild 3.1 zeigt die schematische Darstellung der Zusammenhänge zwischen Entwicklungs- und Zielplattform sowie Roboterhardware.

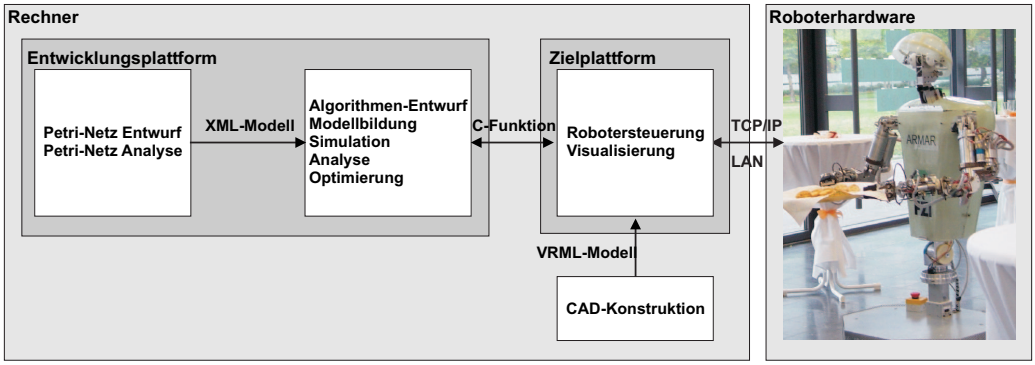


Bild 3.1: Schematische Darstellung der Zusammenhänge zwischen Entwicklungs- und Zielplattform sowie Roboterhardware

Im Weiteren wird zunächst die Vorgehensweise zum Entwurf, zur Simulation und zur Analyse von Petri-Netzen auf der Entwicklungsplattform vorgestellt (Abschnitt 3.2). Daran anschließend erfolgt die softwareseitige Umsetzung des in Kapitel 2 vorgestellten Konzepts zur Planung, Ausführung und Überwachung von Roboteraufgaben (Abschnitt 3.3). Den Abschluß des Kapitels bildet ein Portierungskonzept, das die entwickelten Algorithmen auf die Zielplattform transferiert (Abschnitt 3.4).

3.2 Entwurf, Simulation und Analyse von Petri-Netzen

Bei der Steuerung von Robotersystemen mit Petri-Netzen treten aufgabenspezifisch sehr unterschiedliche Petri-Netz-Topologien auf (vgl. Kapitel 2). Aus diesem Grund wird zunächst ein Konzept vorgeschlagen, mit dem einfach und systematisch Petri-Netze entworfen und anschließend auf der Entwicklungsplattform simuliert und analysiert werden können.

Zum Entwurf wird der frei verfügbare Petri-Netz Editor Netlab [212] verwendet. Vorteile von Netlab sind dabei die einfache und systematische Eingabe und Speicherung (XML-Format) von Petri-Netzen der Klasse Stellen/Transitions-Netze (S/T-Netze) per Drag & Drop. Die zugrunde liegende Theorie zu Petri-Netzen ist in [221] bzw. Anhang A zu finden.

Um die mit Netlab entworfenen Petri-Netze zur Ausführung und Überwachung von Roboteraufgaben verwenden zu können, müssen diese Netze auf der Entwicklungsplattform ausführbar sein. Zur Evaluierung und Weiterentwicklung ist zudem eine animierte Visualisierung der simulierten Netze sinnvoll. Die Umsetzung der in XML-Repräsentation gespeicherten Netlab-Netze erfolgt mit einer frei erhältlichen ('Open Source') MATLAB®-XMLParser-Toolbox (**XMLTree**) [89]. Zur Ausführung der entworfenen Petri-Netze in SIMULINK® wurde eine neue SIMULINK®-Funktion (*PetriEXE*) implementiert. Diese Funktion erhält als Parametervektor lediglich die vom XMLParser gelieferte Netz-Konfiguration des auszuführenden Petri-Netzes. Der Parametervektor enthält im Einzelnen:

- den Netznamen ($*$.xml),
- die Anzahl an Plätzen, Transitionen und Verbindungen im Netz,
- die X/Y-Koordinaten der Plätze, Transitionen und Verbindungen im Netz zur Visualisierung,
- die initiale Markenbelegung sowie
- die Markenkapazitäten und Kantengewichte.

Eingangsgrößen der SIMULINK®-Funktion **PetriEXE** sind die zeitlichen Verläufe der geschalteten Transitionen $\boldsymbol{T}(t)$. Ausgangsgrößen der SIMULINK®-Funktion sind die zeitlichen Verläufe der resultierenden Platzbelegung $\boldsymbol{A}(t)$ des simulierten Netzes. Innerhalb der SIMULINK®-Funktion sind Regeln bei Aktivierung einer Transition (Feuerregel) hinterlegt.
Für die hier implementierte Feuerregel gilt:

WENN Transition aktiv UND alle Vorgängerplätze belegt UND alle Nachfolgeplätze frei
DANN Feuern

Zur grafischen Darstellung des simulierten Petri-Netzes werden die Informationen über die Positionen der Plätze, Transitionen und Verbindungen im Netz in X/Y-Koordinaten verwendet (SIMULINK®-Funktion **PetriVISU**). Weiterhin verfügt die SIMULINK®-Funktion **PetriVISU** über die Möglichkeit, den Markenfluss zur Laufzeit zu animieren. Dazu werden ihr lediglich die Änderungen der Platzbelegung (durch Aktivieren einer Transition) als zusätzliche Parameter mitgeteilt. Ein Vorteil dieser Implementierung ist die Möglichkeit, innerhalb eines SIMULINK®-Modells mehrere Instanzen der Funktion **PetriEXE** für verschiedene Petri-Netze (z. B. mehrere Koordinierungs- und Verwaltungsnetze, vgl. Kapitel 2) zeitgleich auszuführen und zu visualisieren.

Alle in diesem Abschnitt vorgestellten selbst entwickelten Funktionen werden mit so genannten MATLAB®-S-Funktionen umgesetzt [188]. Mit S-Funktionen können kontinuierliche, diskrete und hybride (kontinuierlich und diskrete) Systeme abgebildet werden. Sie benutzen eine spezielle Aufruf-Syntax, was die Interaktion mit dem internen MATLAB®-Solver ermöglicht. Die Integration in

SIMULINK® erfolgt über standardisierte Blöcke in der SIMULINK®-Blockbibliothek. Eine Besonderheit der MATLAB®-S-Funktionen ist, Programmcode entweder in MATLAB® oder C zu schreiben (weitere Details siehe Abschnitt 3.4).

Die Validierung des vorgestellten Entwurfskonzepts erfolgt durch Umsetzung mit einer Testumgebung. Diese beinhaltet neben den bereits beschriebenen Funktionen zum Einlesen (Parsen) von XML-Dateien und zur Ausführung und Visualisierung von beliebigen Petri-Netzstrukturen ein, in Abhängigkeit von der Netz-Topologie, automatisch generiertes Bediener-Interface (GUI), mit dem die Transitionen des simulierten Petri-Netzes einzeln aktiviert werden können (derzeit aktivierte Transitionen Bild 3.2 oben und rechts rot). Bild 3.2 zeigt die Datenflüsse innerhalb der Testumgebung anhand eines Beispiel-Petri-Netzes mit Verzweigungen, Zusammenführungen und Synchronisationen.

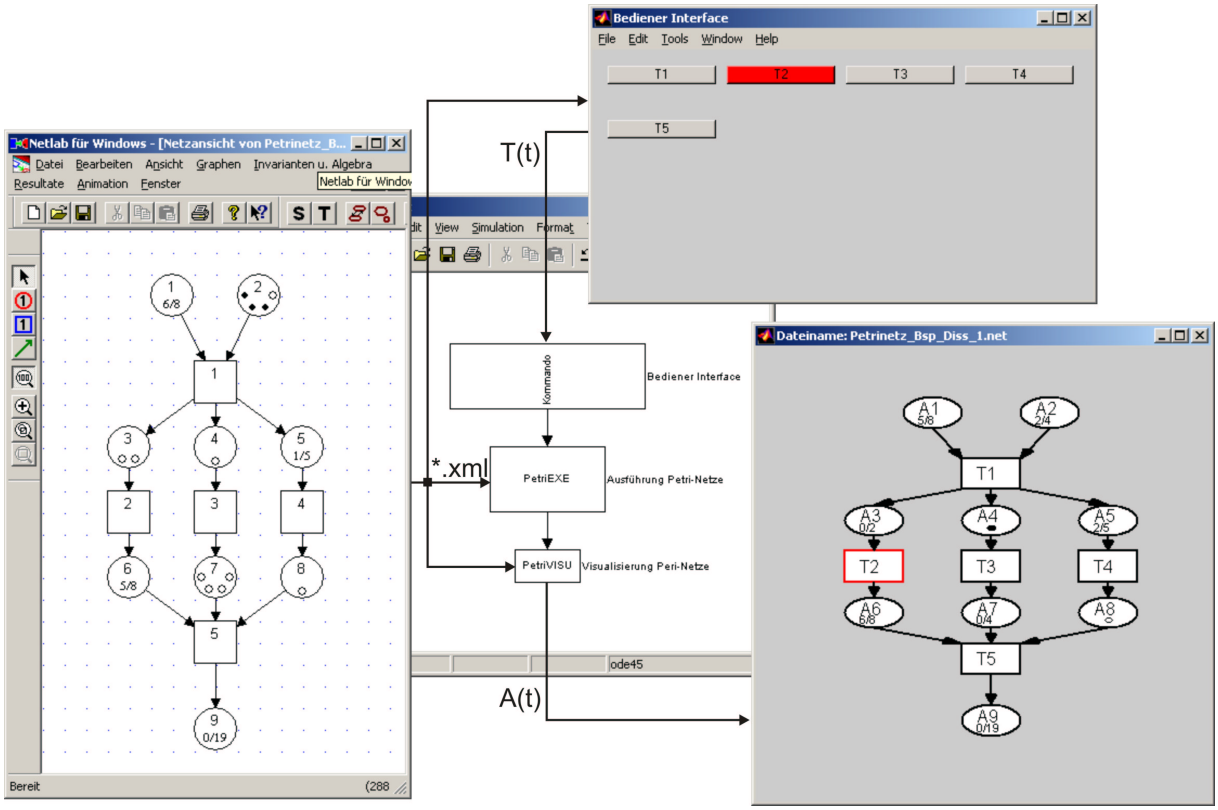


Bild 3.2: Testumgebung zum Entwurf, zur Simulation und zur Analyse von Petri-Netzen. Das Beispiel zeigt den Entwurf mit dem grafischen Editor Netlab (links), die ausführbare Umsetzung des Netzes (Mitte) sowie das automatisch generierte Bediener-Interface (oben) und die Visualisierung (rechts)

3.3 Umsetzung der Roboter-Steuerungsarchitektur

Für die erfolgreiche Implementierung des in Kapitel 2 vorgestellten Konzepts zur Planung, Ausführung und Überwachung von Roboteraufgaben sind mehrere Fragestellungen zu lösen. Zunächst ist eine

allgemeine Grundstruktur zur Umsetzung einer 3-Ebenen-Steuerungsarchitektur mit zusätzlichen Komponenten zur Überwachung, Wissensspeicherung und Ressourcenverwaltung auf der Entwicklungsplattform mit Petri-Netzen zu entwerfen. Als problematisch erweist sich dabei die zeitgleiche Umsetzung von verschiedenen Verwaltungs- und Koordinierungsnetzen (vgl. Kapitel 2) bei der Ausführung und Verwaltung komplexer Roboterhandlungen. Gemäß Bild 3.3 setzt sich die in dieser Arbeit entwickelte Grundstruktur aus zwei Komponenten zusammen, einem Bediener-Interface (Bild 3.3 oben rechts) und einem Simulationsmodell der Roboter-Steuerungsarchitektur (Bild 3.3 unten Mitte).

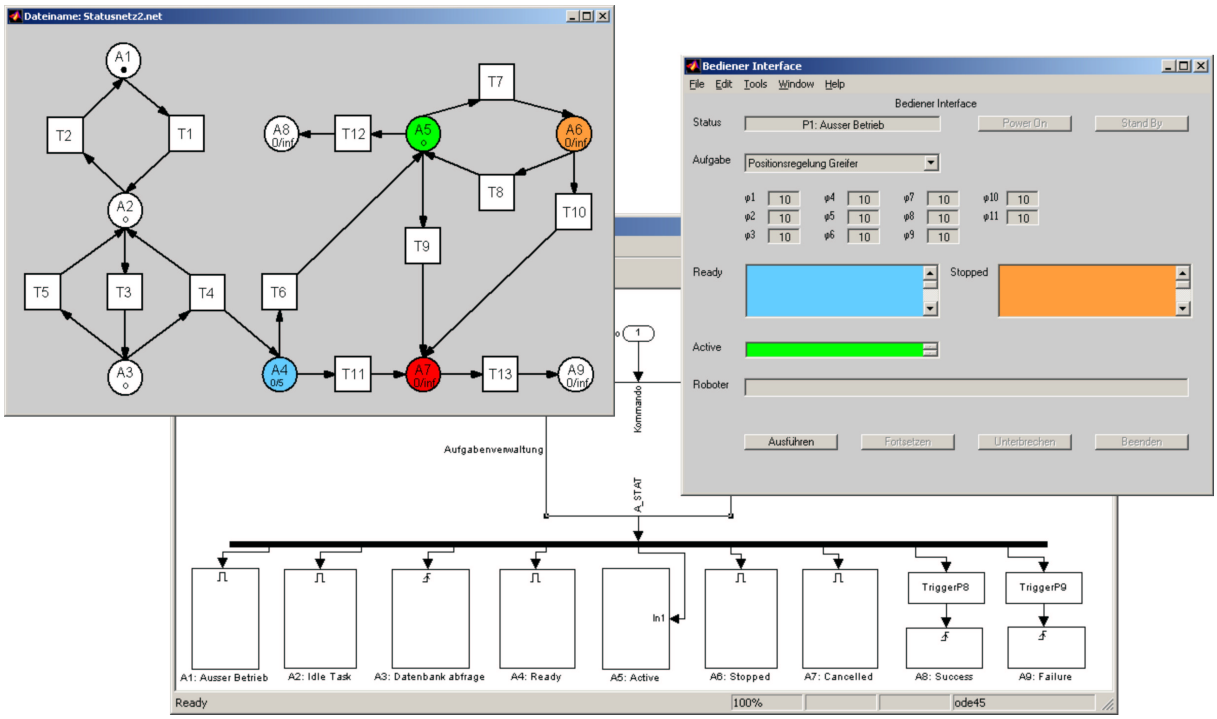


Bild 3.3: Umsetzung der Roboter-Steuerungsarchitektur nach Kapitel 2

Die Schnittstelle zwischen Benutzer und Simulationsmodell (Robotersystem) stellt ein Bediener-Interface (GUI) dar, mit dem zur Laufzeit der Simulation Benutzer-Kommandos an das Robotersystem übermittelt werden können. Solche Kommandos sind z. B. neue Aufgaben oder Abbruchs-, Unterbrechungs- oder Fortsetzungsbefehle für bereits existierende Aufgaben (vgl. Kapitel 2). Das Simulationsmodell der Roboter-Steuerungsarchitektur muss in der Lage sein, den Ablauf bei der Umsetzung von komplexen Handlungen zu steuern. Dazu enthält es nach Kapitel 2 Komponenten zur sensomotorischen Steuerung, zur Wissenspeicherung (Datenbank), zur Ressourcenverwaltung und zur Ausnahmesituationsbehandlung.

Die Umsetzung dieser übergeordneten Ablaufsteuerung erfolgt wiederum mit einem Petri-Netz, im Weiteren Statusnetz genannt (Bild 3.3 oben links und Bild 3.4), das durch Erweiterung des in Abschnitt 2.3 vorgestellten Verwaltungsnetzes entsteht. Jeder ereignisdiskrete Zustand des Statusnetzes entspricht einer Komponente der Steuerungsarchitektur auf der Modellebene. Wichtig ist dabei, dass nur diejenigen Subsysteme des Modells aktiv sind, deren Plätze im Statusnetz mit Marken belegt sind. Die Umsetzung dieser Funktionalität erfolgt mit Hilfe von standardisierten SIMULINK®-Bibliothek-Funktionen (**Enable**- und **Trigger**-Funktionen).

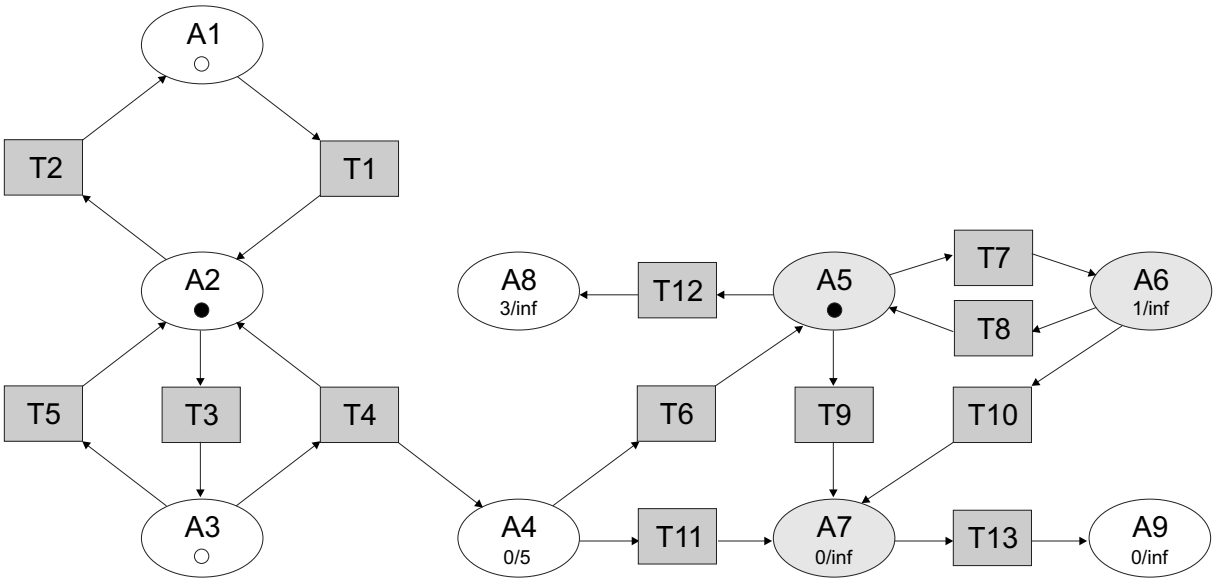


Bild 3.4: Statusnetz

Die Plätze A1 und A2 repräsentieren den allgemeinen Status des Robotersystems. Im Zustand A1 befindet sich das Robotersystem im 'Stand-By' Modus. Durch Aktivierung der 'Power-On' Transition T1 wird das Robotersystem eingeschaltet (Markierung von A2) und befindet sich danach im Zustand 'Bereit'. Aus diesem Zustand kann das Robotersystem wieder in den 'Stand-By' Modus zurückkehren (Aktivierung von T2). Hat der Benutzer allerdings eine Aufgabe kommandiert, wird die Transition T3 aktiviert und es erfolgt eine Datenbankabfrage auf dem Platz A3. Ist für diese Aufgabe Aufgabenwissen in Form von Elementaraktionen gespeichert, dann setzt die Aufgabenplanung auf der oberen Ebene der Steuerungsarchitektur dieses Wissen in eine petrinetzbasierte Beschreibung als Aktionsfolge AF in Form eines Koordinierungsnetzes um, und aktiviert die Transition T4. Dadurch wird die Aufgabe in den Aufgabenspeicher auf der mittleren Ebene der Steuerungsarchitektur weitergeleitet (Markierung von A4). Gleichzeitig nimmt das Robotersystem wieder den Zustand 'Bereit' (Markierung von A2) ein (Realisierung von T4 als Verzweigungstransition), um auch während der Verarbeitung einer Aufgabe weitere Benutzerkommandos entgegen nehmen zu können. Ist die Aufgabe nicht ausführbar oder dem Robotersystem nicht bekannt (keine entsprechenden Elementaraktionen in der Datenbank hinterlegt), wird die Transition T5 aktiviert. Danach befindet sich das Robotersystem wiederum im Zustand 'Bereit' und wartet auf ein neues Kommando.

Jede im Aufgabenspeicher abgelegte Marke (Markierung von A4) entspricht jetzt einer separaten Aktionsfolge AF. Durch die vom Entwickler festzulegende Markenkapazität des Platzes A4 lässt sich dabei die Größe des Aufgabenspeichers, in Abhängigkeit von der vorhandenen Rechenleistung des Robotersystems, reglementieren. Warten eine oder mehrere Aufgaben im Aufgabenspeicher, dann prüft die Ressourcenverwaltung, ebenfalls auf der mittleren Ebene der Steuerungsarchitektur, ob alle für die erfolgreiche Ausführung der Aktionsfolge notwendigen Hardwareressourcen verfügbar sind. Ist dies der Fall, wird die Ausführung dieser Aktionsfolge freigegeben und der Platz 'Aktiv' (Markierung von A5) des Statusnetzes mit dieser Marke belegt. Derzeit ist eine Markenkapazität von $\mathcal{K} = 1$ des Platzes 'Aktiv' implementiert. Damit ist allerdings nur die aktive Ausführung *einer* Aktionsfolge möglich. Eine höhere Markenkapazität des 'Aktiv' Platzes und damit die zeitgleiche Ausführung mehrerer verschiedener Aktionsfolgen sind mit dem vorgestellten Konzept prinzipiell möglich. Dies erfordert allerdings eine er-

weiterte Funktionalität der Ressourcenverwaltung, die sich dann zusätzlich um die Prioritätsverwaltung der im Aufgabenspeicher abgelegten Aktionsfolge kümmern muss. In Abhängigkeit von der derzeit aktiven Aktionsfolge wird dann das entsprechend parametrierte Koordinierungsnetz mit den zugehörenden Funktionen zur Trajektoriengenerierung und Regelung ausgeführt (unterste Ebene der Steuerungsarchitektur). Sind nicht alle benötigten Hardwareressourcen bereit oder verfügbar, dann wird die Ausführung dieser Aktionsfolge blockiert. Hat das Robotersystem die routinemäßige Ausführung einer Aktionsfolge erfolgreich (Regelziel erreicht) oder erfolglos (Regelziel innerhalb eines vorgegebenen Zeitintervalls nicht erreicht) beendet, dann werden vom Robotersystem selbstständig die entsprechenden Transitionen T12 und T13 aktiviert und die zugehörenden Marken in den Plätzen 'Success' (Markierung von A8) bzw. 'Failure' (Markierung von A9) gespeichert.

Treten bei der Abarbeitung einer Aktionsfolge Probleme auf, so kann der Benutzer oder das Robotersystem mit der in Abschnitt 2.3 vorgestellten petrinetzbasierten Systematik zur Ausnahmesituationsbehandlung aktiv in das Geschehen eingreifen. Hierzu existieren im Statusnetz, genau so wie im Verwaltungsnetz aus Abschnitt 2.3, die Plätze 'Stopped' zum zeitweiligen Unterbrechen der Aktionsfolge und 'Cancelled' zum irreversiblen Abbruch der Ausführung der Aktionsfolge. Die Umsetzung dieser Funktionalität erfolgt für jedes Roboter-Teilsystem mit Hilfe eines eigenen Verwaltungsmoduls. Dort findet sowohl die routinemäßige Verarbeitung der Aktionsfolge (Entscheidung darüber ob Aufgabenziel erreicht 'Success' oder nicht 'Failure') als auch die Ausnahmesituationsbehandlung (Abarbeitung der Aufgabe stoppen, fortsetzen oder abbrechen) statt. Tabelle 3.1 zeigt die Bedeutungen der Plätze und Transitionen des Statusnetzes im Detail.

Platz	Bedeutung
A1	Stand-By
A2	Idle Task (Status Bereit)
A3	Datenbankabfrage
A4	ausführungsbereite Aktionsfolge (Ready)
A5	routinemäßige Abarbeitung der Aktionsfolge (Active)
A6	zeitweilige unterbrochene Aktionsfolge (Stopped)
A7	irreversible abgebrochene Aktionsfolge (Cancelled)
A8	erfolgreich durchgeführte Aktionsfolge (Success)
A9	nicht erfolgreich durchgeführte Aktionsfolge (Failure)
Transition	**Bedeutung**
T1	Power On aktiviert
T2	Stand-By aktiviert
T3	Aufgabe kommandiert
T4	Aufgabe ist bekannt und ausführbar
T5	Aufgabe ist nicht bekannt und/oder nicht ausführbar
T6	Abarbeitung der Aktionsfolge gestartet
T7	Abarbeitung der Aktionsfolge unterbrochen
T8	Abarbeitung der Aktionsfolge routinemäßig fortgesetzt
T9-T11	Abarbeitung der Aktionsfolge irreversibel abgebrochen
T12	Abarbeitung der Aktionsfolge erfolgreich durchgeführt
T13	Abarbeitung der Aktionsfolge nicht erfolgreich durchgeführt

Tabelle 3.1: Bedeutung der Plätze und Transitionen des Statusnetzes

3.4 Portierungskonzept

Nach der Einteilung in Abschnitt 3.1 ist ein Portierungskonzept notwendig, das die entwickelten Algorithmen von der Entwicklungsplattform MATLAB/SIMULINK® [188] auf die Zielplattform Modular Controller Architecture (MCA) [244] transferiert.

Die Zielplattform ist das modular aufgebaute, transparente und echtzeitfähige, frei verfügbare C/C++ Framework MCA [244] zur Steuerung von Robotern und anderer Hardwarekomponenten. Die Hauptentwicklungsplattform ist Linux/RTLinux. Weiterhin existieren Versionen für Windows und Solaris. Alle Methoden in MCA sind als einfache Module mit standardisierten Schnittstellen (Controller-Input, Controller-Output, Sensor-Input, Sensor-Output, Parameter-Schnittstelle) realisiert. Diese Module sind über datentransportierende Kanten miteinander verbunden und können zu Modulgruppen zusammengefasst werden. Das Hauptprogramm (Part), das für die Ausführung und Verwaltung von Modulen und Modulgruppen zuständig ist, ermöglicht den Zugriff auf den Programmaufbau und die allgemeinen Modulschnittstellen und stellt diesen mittels TCP/IP-Schnittstelle zur Verfügung. Bild 3.5 links zeigt ein einzelnes MCA-Modul mit den Funktionen **Controle** (Verarbeitung von Steuerungsdaten) und **Sense** (Verarbeitung von Sensordaten). Bild 3.5 rechts zeigt beispielhaft den Aufbau einer Modulgruppe.

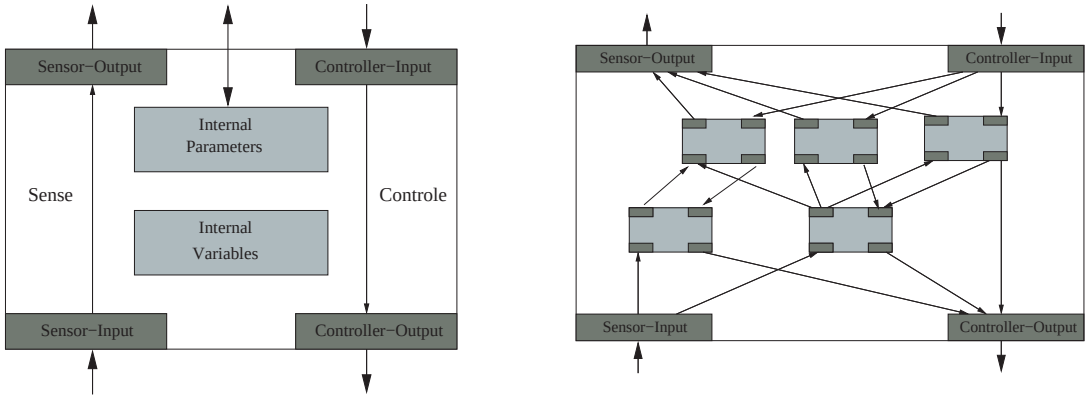


Bild 3.5: Aufbau eines MCA-Moduls (links), Beispiel für eine MCA-Modulgruppe (rechts) [244]

Weiterhin existieren als zusätzliche Applikationen die grafische Benutzeroberfläche MCAGUI und das grafische Steuerungsprogramm MCAadmin. Über MCAGUI können die Eingänge Controller-Input und Sensor-Output eines Moduls angesteuert bzw. visualisiert werden. Die Ansteuerung bzw. Visualisierung von Ein- und Ausgängen des Moduls erfolgt bei Controller-Input z. B. durch Schieberegler und bei Sensor-Output z. B. durch LCD's, LED's und 3D-Visualisierungen. In Bild 3.6 links ist die 3D-Visualisierung (Import als VRML-Modell aus dem vorhandenen CAD-Modell) eines Mehrfinger-Robotergreifers in MCAGUI zu sehen. Mit MCAadmin werden einzelne Module oder komplette Steuerungsarchitekturen grafisch als Baumstruktur angezeigt (Bild 3.6 rechts). Über MCAadmin können Eingänge *und* Parameter von Modulen und Modulgruppen zur Laufzeit modifiziert werden.

Da in dieser Arbeit eine Roboter-Steuerungsstruktur mit mehreren Hierarchieebenen zu Grunde gelegt wird, ist es notwendig, die für die jeweiligen Ebenen entwickelten Funktionen und Algorithmen (vgl. Abschnitte 3.2 und 3.3) auf das Zielsystem (MCA) zu transferieren. Auf der oberen und mittleren Ebene der Steuerungsstruktur sind dies hauptsächlich endliche Zustandsautomaten zur koordinierten Ausführung von geplanten Handlungen (H) und zur Behandlung von Ausnahmesituationen in Form von Petri-Netzen. Auf der unteren ausführenden Ebene sind dagegen Algorithmen zur Trajektoriengenerierung

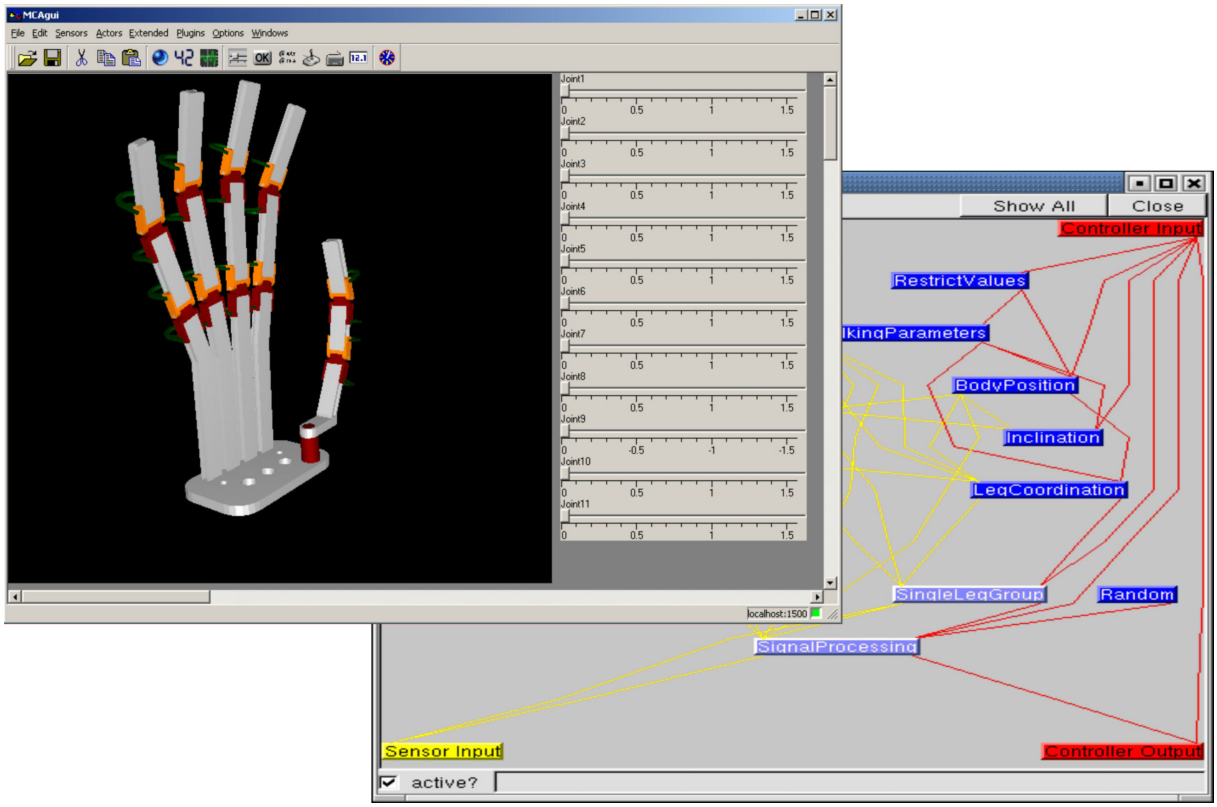


Bild 3.6: Visualisierung des Robotergreifers mit MCAGUI (links) und MCAadmin (rechts)

sowie zur modellbasierten Regelung und Fehlerdetektion implementiert. Aus diesen unterschiedlichen Anforderungen ergibt sich eine von der Hierarchieebene der Steuerungsstruktur abhängige Portierungsstrategie (Bild 3.7).

Um die entworfenen Petri-Netze auf der Zielplattform MCA zu verwenden, steht MCA-Petri zur Verfügung. Dieses in die MCA-Umgebung integrierte Tool ermöglicht das Einlesen und Ausführen von als XML-Dateien gespeicherten Petri-Netzen in MCA. Der Nachweis der Portierung von Petri-Netzen mit MCA-Petri geht allerdings über den Rahmen dieser Arbeit hinaus. Einzelheiten zur detaillierten Umsetzung sind [213] zu entnehmen.

Die Portierung der entwickelten Algorithmen zur Trajektoriengenerierung sowie zur modellbasierten Regelung und Fehlerdetektion von der Entwicklungs- auf die Zielplattform erfolgt mit einer speziellen Form von MATLAB®-S-Funktionen (vgl. Abschnitt 3.2). Diese so genannten CMEX-Wrapper-S-Funktionen erlauben das Einbinden von externen C/C++-Algorithmen in SIMULINK® ohne Änderungen am originalen Code (Code-Beispiel in Anhang). Eine Wrapper-Funktion ist dabei eine MATLAB®-S-Funktion, die in der Lage ist, externe Module bzw. Funktionen aufzurufen und damit den in diesen Modulen bzw. Funktionen hinterlegten Code in SIMULINK® auszuführen. Vorteil dieser Vorgehensweise ist, dass dieselben Funktionen, ohne weiteren Entwicklungsaufwand, sowohl auf der Entwicklungs- als auch auf der Zielplattform verwendet werden können. Bild 3.7 verdeutlicht die Identität von MCA-Modulen und MATLAB®-S-Funktionen. So verfügen beide über intern festgelegte Aufruf-Strukturen, die in Form von Funktions-Templates definiert sind. Es sind zur Umsetzung dieser Strategie zusätzlich zu den externen, die Funktionalität beschreibenden, C-Funktionen jeweils nur noch Ein- und Ausgänge, Übergabeparameter und interne Variablen zur vollständigen Definition festzulegen.

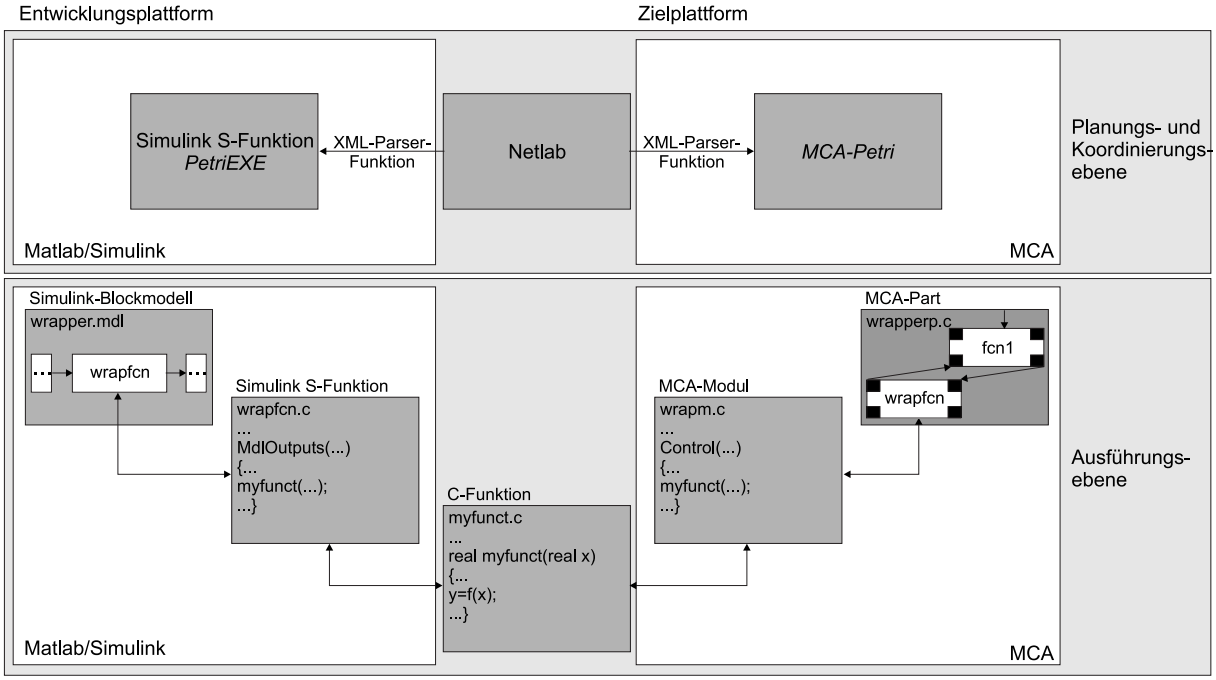


Bild 3.7: Schematische Darstellung der Portierungsstrategie in Abhängigkeit von der Hierarchieebene der Steuerungsstruktur

Um die Funktionalität des vorgestellten Portierungskonzepts zu zeigen, wird als Beispiel ein hybrider Kraft-Positionsregelungsalgorithmus zur Regelung von Greifvorgängen eines anthropomorphen Mehrfinger-Robotergreifers verwendet. Bei diesem Robotergreifer [253] werden die Freiheitsgrade im Gegensatz zu herkömmlichen Roboterhänden [63, 180] nicht durch Elektromotoren, sondern fluidisch durch flexible Aktoren angetrieben. Die ausführliche Darstellung der mathematischen Modelle der Teilkomponenten des flexiblen Robotergreifers (Pumpe, Ventile, Leitungen, Mechanik) und des hybriden Momenten-Positionsregelungsalgorithmus erfolgt in Kapitel 4.2. Im Weiteren dient der Algorithmus lediglich als Testfunktion zur Validierung des vorgeschlagenen neuen Portierungskonzepts. Die allgemeine Struktur des hybriden Kraft-Positionsregelungsalgorithmus ist in Bild 3.8 dargestellt.

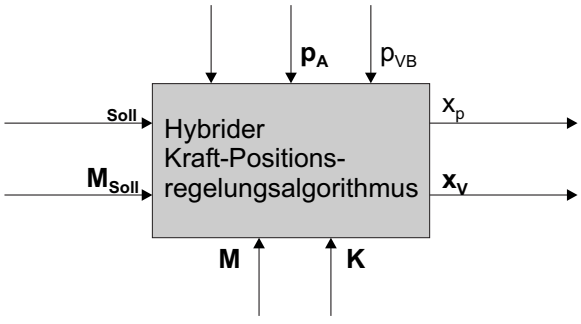


Bild 3.8: Eingangs- und Ausgangsgrößen des hybriden Kraft-Positionsregelungsalgorithmus nach [170]

Für die erfolgreiche Umsetzung eines Greifvorgangs ist die Interaktion zwischen Robotergreifer und Umgebung (zu greifendes Objekt) auf gewünschte Weise zu regeln. Anforderungen an den Regler sind dabei, Führungsgrößenvorgaben φ_{Soll}, $\mathbf{M}_{\mathrm{Soll}}$ in Abhängigkeit von der Kontaktsituation $\mathbf{K}$, in Gelenkwinkel φ

und Gelenkmomente $\mathbf{M}$ umzusetzen. Besonderheit des hier vorgestellten Algorithmus ist, dass für die Stellgrößenerzeugung systembedingt nur eine unabhängige Druckversorgung zur Verfügung steht. Stellgrößen des Kraft-Positionsregelungsalgorithmus sind deshalb die pulsweitenmodulierten Steuersignale für *eine* Pumpe x_P sowie die binären Steuersignale der Ventile $\mathbf{x_V}$.

Zunächst erfolgt der Entwurf des Regelungsalgorithmus als plattformunabhängige C-Funktion (**regler.C**). Zur Validierung der Funktionalität wird eine SIMULINK®-Testumgebung aufgebaut. Gemäß Bild 3.9 setzt sich die Testumgebung aus zwei Komponenten zusammen, dem Bediener-Interface (Bild 3.9 rechts) und dem Regelungsalgorithmus (Bild 3.9 unten). Die Umsetzung des Regelungsalgorithmus erfolgt nach dem oben beschriebenen Vorgehen mit einer CMEX-Wrapper-S-Funktion (**sfun_regler.C**, Bild 3.9 links). Eingänge der Wrapper-S-Funktion sind die vom Benutzer über das Bediener-Interface vorgegebenen Führungs- und Regelgrößen (vgl. Bild 3.8). Ausgänge sind die von der plattformunabhängigen C-Funktion (**regler.C**) berechneten Stellgrößen. Dazu wird die C-Funktion (**regler.C**) während der Laufzeit der Simulation periodisch von der CMEX-Wrapper-S-Funktion (**sfun_regler.C**) aufgerufen und ausgeführt. Die Ergebnisse der Simulation, d.h. die berechneten Stellgrößen, werden im Bediener-Interface angezeigt.

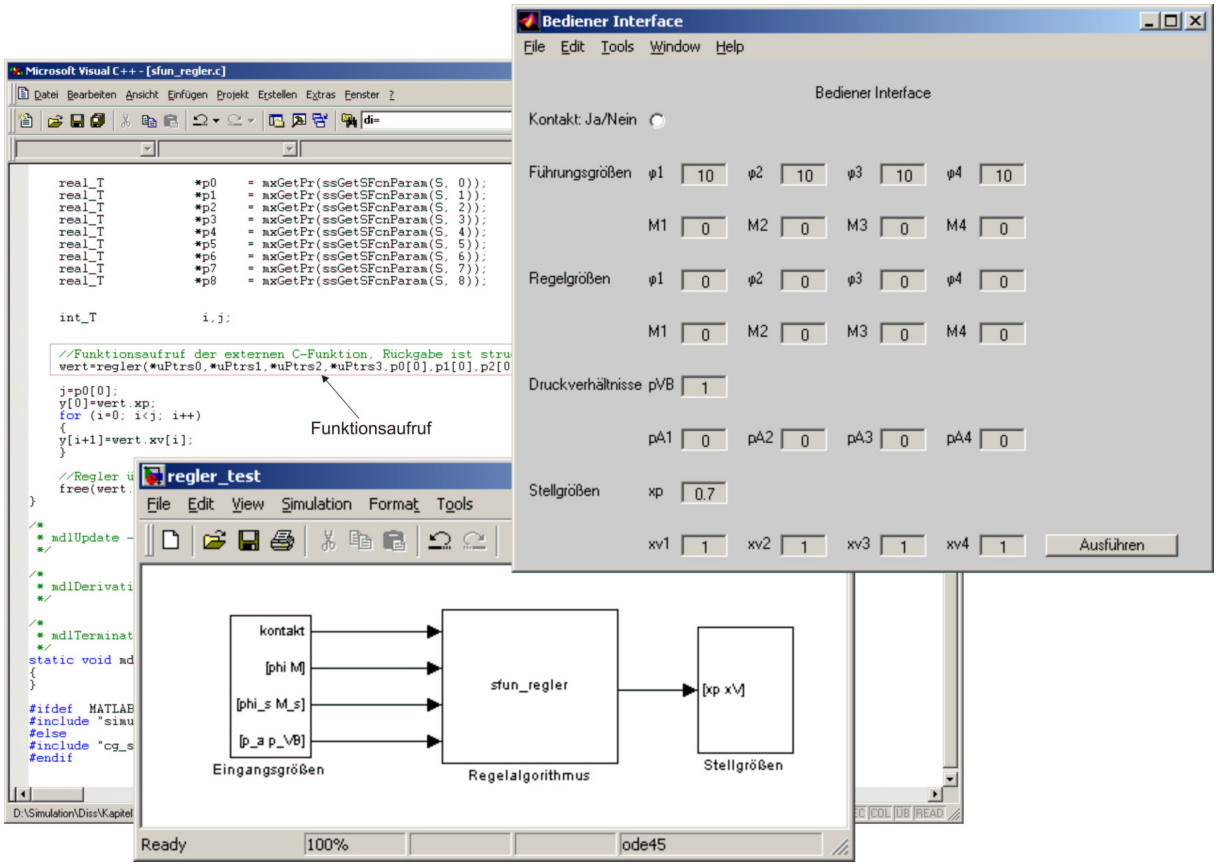


Bild 3.9: Umsetzung der Regler-Testumgebung auf der Entwicklungsplattform

Nach erfolgreicher Validierung des Regelungsalgorithmus wird die vollständig parametrierte C-Funktion (**regler.C**) auf die Zielplattform portiert. Um die erhaltenen Ergebnisse miteinander vergleichen zu können, wird für die Zielplattform eine identische Testumgebung entwickelt. Bild 3.10 zeigt diese

Testumgebung mit den Komponenten Bediener-Interface (MCAGUI, Bild 3.10 rechts) und der MCA-Umsetzung des Regelungsalgorithmus (MCAadmin, Bild 3.10 unten links). Als problematisch erweist sich dabei, dass in MCAGUI nur Controller-Input und Sensor-Output Werte veranschaulicht werden können. Das Modul **ControlerModule.C** besitzt jedoch keinen Sensor-Output. Aus diesem Grund wird das Modul **ControlerModule.C** mit zwei weiteren Modulen **ReplyValues.C** verknüpft und in eine Modulgruppe integriert (Bild 3.10 unten links). Diese Module sind MCA-Bibliothek-Komponenten, die die am Controller-Input anliegenden Werte, ohne zu bearbeiten, direkt an Sensor-Output weitergeben. Weiterführende Implementierungsdetails sind [211] zu entnehmen. Eingänge des MCA-Regler-Moduls **ControlerModule.C** sind wiederum die vom Benutzer über das Bediener-Interface vorgegeben Führungs- und Regelgrößen (vgl. Bild 3.8). Ausgänge sind die von der C-Funktion (**regler.C**) berechneten Stellgrößen. Dazu wird die C-Funktion (**regler.C**) während der Laufzeit der Simulation vom MCA-Regler-Modul **ControlerModule.C** aufgerufen und ausgeführt (Bild 3.10 oben links) und die Ergebnisse im Bediener-Interface angezeigt.

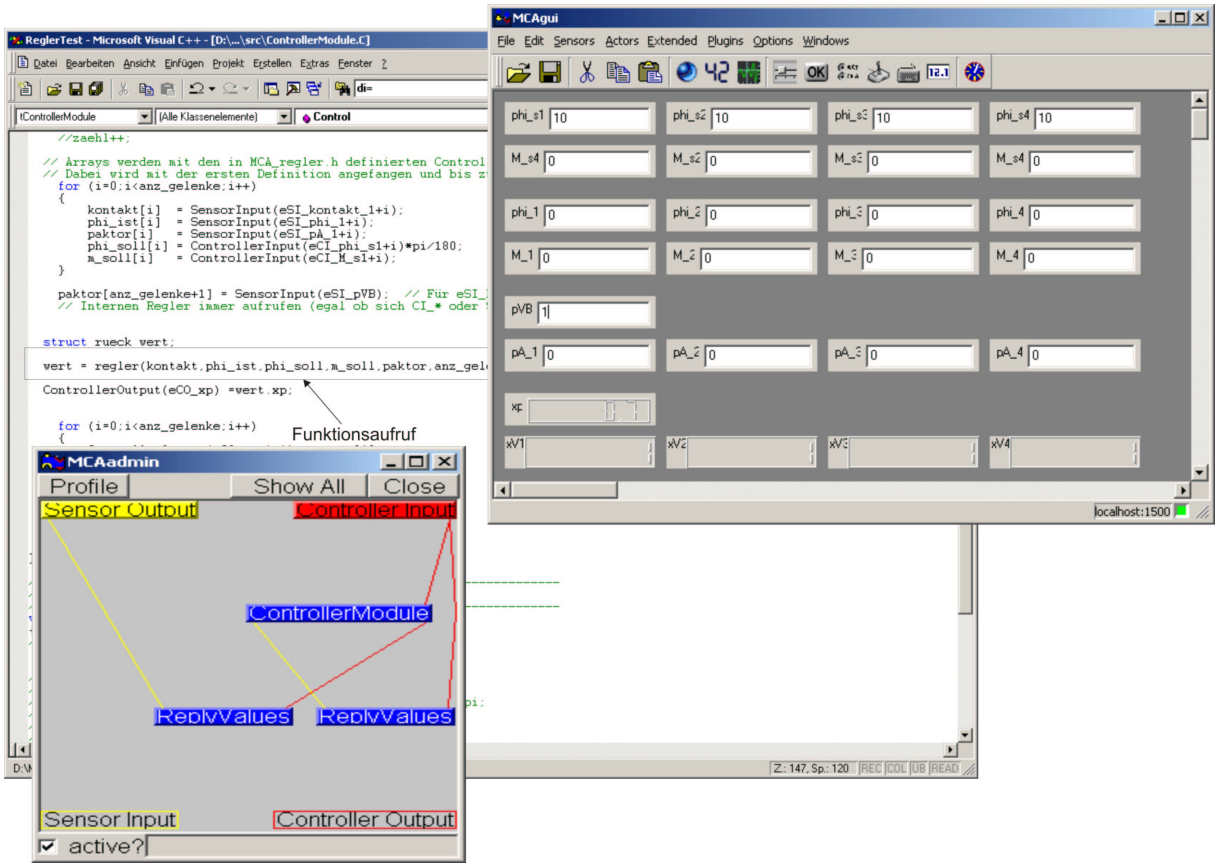


Bild 3.10: Umsetzung der Regler-Testumgebung auf der Zielplattform

Werden bei beiden Realisierungen der Regler-Testumgebung die gleichen Führungs- und Regelgrößen über die jeweiligen Bediener-Interfaces vorgegeben, dann ergeben sich erwartungsgemäß, da zur Berechnung bei beiden Versionen auf dieselbe Funktion (**regler.C**) zurückgegriffen wird, die gleichen Stellgrößenwerte in den jeweiligen Bediener-Interfaces (Bilder 3.9 und 3.10 jeweils rechts). Damit ist die prinzipielle Funktionalität des vorgeschlagenen neuen Portierungskonzepts nachgewiesen.

3.5 Diskussion

Grundsätzliches Ziel der Implementierung ist nach Abschnitt 3.1 die Bereitstellung von lauffähigen Algorithmen auf dem jeweiligen Zielsystem. Da für das in dieser Arbeit erstmals vorgestellte Verfahren zur Planung, koordinierten Ausführung und modellbasierten Überwachung eine Roboter-Steuerungsstruktur mit mehreren Hierarchieebenen zu Grunde gelegt wird, ist es notwendig, die entwickelten Funktionen und Algorithmen (vgl. Abschnitt 3.2 und Abschnitt 3.3) ebenenspezifisch auf das Zielsystem (MCA) zu transferieren. Auf der oberen und mittleren Ebene der Steuerungsstruktur sind dies hauptsächlich endliche Zustandsautomaten zur koordinierten Ausführung von geplanten Handlungen (H) und die Behandlung von Ausnahmesituationen. Auf der unteren ausführenden Ebene sind dagegen Algorithmen zur Trajektoriengenerierung sowie zur modellbasierten Regelung und Fehlerdetektion implementiert. Aus diesen unterschiedlichen Anforderungen ergibt sich eine von der Hierarchieebene der Steuerungsstruktur abhängige Portierungsstrategie (Bild 3.7).

In den Abschnitten 3.2 bis 3.4 wurden zur Umsetzung dieser Anforderungen erfolgreich Werkzeuge und Methoden vorgestellt, die den vollständigen, systematischen Entwurf (Netlab), die Analyse und Simulation (MATLAB/SIMULINK®) sowie die Portierung (MCA-Petri und SIMULINK CMEX-Wrapper-S-Funktionen) der entwickelten Algorithmen auf das Zielsystem (MCA) ermöglichen.

4 Neuartige Generierung von Aufgabenwissen für ausgewählte Roboterkomponenten

4.1 Übersicht

Das im Abschnitt 2.2 vorgestellte neue Konzept zur Planung und Ausführung von Roboteraufgaben mit Petri-Netzen wird im Folgenden zur Generierung von Aufgabenwissen für einen flexiblen Mehrfinger-Robotergreifer (Abschnitt 4.2) sowie einer anthropomorphen Roboter-Halseinheit (Abschnitt 4.3) angewendet. In dieser Arbeit umgesetzte Roboteraufgaben sind Bewegungsvorgänge wie z. B. Zeigegesten des Robotergreifers oder Kopfschütteln der Halseinheit, aber auch komplexe Handlungsabläufe wie Greifbewegungen oder bildbasierte Personen- und Objektverfolgungsvorgänge.

Da es sich bei diesen Roboterkomponenten um komplexe Teilsysteme handelt, kann bei der Generierung des Aufgabenwissens nicht vollständig auf standardisierte Bibliotheksfunktionen zur Trajektoriengenerierung und Regelung zurückgegriffen werden (vgl. Abschnitt 2.2.1). Stattdessen wird durch Ableiten der Systemgleichungen aus physikalischen Grundgesetzen (theoretische Modellbildung) und Vergleich mit Messungen am realen System (experimentelle Modellbildung) [130, 226, 267, 282] zunächst ein mathematisches Modell der entsprechenden Roboterkomponente aufgestellt, auf dessen Basis dann der Entwurf und die Evaluierung der Trajektoriengenerierung und der Regelung erfolgt. Das hierzu eingesetzte Identifikationsverfahren zur Ermittlung von mathematischen Modellen aus Messdaten ist die *Methode der Kleinsten Quadrate* (MKQ). Weiterhin können die mathematischen Modelle zur modellbasierten Aufgabenplanung und Fehlerdetektion sowie zur umfassenden simulativen Erprobung (auch in Grenzbereichen) eingesetzt werden. Zusätzlich ermöglicht der modellbasierte Ansatz die Analyse, den Vergleich und die Bewertung von verschiedenen u. U. noch nicht realisierten Auslegungsvarianten eines Roboter-Teilsystems (z. B. durch Variation der Antriebsart (Abschnitt 4.2.1.2) und/oder der Sensorausstattung (Abschnitt 4.2.1.4)) bzw. von vorhandenen Roboterkomponenten, die nicht immer zu Tests zur Verfügung stehen.

Die modellbasiert entworfenen Algorithmen werden dann zu hierarchisch strukturierten Petri-Netzen (Aktionsfolgen AF) zusammengefasst, im Aufgabenwissen des Robotersystems gespeichert und von der Steuerungsarchitektur petrinetzbasiert ausgeführt (Abschnitte 4.2.3 und 4.3.3).

4.2 Flexibler Mehrfinger-Robotergreifer

4.2.1 Modellbildung

In diesem Abschnitt erfolgt die mathematische Modellierung der einzelnen Komponenten des flexiblen Robotergreifers des Forschungszentrums Karlsruhe [248, 253] (Regelstrecke), bestehend aus dem mechanischen System (Kinematik und Dynamik) und den betriebsartabhängigen Teilsystemen des Antriebs

(Pumpe bzw. Kompressor, Ventile, Leitungen und Fluidaktoren). Die Modellierung baut auf den Arbeiten von [106, 146, 179, 183] auf, fasst aber erstmals die Teilmodelle zu einem integrierten Gesamtmodell zusammen, das die Flexibilität der Fluidaktoren berücksichtigt. Da gemäß Abschnitt 4.1 Greifvorgänge mit dem neuen Konzept umgesetzt werden sollen, müssen zur simulativen Evaluierung der entwickelten Algorithmen auch die zu greifenden Objekte (Geometrie, Materialeigenschaften) inklusive Kontaktsituation (Flächen- und Punktkontakt mit Reibung) modelliert werden. Weiterhin hat neben der Betriebsart auch die Sensorausstattung des Robotergreifers einen bedeutenden Einfluss auf die anwendbaren Regelungskonzepte und damit auf die Performance des Robotergreifers. Deshalb erfolgt zum Abschluss dieses Abschnitts eine Variantendiskussion.

4.2.1.1 Kinematik

Um eine gute Akzeptanz durch den Anwender zu erreichen, sind ein menschenähnliches äußeres Erscheinungsbild - Fünffinger mit opponierendem Daumen - und humanoide Bewegungsabläufe des Robotergreifers erstrebenswert. Diesen Anforderungen stehen technisch begründete Restriktionen entgegen. So sind Abduktionsbewegungen mit einem Fünffinger-Robotergreifer derzeit schwer zu realisieren. Bild 4.1 zeigt die schematische Darstellung der Kinematik des zu modellierenden Robotergreifers.

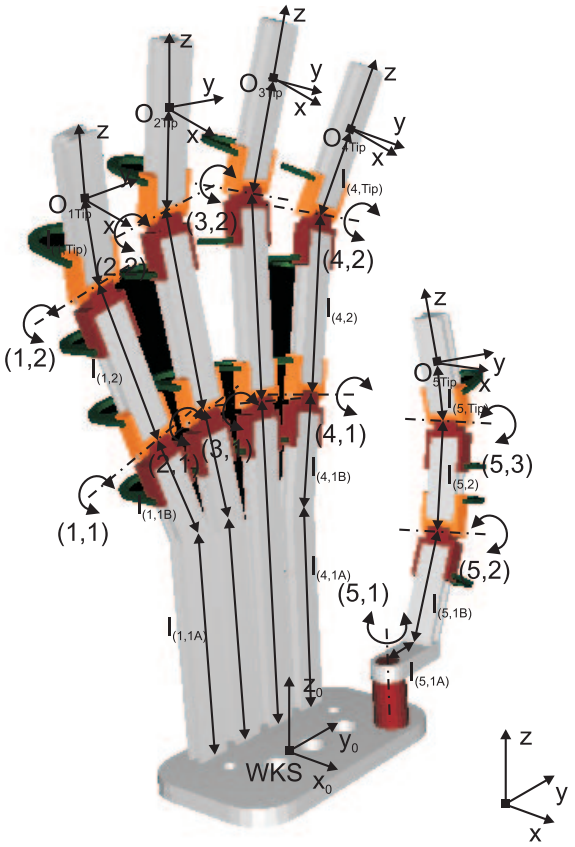


Bild 4.1: Kinematik des Robotergreifers mit wichtigen Koordinatensystemen

Tabelle 4.1 vergleicht Robotergreifer und menschliche Hand in Bezug auf die Anzahl unabhängiger Freiheitsgrade, Winkelbeweglichkeiten, relative Größe und Gewicht.

	menschliche Hand	Robotergreifer
Anzahl Freiheitsgrade [-]	19	$8-11$
Winkelbeweglichkeiten [°]	$0-90$	$10-75$
relative Größe [-]	1	1
Gewicht [g]	k.A.	800

Tabelle 4.1: Vergleich zwischen menschlicher Hand und Robotergreifer [202]

Da die Ansteuerung der Freiheitsgrade des Robotergreifers optional hydraulisch oder pneumatisch erfolgt, werden in nächsten Abschnitt die Teilsysteme des Antriebs für beide Betriebsarten vorgestellt und miteinander verglichen. Danach werden die ermittelten Funktionalbeziehungen zu einem gekoppelten Gesamtsystem (Abschnitt Dynamik) zusammengeführt.

4.2.1.2 Teilsysteme des Antriebs

Beiden Betriebsarten (Hydraulik und Pneumatik) gemeinsam ist der mechanische Aufbau (vgl. Abschnitt Kinematik). Unterschiede ergeben sich in der Fluidversorgung. Bild 4.2 zeigt schematisch die Teilsysteme des Antriebs für den hydraulischen und den pneumatischen Betrieb.

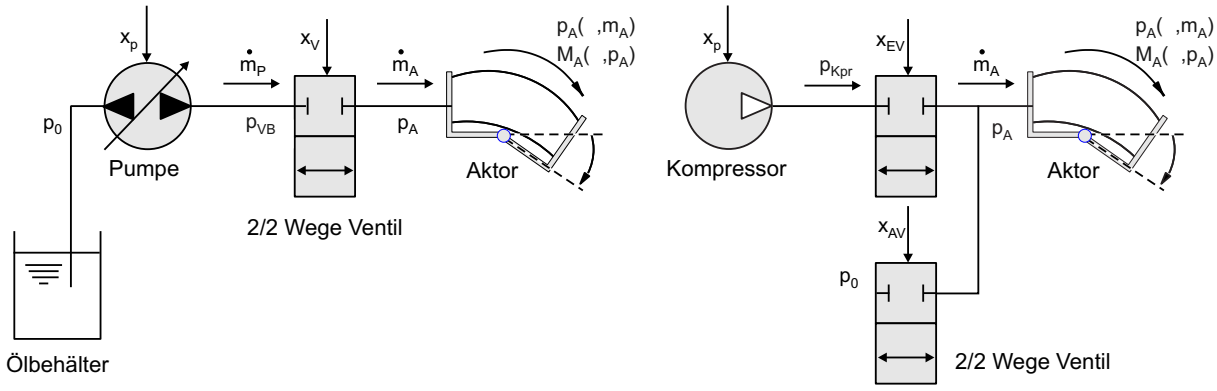


Bild 4.2: Teilsysteme des Antriebs für hydraulischen (links) und pneumatischen (rechts) Betrieb

Hydraulik Bei hydraulischem Betrieb sind die Pumpe und das 2/2 Wege Hydraulik-Ventil zu modellieren (Bild 4.2 links). Zur Beschreibung der Pumpe wird der Zusammenhang zwischen dem geförderten Massenstrom $\dot{m}_p$, dem Gegendruck p_{VB} und der angelegten, pulsweitenmodulierten Spannung x_p experimentell mit Hilfe eines Teststandes ermittelt [179]. Das sich ergebende Kennfeld $\dot{m}_p(x_p, p_{VB})$ beschreibt das statische Verhalten und beinhaltet sämtliche elektrischen, hydraulischen und mechanischen Verluste. Zur Ermittelung des dynamischen Verhaltens sind das Anlaufverhalten der Pumpe und deren Totzeit zu untersuchen. Das Ventil steuert durch Öffnen und Schließen den Massenstrom zum Aktor $\dot{m}_A$. Weiterhin wirkt es durch die Querschnittsveränderung K_V als Drossel und verursacht dadurch einen Druckabfall $\Delta p = p_A - p_{VB}$. Bei der Ansteuerung durch eine Pulsweitenmodulation (PWM) verändert sich K_V in Abhängigkeit von der Pulsweite x_V. Für das „statische“ Verhalten des Ventils ergibt sich ein Kennfeld der Form $\dot{m}_A(x_V, \Delta p)$ [179]. Da der Massenstrom $\dot{m}_A$ die Eingangsgröße in den Aktor ist, muss für das Modell zusätzlich der Zusammenhang zwischen p_A, φ und m_A bekannt sein

(resultierendes Kennfeld $p_A(m_A, \varphi)$). Hierfür wurde der Aktorteststand aus [34] um eine Volumenmessung erweitert [179]. Aus dem mit dem Teststand bestimmten Fluidvolumen kann dann, inkompressibles Fluidverhalten vorausgesetzt, direkt die gesuchte Fluidmasse im Aktor berechnet werden [183].

Pneumatik Gemäß Bild 4.2 rechts sind für den pneumatischen Betrieb die Modelle für den Kompressor und die 2/2 Wege Pneumatik-Ventile zu bestimmen. Der Kompressor wird als Druckspeicher mit konstantem Druck $p_{Kpr} > p_{A,max}$ angenommen [106]. Im pneumatischen Betrieb erhöht sich die Anzahl der Ventile pro Freiheitsgrad auf zwei, da der Kompressor nur einen Versorgungsdruck p_{Kpr} bereitstellt und nicht wie die Pumpe im hydraulischen Betrieb das Medium in zwei Richtungen durch die Ventile befördert. Dazu ist ein Ventil zum Einlassen der Luft mit dem Kompressor verschaltet. Das zweite Ventil zum Auslassen der Luft arbeitet nur gegen den Umgebungsdruck p_0 [183]. Vergleichbar dem hydraulischen Betrieb hängt der Massenstrom zum Aktor $\dot{m}_A$ dann von der Geometrie der verwendeten Ventile K_V, den entsprechenden Druckdifferenzen vor und hinter den Ventilen Δp sowie den an den Ventilen anliegenden binären Steuerspannungen x_{EV} bzw. x_{AV} ab. Da es sich bei Luft um ein kompressibles Medium handelt, kann aus der Fluidmasse im Aktor m_A über die allgemeine Gasgleichung der resultierende Aktordruck p_A berechnet werden. Im pneumatischen Betrieb kann nach [106] zudem *ein* geregeltes 3/2 Wege Pneumatik-Ventil zur unterlagerten Druckregelung eines Freiheitsgrads (DOF) verwendet werden. Wird alternativ ein solches 3/2 Wege Pneumatik-Ventil eingesetzt, so kann dessen Übertragungsverhalten p_A in Abhängigkeit vom Kompressordruck p_{Kpr} und der pulsweitenmodulierten Spannung x_V (resultierendes Kennfeld $p_A(p_{Kpr}, x_V)$) ausreichend genau durch ein Verzögerungsglied 1. Ordnung (PT_1-Glied) approximiert werden [36]. Aufgrund der Größe und des Gewichts des geregelten 3/2 Wege Pneumatik-Ventils wird dieser Ansatz hier allerdings nicht weiter untersucht.

Fluidaktor Als Aktor dient ein flexibler Fluidaktor. Bei einer Fluidzufuhr erhöht sich der Druck im Aktor, wodurch ein Drehmoment auf das Gelenk verursacht und daraus resultierend ein Gelenkwinkel eingestellt wird [248].

Um das reale Verhalten der Aktoren zu erfassen, wird das statische Kennfeld $M_A(\varphi, p_A)$ mit Hilfe des Aktorteststandes aus [34] aufgezeichnet. Über die schrittmotorgesteuerte Positionierung des Teststandes werden beliebige Gelenkwinkel absolut angefahren. Das Drehmoment des Aktors in Abhängigkeit vom Innendruck wird aus den Signalen eines Drehmomentsensors aufgezeichnet. Das mathematische Modell des Aktors wurde in [146] vorgestellt und wird hier kurz skizziert. Bild 4.3 links zeigt die drei wesentlichen Anteile, aus denen sich das Aktormoment zusammensetzt.

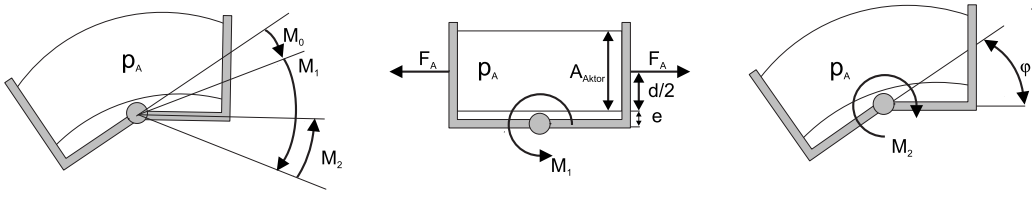


Bild 4.3: Einzelmomente am Aktor (links), vom Druck auf die Stirnflächen erzeugtes Drehmoment M_1 (Mitte), Rückstellmoment M_2 durch Versteifung des Aktormaterials unter Druck (rechts)

- vom Druck auf die Stirnflächen erzeugtes Drehmoment des Aktors M_1 (Bild 4.3 Mitte):

$$M_1(p_A) = F_A \left(\frac{d}{2} + e\right) = p_A \, A_{\text{Aktor}} \left(\frac{d}{2} + e\right). \tag{4.1}$$

- Momente M_0 und M_2, die von den Materialeigenschaften und der Geometrie des Aktors abhängen:

  – Drehmoment durch die elastische Dehnung des Aktormaterials bei $p_A = 0$. In [146] wird M_0 als Polynom 3.Grades in φ angesetzt. Simulationsergebnisse zeigen aber, dass es ausreicht, M_0 als linear anzunehmen, was zu einem deutlich geringeren Rechenaufwand ohne wesentliche Reduktion der Modellgüte führt:

$$M_0(\varphi) = a_{DM0} + a_{DM1}\,\varphi \qquad \text{mit} \quad a_{DM1} < 0. \tag{4.2}$$

  – Rückstellmoment durch Versteifung des Aktormaterials unter Druck (Bild 4.3 rechts). Der Aktor verformt sich unter Druck derart, dass eine zu p_A proportionale Versteifung des Materials eintritt. Dies bewirkt ein Rückstellmoment:

$$M_2(\varphi, p_A) = -c\,p_A\,\varphi. \tag{4.3}$$

Nach Zusammenfassung der Konstanten zu $a_{DM2} = -c$ und $a_{DM3} = A_{\text{Aktor}}\left(\frac{d}{2} + e\right)$ ergibt sich durch Addition der Momente die theoretische Momentenkennlinie des Aktors

$$M_A(\varphi, p_A) = a_{DM0} + (a_{DM1} + a_{DM2}\,p_A)\varphi + a_{DM3}\,p_A. \tag{4.4}$$

4.2.1.3 Dynamik

Um die Dynamik eines Mehrfinger-Robotergreifers zunächst ohne Interaktion mit der Umwelt zu modellieren, wird der Greifer als Verbund von n_F unabhängigen Fingern mit jeweils m_i Gelenken aufgefasst (Gelenkwinkel $\varphi_{(n_F,m_i)}$), die sich als offene Ketten durch den Arbeitsraum bewegen. Wenn dann Fingerglieder, Gelenke und Achsen als starr angenommen werden, ergeben sich folgende aus der Literatur [70, 220, 254] bekannte dynamische Bewegungsgleichungen für einen einzelnen Finger

$$\boldsymbol{\tau}_{\text{Antrieb}} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{f}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) \tag{4.5}$$

mit:

$\mathbf{q}$	$=$	$(\varphi_{(1,1)} \cdots \varphi_{(n_F,m_i)})^T$	$=$	Vektor der verallgemeinerten Gelenkwinkel
$\dot{\mathbf{q}}$	$=$	$(\dot{\varphi}_{(1,1)} \cdots \dot{\varphi}_{(n_F,m_i)})^T$	$=$	Vektor der Gelenkwinkelgeschwindigkeiten
$\ddot{\mathbf{q}}$	$=$	$(\ddot{\varphi}_{(1,1)} \cdots \ddot{\varphi}_{(n_F,m_i)})^T$	$=$	Vektor der Gelenkwinkelbeschleunigungen
$\boldsymbol{\tau}_{\text{Antrieb}}$	$=$	$(M_{A(1,1)} \ldots M_{A(n_F,m_i)})^T$	$=$	Vektor der verallg. Antriebsmomente.

Weiterhin bezeichnen $\mathbf{M}(\mathbf{q})$ die $(\sum_{i=1}^{n_F} m_i \times \sum_{i=1}^{n_F} m_i)$ Trägheitsmatrix, $\mathbf{f}(\dot{\mathbf{q}})$ den $(\sum_{i=1}^{n_F} m_i \times 1)$ Vektor der Reibungskräfte, $\mathbf{g}(\mathbf{q})$ den $(\sum_{i=1}^{n_F} m_i \times 1)$ Vektor der Gravitationskräfte und $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ den $(\sum_{i=1}^{n_F} m_i \times 1)$ Vektor der Coriolis- und Zentripetalkräfte.

Für Robotergreifer mit elastischen Gelenken und Antrieben muss das Dynamikmodell des starren Robotergreifers (4.5) erweitert werden. Die Bewegungsgleichungen solcher Roboter werden z. B. in [144, 224, 260] aufgestellt. Im Gegensatz zum Starrkörpermodell wird hierbei jedes Gelenk mit dem darauf folgenden Fingerglied als Zweimassensystem modelliert [16]. Zur Beschreibung der Bewegung werden dann zwei verallgemeinerte Koordinaten $\mathbf{q} = (\mathbf{q}_1, \mathbf{q}_2)^T$ benötigt, wobei $\mathbf{q}_1$ der Vektor der Antriebspositionen und $\mathbf{q}_2$ der Vektor der Fingergliedpositionen ist. $\boldsymbol{\tau}_{\text{G}}$ bezeichnet das Drehmoment der Gelenkfeder und $\boldsymbol{\tau}_{\text{Antrieb}}$ weiterhin das vom Antrieb erzeugte Moment

$$\begin{aligned} \boldsymbol{\tau}_{\text{Antrieb}} &= \mathbf{J}_{\text{Antrieb}}\,\ddot{\mathbf{q}}_\mathbf{1} + \boldsymbol{\tau}_{\text{G}} \\ \boldsymbol{\tau}_{\text{G}} &= \mathbf{M}(\mathbf{q}_\mathbf{2})\ddot{\mathbf{q}}_\mathbf{2} + \mathbf{c}(\mathbf{q}_\mathbf{2}, \dot{\mathbf{q}}_\mathbf{2})\dot{\mathbf{q}}_\mathbf{2} + \mathbf{f}(\dot{\mathbf{q}}_\mathbf{2}) + \mathbf{g}(\mathbf{q}_\mathbf{2}) \\ \boldsymbol{\tau}_{\text{G}} &= \mathbf{C}(\mathbf{q}_\mathbf{1} - \mathbf{q}_\mathbf{2}). \end{aligned} \tag{4.6}$$

Das Gleichungssystem (4.6) entspricht prinzipiell Gleichung (4.5), jedoch wird das Eingangsdrehmoment über die Gelenkfeder mit dem Vektor der Federkonstanten $\mathbf{C}$ übertragen. Der Motor ist über seine eigene Dynamik an das Gelenkmoment gekoppelt. Bei dem in dieser Arbeit verwendeten Robotergreifer des Forschungszentrums Karlsruhe wird das Eingangsdrehmoment von einem fluidischen System erzeugt (vgl. Abschnitt Teilsysteme des Antriebs). Dieses Eingangsdrehmoment $\boldsymbol{\tau}_{\text{Antrieb}}$ hängt zwar vom aktuellen Gelenkwinkel ab, wird aber im Unterschied zu Gleichung (4.6) direkt übertragen [36, 39]. Somit lässt sich die Dynamik des Robotergreifers, trotz der durch die flexiblen Fluidaktoren erzeugten Elastizität, nach Gleichung (4.5) mit $\boldsymbol{\tau}_{\text{Antrieb}}(\mathbf{q})$ modellieren.

Für den Mehraktorfall werden die vorgestellten Modelle des Antriebs (Hydraulik bei inkompressiblen Fluiden, Pneumatik bei kompressiblen Fluiden) [170, 184] und das mit Gleichung (4.5) beschriebene Dynamikmodell erstmals zu einem Gesamtmodell integriert. Für den Einsatz in der Robotik sind sowohl hydraulisch als auch pneumatisch angetriebene Robotergreifer mit jeweils $n_F = 5$ Fingern und $m_i = 3$ (Daumen) bzw. $m_i = 2$ (andere Finger) Gelenken pro Finger denkbar [214]. Das resultierende gekoppelte mathematische Modell lässt sich dann durch das folgende nichtlineare Differentialgleichungssystem beschreiben:

$$\ddot{\boldsymbol{\varphi}} = \mathbf{f}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}, \mathbf{M}_A(\boldsymbol{\varphi}, \mathbf{p}_A) - \mathbf{M}_z(\mathbf{z})) \tag{4.7}$$

$$\mathbf{p}_A = \mathbf{f}(\mathbf{m}_A, \boldsymbol{\varphi}) \tag{4.8}$$

$$\dot{\mathbf{m}}_\mathbf{A} = \begin{cases} \mathbf{f}(\Delta\mathbf{p}, \mathbf{x}_V, x_P) & \text{für hydraulischen Betrieb} \\ \mathbf{f}(\Delta\mathbf{p}, \mathbf{x}_{EV}, \mathbf{x}_{AV}) & \text{für pneumatischen Betrieb} \end{cases} \tag{4.9}$$

mit den Gelenkwinkeln[1] $\boldsymbol{\varphi}$ und den Gelenkwinkelgeschwindigkeiten $\dot{\boldsymbol{\varphi}}$ sowie den Fluidmassen in den Aktoren $\mathbf{m}_A$ als Zustände des Systems. Die Drücke in den Aktoren $\mathbf{p}_A$ werden aus diesen Zuständen berechnet. Mit den Stellgrößen[2] x_P (Steuerspannung der Pumpe) und $\mathbf{x}_V$ (Steuerspannungen der Ventile) lassen sich im hydraulischen Betrieb die jeweiligen Massenströme durch die Pumpe und durch die Ventile in die Aktoren beeinflussen. Im pneumatischen Betrieb werden als Stellgrößen der Regelung dagegen die Steuerspannungen der jeweiligen Einlass- und Auslassventile $\mathbf{x}_{EV}$ und $\mathbf{x}_{AV}$ verwendet.

Da der reale Robotergreifer des Forschungszentrums Karlsruhe [183, 184, 253] noch nicht mit der entsprechenden Sensorik (vgl. Abschnitt 4.2.1.4) ausgestattet ist, kann derzeit keine experimentelle Evaluierung des gekoppelten Gesamtmodells erfolgen. Allerdings basiert das hier erstmals präsentierte gekoppelte Gesamtmodell auf den in [106, 179, 183] ausführlich experimentell evaluierten Einzelmodellen. Weiterhin wurde das resultierende Gesamtmodell zahlreichen Tests unterzogen, um das quantitative Verhalten mit dem des realen Robotergreifers zu vergleichen. So sind beispielsweise die Zeiten, die Greifermodell und realer Robotergreifer bei identischer Ansteuerung für das Schließen bzw. Öffnen, bestimmte Gesten (z. B. Zeigegeste) oder spezielle Griffe (z. B. Pinzettengriff) benötigen, mögliche qualitative Maße zur Beurteilung der Modellgüte des Gesamtmodells. Daraus resultierend werden die Modellparameter so gewählt, dass ein möglichst reales Verhalten des Gesamtmodells erreicht werden kann.

Beim Vergleich von pneumatischer und hydraulischer Antriebsart im Mehraktorfall ergeben sich bei der Hydraulik Vorteile im Hinblick auf die Mobilität des Robotergreifers, da kein externer Kompressor zur Druckerzeugung notwendig ist. Nachteilig im hydraulischen Betrieb ist dagegen, dass zur Regelung *aller* unabhängigen Freiheitsgrade des Robotergreifers nur *eine* Druckversorgung (Pumpe) zur Verfügung steht. Dies führt zu Restriktionen in Bezug auf die Anwendbarkeit von Regelungskonzepten (Abschnitt 4.2.2) und damit in bestimmten Situationen (z. B. Ausführung gegenläufiger Bewegungen) zu Einschränkungen der Greiferperformance (Abschnitt 4.2.2).

[1] die fett gedruckten Variablen bezeichnen Vektoren der Dimension $(\sum_{i=1}^{n_F} m_i \times 1)$

[2] die normal gedruckten Variablen bezeichnen Skalare

4.2.1.4 Einfluss der Sensorausstattung

Neben der Betriebsart hat die Sensorausstattung des Robotergreifers einen bedeutenden Einfluss auf die Anwendbarkeit von Regelungskonzepten. Deshalb werden im Weiteren drei Varianten des Robotergreifers mit unterschiedlicher Sensorausstattung betrachtet. Variante I ist die dabei die Referenz-Variante mit maximaler Sensorausstattung. Bei Variante II handelt es sich um einen Robotergreifer mit reduzierter Sensorausstattung, d.h. fehlender taktiler Sensorik. Variante III hat keine integrierte Sensorik [248]. Diese Variante ermöglicht somit lediglich die *Steuerung* von Greifvorgängen. Weiterhin sind alle möglichen Kombinationen von Sensorausstattungen denkbar. Um die Variantenanzahl jedoch überschaubar zu halten, wird hier auf die Diskussion weiterer Möglichkeiten verzichtet. Tabelle 4.2 zeigt die drei hier untersuchten Varianten des Robotergreifers, die sich lediglich hinsichtlich der Sensorausstattung unterscheiden.

Sensorart Robotergreifer	taktile Sensorik	Winkelsensorik	Drucksensorik
Variante I	X	X	X
Variante II		X	X
Variante III			

Tabelle 4.2: Varianten des Robotergreifers in Abhängigkeit von der Sensorausstattung

Die Bestimmung der Gelenkpositionen eines Robotergreifers durch Winkelsensorik ist von großer Bedeutung, da eine zuverlässige rein bildbasierte Erfassung der Gelenkpositionen nur schwer realisierbar ist [273]. Damit ist die integrierte Winkelsensorik die Grundlage für die erfolgreiche Positionsregelung der Freiheitsgrade des Robotergreifers. Zusätzlich können aus den Gelenkpositionen, unter Verwendung der inversen Greiferkinematik (vgl. Abschnitt 1.2.2), kartesische Fingerpositionen berechnet werden, die beispielsweise zur übergeordneten Greifplanung [107, 214] oder zur Kollisionsvermeidung [159, 160] einzusetzen sind. Es existieren zahlreiche verschiedene Verfahren zur Winkelpositionsmessung. Einen guten Überblick gibt [183].

Die taktile Sensorik eines Robotergreifers dient dem Erkennen von Kontaktzuständen und den daraus resultierenden Kontaktkräften beim Durchführen von Greifvorgängen. Damit können durch Verwendung eines geeigneten Kraft- bzw. Momentenregelungsverfahrens (Abschnitt 4.2.2) die für einen erfolgreichen Greifvorgang (d. h. ausreichend große Greifkräfte zum sicheren Transportieren des gegriffenen Objekts, aber gleichzeitig nicht zu große Greifkräfte um Objektbeschädigung zu vermeiden) erforderlichen Kontaktkräfte eingeregelt werden. Auch bei der taktilen Sensorik existieren zahlreiche verschiedene Verfahren. Die Hauptkriterien zur Auswahl des Messprinzips sind nach [183] die zur Verfügung stehende Greifermechanik und die gewünschte Performance des Robotergreifers.

Drucksensoren unterscheiden sich von den beiden bereits vorgestellten Sensorarten, da sie nicht zwingend innerhalb der Finger (taktile Sensorik) bzw. der Greifergelenke (Winkelsensorik) angebracht werden müssen. Eine sinnvolle Platzierung ist beispielsweise im Bereich der Ventilbank. Sie unterliegen damit nicht so extremen räumlichen Restriktionen, was sich positiv auf die Kosten und die mögliche Sensorauflösung auswirkt. Drucksensoren lassen sich bei fluidisch betriebenen Robotergreifern beispielsweise zur modellbasierten Kontakterkennung [38] und Fehlerdetektion [174], zur unterlagerten Druckregelung [106] oder zum Überlastungsschutz der Antriebskomponenten einsetzen. Ein Überblick über die zahlreichen verschiedenen Wirkprinzipien von Drucksensoren ist in [183] zu finden.

Durch die fehlende taktile Sensorik ist es bei der Variante II nicht möglich, den Kontaktzustand und das aus dem Objektkontakt resultierende externe Störmoment auf den Robotergreifer zu messen. Dies

kann beispielsweise durch Verwendung einer modellbasierte Zustandserkennung, die in der Lage ist, den Kontaktzustand und die externen Störmomente modellbasiert zu schätzen [35], kompensiert werden. Damit sind die im nächsten Abschnitt vorgestellten Regelungskonzepte bei der Variante II weiterhin anwendbar. Demgegenüber ist bei der Variante III, ohne integrierte Sensorik, lediglich eine Steuerung der Freiheitsgrade möglich [228].

4.2.1.5 Objektmodellierung

Eine der wichtigsten Aufgaben für einen Mehrfinger-Robotergreifer ist das Greifen von unterschiedlichen Gegenständen. Um die entwickelten Algorithmen simulativ zu evaluieren, sind neben den Funktionen zur Trajektoriengenerierung und Regelung, dem Robotergreifer (Kinematik, Dynamik und Teilsysteme des Antriebs) auch die zu greifenden Objekte (Lage im Raum, Geometrie und mechanische Eigenschaften) zu modellieren. Da der Schwerpunkt dieses Kapitels bei der Umsetzung des neuen Konzept zur Generierung von petrinetzbasierten Aufgabenwissen liegt, werden hier ausschließlich zylindrische Objekte modelliert.

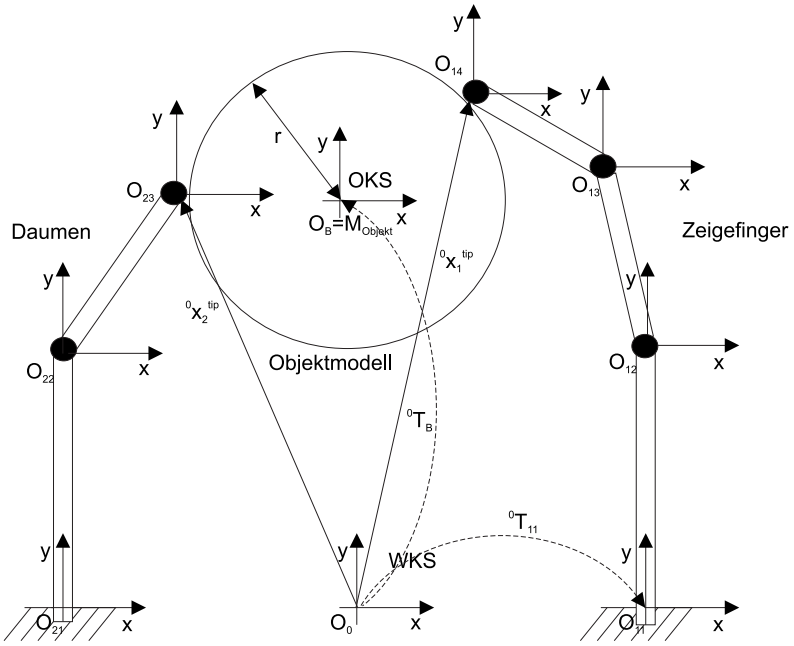


Bild 4.4: Koordinatensysteme im Kontaktfall

Die Objektposition und -ausrichtung in Bezug zum WKS (Weltkoordinatensystem, Ursprung O_0) werden durch das OKS (Objektkoordinatensystem, Ursprung O_B) bestimmt. Der Ursprung des OKS ist der Mittelpunkt M_{Objekt} des Objekts im WKS. Die Objektgröße wird durch den Radius r und die Höhe h festgelegt. Für die Kontaktmodellierung werden im Weiteren geometrische Betrachtungen erforderlich. Aus diesem Grund werden die Gelenkwinkel $\varphi_{(i,j)}$ unter Verwendung der direkten Kinematik der Finger bezüglich des Ursprungs der Fingerkoordinatensysteme $O_{(i,j)}$ und der homogenen Koordinatentransformation ${}^0T_{(i,j)}$ in kartesische Koordinaten der Fingerspitze ${}^0x_i^{tip}$ umgerechnet (vgl. Bild 4.4). Die Objekteigenschaften werden durch die Objektmasse m sowie die Federkonstanten k_O definiert (Feder-Masse-System). Damit ergibt sich die resultierende Kontaktkraft F_K radial zum Objektmittelpunkt M_{Objekt}:

$$F_K = \begin{cases} m\,d\ddot{x} + k_O\,dx & \text{für} \quad d_x = r - d_s > 0 \\ 0 & \text{sonst.} \end{cases} \tag{4.10}$$

Wobei d_x die Objektverformung infolge der Kontaktsituation, r den Radius des Objekts und d_s das mit Gleichung (4.13) berechnete Abstandsmaß darstellt.

4.2.1.6 Kontaktmodellierung

Neben den zu greifenden Objekten ist weiterhin die Kontaktsituation zwischen Robotergreifer und Objekt zu modellieren. In der Literatur existieren zahlreiche Ansätze zur Kontaktmodellierung [201, 239, 254]. In dieser Arbeit wird ein reibungsbehaftetes Kontaktmodell verwendet, das sowohl Flächen- als auch Punktkontakt beinhaltet. Bild 4.5 zeigt beispielhaft die beiden Kontaktarten (Flächenkontakt links und Punktkontakt rechts) des Robotergreifers mit dem durch Gleichung (4.10) definierten Objekt inkl. geometrischer Berechnung der Objektverformung d_x. Punktkontakt bedeutet dabei in dieser Arbeit, dass der Robotergreifer nur mit der Fingerspitze x_i^{tip} das Objekt kontaktiert. Befinden sich zusätzlich zu der Fingerspitze auch Teile einzelner Fingerglieder in Kontakt mit dem zu greifenden Objekt, dann handelt es sich um einen Flächenkontakt.

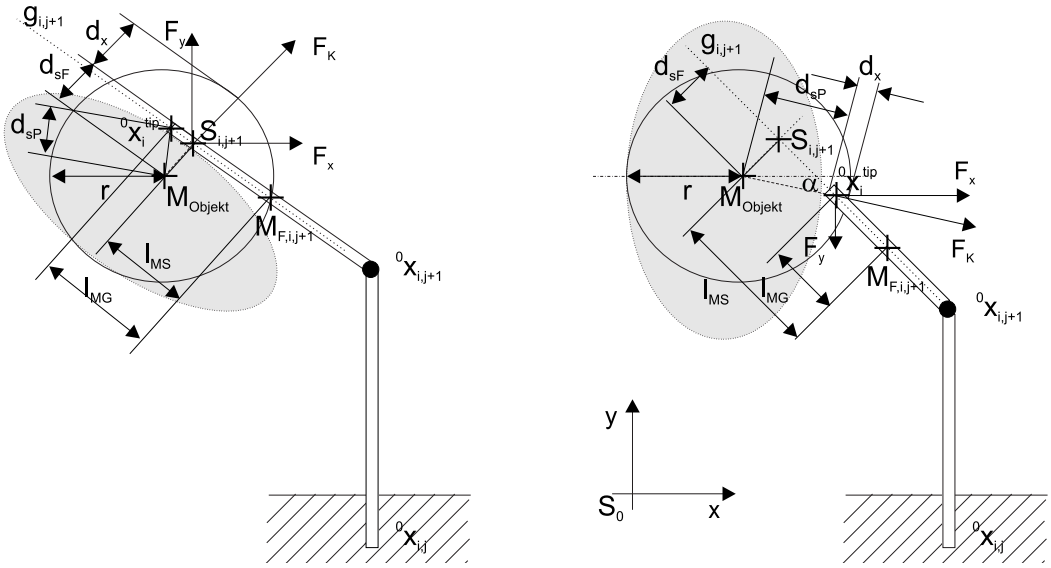


Bild 4.5: Flächenkontakt (links) und Punktkontakt (rechts) des Robotergreifers mit einem verformbaren Objekt (grau)

Als Schnittpunkt $S_{(i,j)}$ wird die Position mit dem minimalen Abstand des Objektmittelpunkts M_{Objekt} zu der Geraden $g_{(i,j)}$ definiert. Die Gerade wird dabei mit Hilfe der kartesischen Koordinaten zweier aufeinander folgender Gelenkpunkte ${}^0x_{(i,j)}$ und ${}^0x_{(i,j+1)}$ bestimmt. Mit d_{sF} wird das lokale Minimum des Abstandes $(M_{Objekt}, S_{(i,j)})$ und mit d_{sP} das lokale Minimum des Abstandes $(M_{Objekt}, {}^0\,x_i^{tip})$ bezeichnet:

$$\begin{aligned} d_{sF} &= \|M_{Objekt}^{WKS} - S_{(i,j)}^{WKS}\|_2 \\ d_{sP} &= \|M_{Objekt}^{WKS} - {}^0\,x_i^{tip}\|_2. \end{aligned} \tag{4.11}$$

Mit l_{MS} wird weiterhin der Abstand des Gelenkmittelpunkts $M_{F(i,j)}$ vom Schnittpunkt $S_{(i,j)}$ und mit l_{MG} der Abstand des Gelenkmittelpunkts $M_{F(i,j)}$ von den Koordinaten der Fingerspitze ${}^0x_i^{tip}$ bezeichnet:

$$\begin{aligned} l_{MS} &= \|M_{F(i,j)}^{WKS} - S_{(i,j)}^{WKS}\|_2 \\ l_{MG} &= \|M_{F(i,j)}^{WKS} - {}^0\,x_i^{tip}\|_2. \end{aligned} \tag{4.12}$$

Die Gleichungen (4.11) und (4.12) führen auf das zur Berechnung der Objektverformung dx benötigte Abstandsmaß d_s:

$$d_s = \begin{cases} d_{sF} & \text{für} \quad l_{MS} \leq l_{MG} \\ d_{sP} & \text{für} \quad l_{MS} > l_{MG}. \end{cases} \tag{4.13}$$

Mit Hilfe des Abstandsmaßes d_s aus Gleichung (4.13) lässt sich dann die Kontaktart bestimmen:

$$\text{Kontaktart} = \begin{cases} \text{Flächenkontakt} & \text{für} \quad d_s \leq r \wedge l_{MS} \leq l_{MG} \\ \text{Punktkontakt} & \text{für} \quad d_s \leq r \wedge l_{MS} > l_{MG} \\ \text{kein Kontakt} & \text{sonst.} \end{cases} \tag{4.14}$$

Bei beiden Modellen erfolgt die Berechnung der Kontaktkraft F_K nach Gleichung (4.10). Der Unterschied zwischen beiden Modellen ergibt sich aus der geometrischen Berechnung der Objektverformung d_x und der Wirkungsrichtung der Kontaktkraft F_K (vgl. Bild 4.5).

Weiterhin muss bei Objektkontakt die Kopplung der Gelenkmomente $M_{z(i,j)}$ und $M_{z(i,j+1)}$ berücksichtigt werden, da die an einem distalen Fingerglied angreifenden externen Kräfte auch die proximalen Gelenke beeinflussen (Bild 4.5). Durch Freischneiden des Robotergreifers lassen sich die in den Gelenken auftretenden Kräfte und Momente berechnen [108]. Bild 4.6 zeigt das Gesamtmodell eines Fingers mit zwei Gelenken unter Einwirkung einer externen Kraft F_K am distalen Fingerglied (Teilkörper 2).

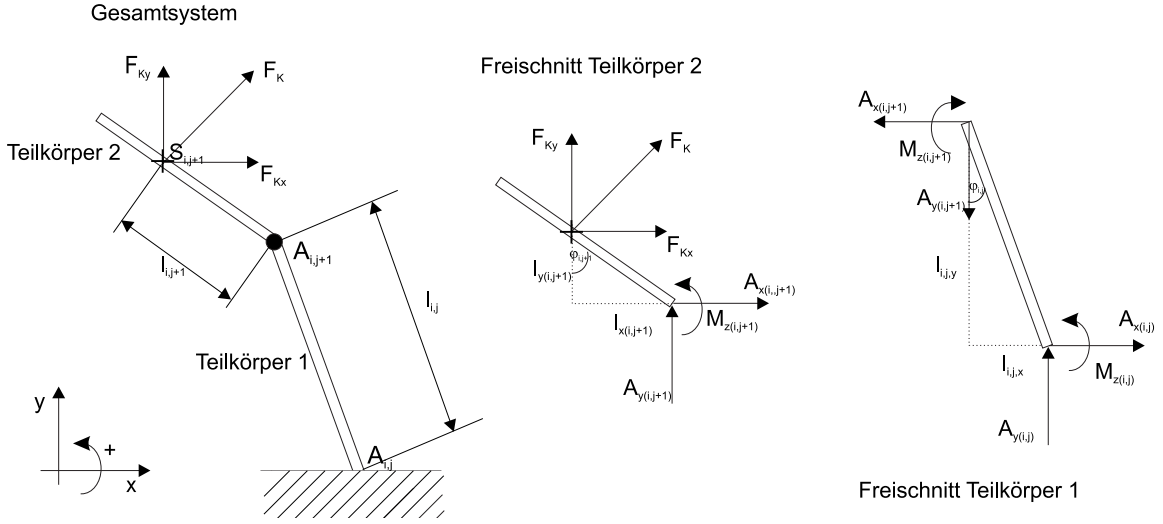


Bild 4.6: Gesamtmodell (links) und Freischnitte (Mitte und rechts) eines Fingers mit zwei Gelenken unter Einwirkung einer externen Kraft F_K

Allgemein gelten folgende Gleichgewichtsbedingungen für die freigeschnittenen Teilkörper:

$$\sum F_{Kx(i,j)} = 0 \quad \sum F_{Ky(i,j)} = 0 \quad \sum M_{z(i,j)} = 0. \tag{4.15}$$

Mit den Strecken $l_{y(i,j+1)}$ und $l_{x(i,j+1)}$ aus Bild 4.6 (Mitte) berechnet sich das resultierende Moment $M_{z(i,j+1)}$ am distalen Gelenk zu:

$$M_{z(i,j+1)} = F_{Kx} \underbrace{l_{(i,j+1)} \cos(\varphi_{(i,j+1)})}_{l_{y(i,j+1)}} + F_{Ky} \underbrace{l_{(i,j+1)} \sin(\varphi_{(i,j+1)})}_{l_{x(i,j+1)}}. \tag{4.16}$$

Wird die Gleichung (4.15) auch auf den zweiten Teilkörper angewendet, so ergibt sich unter Verwendung der Strecken $l_{y(i,j)}$ und $l_{x(i,j)}$ aus Bild 4.6 (rechts) das gesuchte Moment $M_{z(i,j)}$ am proximalen Fingergelenk in Abhängigkeit von der Fingerkinematik, der aktuellen Gelenkstellung und der einwirkenden Kraft:

$$M_{z(i,j)} = M_{z(i,j+1)} + F_{Kx} \underbrace{l_{(i,j)} \cos(\varphi_{(i,j)})}_{l_{y(i,j)}} + F_{Ky} \underbrace{l_{(i,j)} \sin(\varphi_{(i,j)})}_{l_{x(i,j)}}. \tag{4.17}$$

F_K, F_{Kx} und F_{Ky} bezeichnen dabei die Kontaktkraft und ihre x- und y-Komponente in der Ebene. $A_{(i,j)}, A_{x(i,j)}$ und $A_{y(i,j)}$ bezeichnen die Lagerkräfte in den Gelenken sowie ihre x- und y-Komponente in der Ebene. $M_{z(i,j)}$ ist das resultierende Moment am Gelenk $A_{(i,j)}$. Die Strecken $l_{(i,j)}$ bezeichnen die Abstände von $\|^0 x_{(i,j)} -^0 x_{(i,j+1)}\|_2$ (Punktkontakt) bzw. $\|^0 x_{(i,j+1)} - S_{(i,j+1)}\|_2$ (Flächenkontakt).

Alternativ kann bei Verwendung von MSC@ADAMS als Modellbildungstool dessen leistungsfähige dynamische 3D-Kontaktmodellierung zur Simulation von Greifbewegungen verwendet werden. Dazu wird MSC@ADAMS über eine vordefinierte Schnittstelle in das vorhandene SIMULINK®-Modell integriert [203].

4.2.2 Modellbasierte Regelung

4.2.2.1 Übersicht

Auf der Basis der in den vorigen Abschnitten vorgestellten mathematischen Modellen werden in diesem Abschnitt Algorithmen zur Positions- und/oder Kraftregelung der Freiheitsgrade des Robotergreifers vorgestellt sowie die Vor- und Nachteile dieser Konzepte diskutiert. Dies beinhaltet die Beschreibung einer allgemeinen Grundstruktur eines diskret-kontinuierlichen Regelungskonzepts. Die Umsetzung erfolgt durch verschiedene aus der Literatur bekannte Ansätze, zu denen sowohl Nachgiebigkeits- [17, 185, 215, 216, 218, 254], als auch hybride Kraft-Positions-Regelungskonzepte [222, 254] zählen. Die im Weiteren vorgestellten Regelungskonzepte sind für den im Abschnitt 4.2.1 vorgestellten Robotergreifer entwickelt, lassen sich z. T. aber auch auf andere Mehrfinger-Robotergreifer übertragen.

Im Weiteren wird dabei von der in Abschnitt 4.2.1.4 beschriebenen Variante I mit voller sensorischer Ausstattung ausgegangen. Werden zur Kontaktdetektion und Störmomentberechnung modellbasierte Verfahren verwendet (z. B. modellbasierte Zustandserkennung nach [35]), dann können die hier vorgestellten Regelungsalgorithmen auch mit der Variante II (fehlende taktile Sensorik) umgesetzt werden. Bei der Variante III, ohne integrierte Sensorik, ist dagegen keines der hier beschriebenen Verfahren anwendbar [228] (Abschnitt 4.2.1.4).

Um mit einem Robotergreifer Gegenstände zu greifen, werden die von der Trajektoriengenerierung vorgegebenen Referenztrajektorien (Gelenkwinkel und Gelenkmomente über der Zeit) von Reglern auf der Gelenkebene umgesetzt. Zu beachten ist hierbei, dass sich durch die verwendete Antriebsart zusätzliche Einschränkungen ergeben, wenn für die Regelung *mehrerer* Gelenke nur *eine* unabhängige Druckversorgung zur Verfügung steht. Dadurch ergeben sich Restriktionen, welche bei der Auswahl des Regelungskonzepts zu berücksichtigen sind. So kann sich der Regelungsalgorithmus im hydraulischen Betrieb pro Abtastintervall nur für eine Pumpenrichtung und damit für ein entsprechendes pulsweitenmoduliertes Steuersignal x_P entscheiden. Damit ist es nur in Sonderfällen möglich, in einem Intervall gegenläufige Bewegungen verschiedener Fingergelenke zu realisieren. Bild 4.7 zeigt die sich ergebende Struktur von Regler und Regelstrecke (Antriebssystem, mechanisches System und Objektmodell) für den fluidisch angetriebenen Robotergreifer.

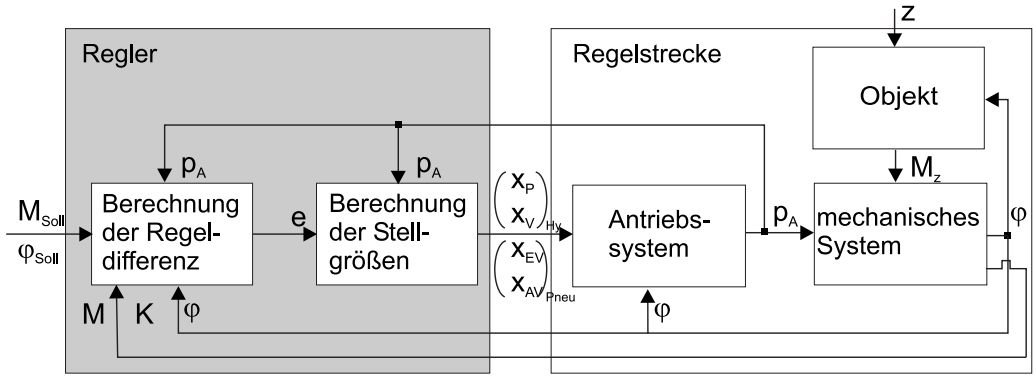


Bild 4.7: Schematische Darstellung von Regler und Regelstrecke (Antriebssystem, mechanisches System und Objektmodell) für den fluidisch angetriebenen Robotergreifer (Variante I)

Die allgemeine Struktur des entwickelten Regelungskonzepts gliedert sich in die zwei Teilsysteme, die Berechnung der Regeldifferenz und die nachfolgende Berechnung der Stellgrößen [64, 170]. Eingangsgrößen des Blocks Berechnung der Regeldifferenz sind die von der Trajektoriengenerierung berechneten Führungsgrößen $\boldsymbol{\varphi}_{Soll}$ und $\mathbf{M}_{Soll}$, die derzeitigen Drücke in den Aktoren $\mathbf{p_A}$, die aktuellen Gelenkwinkel $\boldsymbol{\varphi}$ und Gelenkmomente $\mathbf{M}$ sowie der Kontaktzustand $\mathbf{K} \in [0, 1]$. Ausgangsgröße ist die berechnete Regeldifferenz $\mathbf{e}$. Aufgabe des Blocks Berechnung der Stellgrößen ist es, die berechnete Regeldifferenz $\mathbf{e}$ zu minimieren. Die dazu zur Verfügung stehenden Stellgrößen sind dabei in Abhängigkeit von der Antriebsart die pulsweitenmodulierte Spannung der Pumpe x_P und die binären Steuerspannungen der Ventile $\mathbf{x_V}$ (Hydraulik) bzw. die binären Steuerspannungen der Einlass- und Auslassventile $\mathbf{x_{EV}}$ und $\mathbf{x_{AV}}$ (Pneumatik). Optional können die Ventile auch pulsweitenmoduliert angesteuert werden [106]. Dieser Ansatz wird hier allerdings nicht weiter untersucht.

4.2.2.2 Berechnung der Regeldifferenz

Eine bedeutende Anforderung an die Regelung eines Robotergreifers ist die Möglichkeit zur Interaktion mit der Umwelt. Dies ist wichtig für eine Reihe von praktisch relevanten Aufgaben, bei denen der Robotergreifer Gegenstände zu greifen, zu manipulieren und wieder abzusetzen hat (engl.: Pick and Place, Catch and Ferry, Assembly, ...).

Strategien zur Regelung der Interaktion sind nach [254] prinzipiell in zwei Gruppen einteilbar:

- Indirect force control:
  Kraftregelung über die Regelung der Position, ohne explizite Rückführung und damit Regelung der Kontaktkraft (z. B. Nachgiebigkeitsregelungen). Bei reiner Positionsregelung ergibt sich bei Kontakt mit der Umwelt eine Kontaktkraft, welche zu einer Abweichung des Manipulators von der vorgegebenen Trajektorie führt. Aufgabe des Reglers ist es diese Abweichung zu minimieren. Dies führt zu einem Ansteigen der Kontaktkraft bis die maximale Stellgröße der Aktoren erreicht ist, oder der Bruch eines der an der Interaktion beteiligten Teile erfolgt. Aus dieser Sicht ist ein nachgiebiges Verhalten der an der Interaktion beteiligten Systeme (Roboter, Umwelt) wünschenswert. Dabei ist zwischen aktiver und passiver Nachgiebigkeit zu unterscheiden [254]. Bei der passiven Nachgiebigkeit ist die mechanische Struktur des Manipulators selbst nachgiebig. Bei der aktiven Nachgiebigkeit wird das gewünschte Verhalten des Manipulators durch den Regler realisiert.
- Direct force control:
  Schließen des Regelkreises durch Rückführung der Kontaktkraft. Führungsgröße des Regelkreises ist die geforderte Kontaktkraft (z. B. parallele-, hybride Kraft-Positionsregelungen).

In [64] wurden diese Konzepte im Hinblick auf die Regelung der Interaktion des fluidisch angetriebenen Robotergreifers mit der Umwelt untersucht. Die für das Durchführen von Greifvorgängen am besten geeigneten Konzepte (Nachgiebigkeitsregelung und hybride Kraft-Positionsregelung) werden im Weiteren vorgestellt.

Nachgiebigkeitsregelung Die Nachgiebigkeit eines Roboter-Manipulators ist als das Verhältnis der äußeren Kräfte zu der Positionsabweichung definiert. Steife Systeme haben den Vorteil, größere Kräfte umsetzen zu können. Bei der Einwirkung zu großer Kräfte besteht allerdings die Gefahr der Zerstörung des Manipulators oder der Umgebung. Deshalb ist in vielen Fällen eine Verminderung der Steifigkeit des Manipulators gewünscht. Bei dem fluidisch angetriebenen Robotergreifer ergibt sich eine passive Nachgiebigkeit aufgrund der flexiblen Fluidaktoren (vgl. Abschnitt 4.2.1.3). Untersuchungen in [183] haben gezeigt, dass das hydraulische System dabei eine fast doppelt so große maximale Steifigkeit aufweist wie das pneumatische System. Somit wird eine aktive Nachgiebigkeitsregelung bei einem solchen flexiblen System nur zur *Änderung* der konstruktiv festgelegten Nachgiebigkeit eingesetzt.

Aktive Konzepte lassen sich weiter in Admittanz-, Impedanz- und Steifigkeitskonzepte unterteilen [64, 254]. Bei der Admittanzregelung werden die Regelabweichungen zwischen geforderten und existierenden Kontaktkräften mit Verstärkungs- und Dämpfungsmatritzen in Positionsabweichungen transformiert. Damit ist die sich ergebende Stellgröße ein Kompromiss aus vorgegebener Position und gewünschter Kontaktkraft. Über die Parametrierung der Verstärkungs- und Dämpfungsmatritzen kann das elastische Verhalten des Robotersystems eingestellt werden [17, 205, 239]. Im Gegensatz zur Admittanzregelung, bei der die vorgegebenen Sollkräfte zu Positionsänderungen des Endeffektors führen, werden bei der Impedanzregelung durch Vorgabe der Positionssollwerte die entsprechenden Kontaktkräfte bestimmt. Die Steifigkeitsregelung ist eine vereinfachte Variante der Admittanzregelung, bei der die Transformation von Kraftabweichung zu Positionsabweichung nur mit einer Verstärkungsmatrix berechnet wird [17]. In [64] wurden die beschriebenen Nachgiebigkeitskonzepte zur Regelung des fluidisch angetriebenen Robotergreifers umgesetzt und miteinander verglichen. Das für das Durchführen von Greifvorgängen am besten geeignete Konzept (Admittanzregelung im Gelenkwinkelraum) wird hier vorgestellt. Bild 4.8 zeigt die detaillierte Struktur der Admittanzregelung.

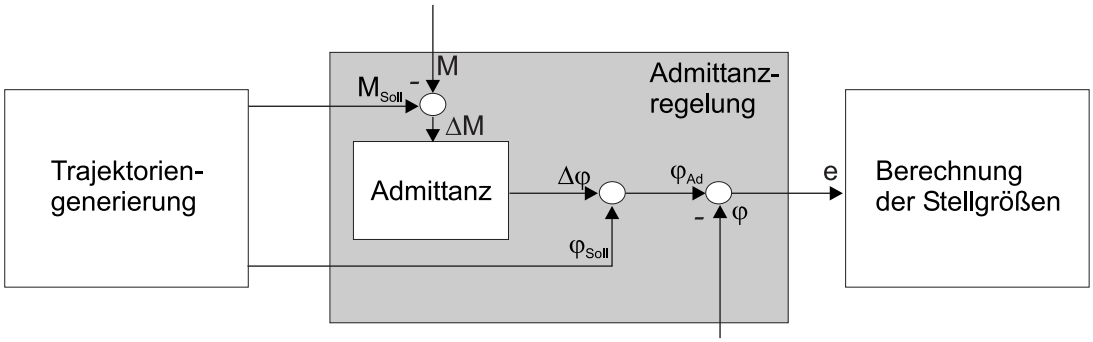


Bild 4.8: Detaillierte Struktur der Admittanzregelung

Bei einer Admittanzregelung gilt für die Momentendifferenz $\Delta M_{(i,j)}$:

$$\Delta M_{(i,j)}[k] = M_{Soll(i,j)}[k] - M_{(i,j)}[k]. \tag{4.18}$$

Die Gelenkwinkeldifferenz $\Delta\varphi_{(i,j)}$ berechnet sich aus $\varphi_{Ad(i,j)}$ und $\varphi_{Soll(i,j)}$:

$$\Delta\varphi_{(i,j)}[k] = \varphi_{Ad(i,j)}[k] - \varphi_{Soll(i,j)}[k]. \tag{4.19}$$

Der Zusammenhang zwischen Momentendifferenz $\Delta M_{(i,j)}$ und Gelenkwinkeldifferenz $\Delta\varphi_{(i,j)}$, im Weiteren Admittanz genannt, ist durch folgende Gleichung gegeben:

$$\Delta M_{(i,j)}[k] = K_{Ad}\,\Delta\varphi_{(i,j)}[k] + D_{Ad}\,\Delta\dot{\varphi}_{(i,j)}[k]. \tag{4.20}$$

Darin bezeichnet K_{Ad} den Steifigkeitsparameter und D_{Ad} den Dämpfungsparameter der Admittanz. Über diese Parameter lässt sich die Nachgiebigkeit des Robotergreifers einstellen.

Einsetzen von Gl. (4.18) und Gl. (4.19) in Gl. (4.20) und Umformen ergibt folgenden rekursiven, diskreten Algorithmus zur Berechnung der modifizierten Führungsgröße $\varphi_{Ad(i,j)}[k]$ in Abhängigkeit von den gewählten Nachgiebigkeitsparametern K_{Ad}, D_{Ad} und der Abtastzeit t_A [64]:

$$\begin{aligned}\varphi_{Ad(i,j)}[k] = \varphi_{Soll(i,j)}[k] + \frac{t_A}{D_{Ad} + t_A\,K_{Ad}}(M_{Soll(i,j)}[k] - M_{(i,j)}[k]) \\ + \frac{D_{Ad}}{D_{Ad} + t_A\,K_{Ad}}(\varphi_{Ad(i,j)}[k-1] - \varphi_{Soll(i,j)}[k-1]).\end{aligned} \tag{4.21}$$

Während der freien Bewegung, d. h. ohne Objektkontakt und bei korrekter Planung des auszuführenden Griffs, sind die von der Bahnplanung vorgegebenen Gelenkwinkel $\varphi_{Soll(i,j)}$ vom Regelungsalgorithmus einzuregeln, während die erforderlichen Sollmomente $M_{Soll(i,j)}$ bei korrekter Planung verschwinden. Damit arbeitet der Admittanz-Regler als reiner Positionsregler, da $\Delta M_{(i,j)} = 0$ und damit automatisch $\varphi_{Ad(i,j)} = \varphi_{Soll(i,j)}$ gilt.

Bei Objektkontakt und korrekter Planung ergibt sich ein $\Delta M_{(i,j)} \neq 0$. Diese Momentendifferenz wird mit Hilfe von Gl. (4.20) in eine Gelenkwinkeldifferenz $\Delta\varphi_{(i,j)}$ umgerechnet. In Abhängigkeit von den frei wählbaren Parametern K_{Ad} und D_{Ad} ergibt sich daraus ein $\Delta\varphi_{(i,j)} \neq 0$. Addition von $\varphi_{Soll(i,j)}$ und $\Delta\varphi_{(i,j)}$ führt dann nach Gl. (4.21) zu einer modifizierten Gelenkwinkelführungsgröße $\varphi_{Ad(i,j)} \neq \varphi_{Soll(i,j)}$. Aus der Differenz von $\varphi_{Ad(i,j)}$ und $\varphi_{(i,j)}$ berechnet sich dann die Regeldifferenz $e_{(i,j)}$ als Eingangsgröße des nachfolgenden Regelungsalgorithmus.

Hybride Kraft-Positionsregelung Eine weitere Möglichkeit die Regeldifferenz zu berechnen und damit die Interaktion zwischen Robotergreifer und Umgebung (Objekt) auf gewünschte Weise zu regeln, ist die Verwendung eines hybriden Kraft-Positionsregelungskonzepts. Hybride Kraft-Positionsregelungen ermöglichen eine getrennte Regelung der Position und der Kraft für jeden Freiheitsgrad. Neben den hybriden Kraft-Positionsregelungen existieren in der Literatur weiterhin parallele Kraft-Positionsregelungen [254], die in ihrem Aufbau und ihrer Wirkungsweise den Nachgiebigkeitskonzepten aus Abschnitt 4.2.2.2 entsprechen und auf die deshalb hier nicht weiter eingegangen wird. Bild 4.9 zeigt die detaillierte Struktur der hybriden Kraft-Positionsregelung.

Um eine definierte Kraft auf ein zugreifendes Objekt auszuüben, muss im Kontaktfall ein vorgegebenes Gelenkmoment $M_{Soll(i,j)}$ von der Regelung eingeregelt werden. Um dies zu realisieren, wird die Differenz aus vorgegebenen Gelenkmoment $M_{Soll(i,j)}$ und gemessenem Störmoment $M_{(i,j)}$ (Resultierend aus dem Objektkontakt) berechnet. Ist der Robotergreifer mit Kraftsensoren ausgerüstet (vgl. Abschnitt 4.2.1.4), kann über die von dem entsprechenden Sensor gemessene Kontaktkraft F_K und die bekannte Sensorposition auf dem Fingerglied $l_{(i,j)}$ das Störmoment $M_{(i,j)}$ analog zu der Gleichung (4.16) bzw. (4.17) berechnet werden. Ist keine taktile Sensorik vorhanden, muss die erforderliche Berechnung des Störmoments modellbasiert erfolgen (z. B. über eine modellbasierte Zustandserkennung [35]). Kräfte, die außerhalb der Wirkungsrichtung der Gelenkfreiheitsgrade auf den Robotergreifer einwirken, werden durch einen im Handgelenk des Roboters verbauten 6-DOF Kraft-Momenten-Sensor erfasst und von der hybriden Kraft-Positionsregelung des Roboterarms ausgeregelt [154]. Weiterhin wird mit Hilfe

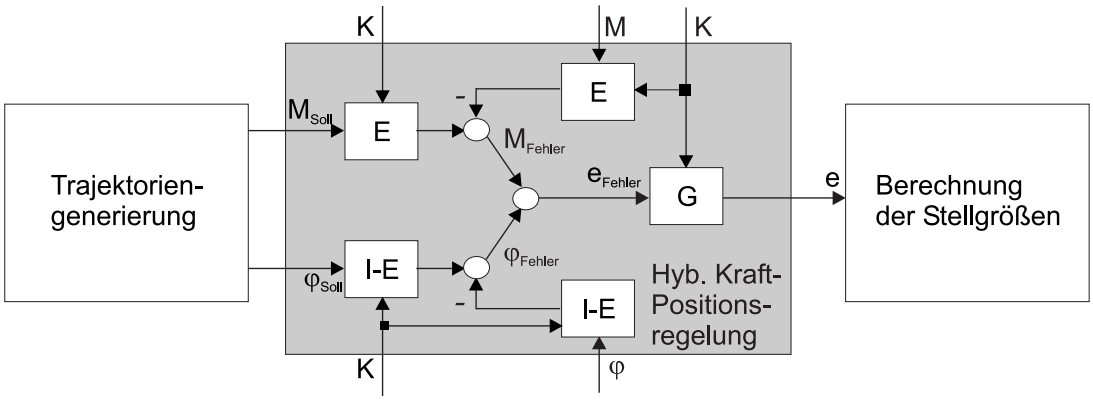


Bild 4.9: Detaillierte Struktur der hybriden Kraft-Positionsregelung

der taktilen Sensorik der Kontaktzustand berechnet. Überschreitet die gemessene Kontaktkraft F_K den Schwellwert $F_{Schwell}$, dann besteht ein Kontakt mit der Umgebung. Mit Hilfe der Funktion $K_{(i,j)}$ wird dabei die Kontaktsituation beschrieben:

$$K_{(i,j)}[k] = \begin{cases} 1 = \text{Kontakt} & \text{für} \quad F_K > F_{Schwell} \\ 0 = \text{kein Kontakt} & \text{sonst.} \end{cases} \tag{4.22}$$

Damit ergibt sich für die bei bestehendem Objektkontakt zu minimierende Regelabweichung:

$$M_{Fehler(i,j)}[k] = M_{Soll(i,j)}[k]\, K_{(i,j)}[k] - M_{(i,j)}[k]\, K_{(i,j)}[k]. \tag{4.23}$$

Bei freier Bewegung ist die zu minimierende Regelabweichung die Differenz der Führungsgrößenvorgabe $\varphi_{Soll(i,j)}$ und der mit der Winkelsensorik gemessenen Gelenkwinkel $\varphi_{(i,j)}$:

$$\varphi_{Fehler(i,j)}[k] = \varphi_{Soll(i,j)}[k]\,(1 - K_{(i,j)}[k]) - \varphi_{(i,j)}[k]\,(1 - K_{(i,j)}[k]). \tag{4.24}$$

Im Mehrgrößenfall erfolgt die Auswahl der Gelenke für den momentengeregelten (mit Objektkontakt) bzw. den positionsgeregelten (freie Bewegung) Modus nicht mit der Gleichung (4.22), sondern mit Hilfe der Selektions- $\mathbf{E}$ und der Einheitsmatrix $\mathbf{I}$ (vgl. Bild 4.9). Für diese Matrizen gelten:

$$\mathbf{E} = \begin{pmatrix} K_{(1,1)} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & K_{(n_F,m_i)} \end{pmatrix} \quad \mathbf{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Um die Größenordnungsunterschiede der Momenten- und Gelenkwinkeldifferenzen auszugleichen, wird weiterhin eine neue empirisch ermittelte Wichtungsfunktion $g_{W(i,j)}(K_{(i,j)})$ definiert:

$$g_{W(i,j)}[k] = (\frac{\pi}{1.2})\, K_{(i,j)}[k] + (1 - K_{(i,j)}[k]). \tag{4.25}$$

Als Eingangsgröße des Blocks *Berechnung der Stellgrößen* folgt damit:

$$e_{(i,j)}[k] = (\varphi_{Fehler(i,j)}[k] + M_{Fehler(i,j)}[k])\, g_{W(i,j)}[k]. \tag{4.26}$$

4.2.2.3 Berechnung der Stellgrößen

Die zur Minimierung der Regeldifferenz zur Verfügung stehenden Stellgrößen hängen wie bereits beschrieben von der verwendeten Antriebsart ab. So sind im hydraulischen Betrieb die pulsweitenmodulierte Spannung der Pumpe x_P sowie die binären Steuerspannungen der Hydraulik-Ventile $\mathbf{x_V}$ und im pneumatischen Betrieb die binären Steuerspannungen der Einlass- und Auslassventile $\mathbf{x_{EV}}$ und $\mathbf{x_{AV}}$ durch den entsprechenden Regelungsalgorithmus zu verändern. Im Weiteren erfolgt deshalb zunächst die Darstellung des in [170] erstmals präsentierten neuen kontinuierlich-ereignisdiskreten Regelungsalgorithmus zur Berechnung der Stellgrößen im hydraulischen Betrieb. Im Anschluss daran wird kurz der verwendete Zweipunktregler zur Berechnung der Stellgrößen im pneumatischen Betrieb vorgestellt [106].

Hydraulik Die Aufgabe des kontinuierlich-ereignisdiskreten Hydraulik-Regelungsalgorithmus ist es, aus der berechneten Regeldifferenz die pulsweitenmodulierten Steuerspannungen für die Pumpe und die Ventile zu generieren. Die Idee des entwickelten hybriden Regelungsalgorithmus beruht darauf, zu jedem Abtastzeitpunkt eine Gruppe von Aktoren mit ähnlichen Anforderungen an die Pumpe zusammenzufassen und die zugehörigen Ventile zu öffnen. Die Pumpe wird dann entsprechend eines Kompromisses zwischen diesen Aktoren angesteuert. Wenn die Aktoren bzw. die Gelenke sich innerhalb von vorgegebenen Totzonenbereichen $e_{tot_{aus}}$ und $e_{tot_{ein}}$ mit $e_{tot_{aus}} > e_{tot_{ein}}$ (Hysterese) befinden, scheiden sie aus der Gruppe aus. Für die Berechnung der Totzonenbereiche ergibt sich wiederum in Abhängigkeit von der Kontaktsituation $K_{(i,j)}$ [170]:

$$e_{tot_{aus}}[k] = \alpha[k]\,\varphi_{tot} + \beta[k]\,M_{tot} \qquad e_{tot_{ein}}[k] = \frac{e_{tot_{aus}}[k]}{2} \tag{4.27}$$

mit

$$K_{(i,j)}[k] = \begin{cases} 1 & \Rightarrow \quad \alpha[k] = 0 \wedge \beta[k] = 1 \\ 0 & \Rightarrow \quad \alpha[k] = 1 \wedge \beta[k] = 0. \end{cases} \tag{4.28}$$

Die vom Regelungsalgorithmus zu tolerierenden Regelgrößenabweichungen werden mit φ_{tot} und M_{tot} bezeichnet. Mit den Ergebnissen aus Gleichung (4.27) berechnet sich dann die Regeldifferenz $e^*_{(i,j)}$:

$$e^*_{(i,j)}[k] = \begin{cases} 0 & \text{für} \quad (|e_{(i,j)}[k]| < e_{tot_{ein}}[k]) \\ & \vee \quad (|e_{(i,j)}[k]| < e_{tot_{aus}}[k] \wedge |e^*_{(i,j)}[k-1]| = 0) \\ e_{(i,j)}[k] & \text{sonst.} \end{cases} \tag{4.29}$$

Die Gruppenbildung erfolgt über alle Aktoren bzw. Gelenke mit positiver bzw. negativer Regeldifferenz:

$$E_{pos}[k] = \sum_{i=1}^{n_F} \sum_{j=1}^{m_i} \begin{cases} e^*_{(i,j)}[k] & \text{für} \quad e^*_{(i,j)}[k] > 0 \\ 0 & \text{sonst.} \end{cases} \tag{4.30}$$

$$E_{neg}[k] = \sum_{i=1}^{n_F} \sum_{j=1}^{m_i} \begin{cases} e^*_{(i,j)}[k] & \text{für} \quad e^*_{(i,j)}[k] < 0 \\ 0 & \text{sonst.} \end{cases} \tag{4.31}$$

Der Regelungsalgorithmus schaltet dann ereignisdiskret zu jedem Abtastzeitpunkt t_A für jedes Verhältnis von E_{pos} zu E_{neg} in Abhängigkeit von dem Schaltfaktor k_{switch} ($k_{switch} > 1$) und seinem derzeitigen Zustand. Bild 4.10 zeigt die Transitionen des implementierten Zustandsautomaten. Im Pumpenzustand 1 werden alle Aktoren der E_{pos}-Gruppe, im Pumpenzustand -1 alle Aktoren der E_{neg}-Gruppe geregelt. Im Pumpenzustand 0 wird die Pumpe ausgeschaltet.

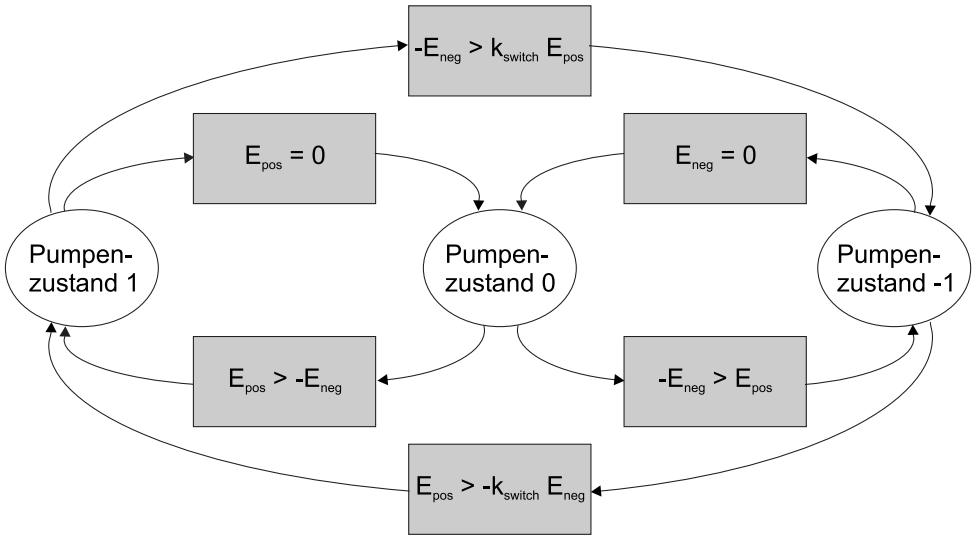


Bild 4.10: Zustandsautomat des Regelungsalgorithmus mit k_{switch} als Konstante zur Prioritätenumschaltung [170]

In Abhängigkeit vom geschalteten Pumpenzustand berechnet sich die Steuerspannung der Pumpe mit Hilfe der Regeldifferenzen E_{pos} bzw. E_{neg} nach folgendem zeitdiskreten PI-Algorithmus [170]:

$$x_P[k+1] = K_R[k+1]E_{(pos,neg)}[k+1] + u_I[k+1] \tag{4.32}$$

$$u_I[k+1] = u_I[k] + \frac{K_R[k]}{T_N[k]} t_A \, E_{(pos,neg)}[k]. \tag{4.33}$$

Die Adaption der Reglerparameter K_R und T_N aus den Gleichungen (4.32) und (4.33) erfolgt wiederum in Abhängigkeit des Kontaktzustands $K_{(i,j)}$ und kann später um weitere Größen, wie z. B. aktuelle Gelenkwinkel und -momente, erweitert werden:

$$K_R[k] = \alpha[k]\, K_{RF} + \beta[k]\, K_{RP} \qquad T_N[k] = \alpha[k]\, T_{NF} + \beta[k]\, T_{NP}. \tag{4.34}$$

K_{RF} und K_{RP} bezeichnen die Reglerverstärkung und T_{NF} und T_{NP} die Zeitkonstante des PI-Regelungsalgorithmus im positions- bzw. im momentengeregelten Modus.

Die Steuerspannung der Ventile $x_{V_{(i,j)}}$ ergibt sich in Abhängigkeit von der Regeldifferenz und dem derzeitigen Pumpenzustand:

$$x_{V_{(i,j)}}[k] = \begin{cases} 1 = \text{Ventil offen} & \text{für} \quad e^*_{(i,j)}[k] > 0 \wedge \text{Pumpenzustand} = 1 \\ & \vee \quad e^*_{(i,j)}[k] < 0 \wedge \text{Pumpenzustand} = -1 \\ 0 = \text{Ventil geschlossen} & \text{sonst.} \end{cases} \tag{4.35}$$

Druckdifferenzen zwischen Eingang und Ausgang der Pumpe führen dazu, dass sich trotz einer positiven Steuerspannung x_P ein negativer Durchfluss ergibt, wenn die Pumpe gegen einen zu hohen Druck p_{VB} fördert. Aus diesem Grund sollen in einem weiteren Schritt in die Entscheidung des Reglers, ob die Ventile geöffnet oder geschlossen werden, nicht nur die Regelabweichungen, sondern auch die aktuell anliegenden Druckdifferenzen zwischen Pumpe, Ventilen und Aktoren einbezogen werden. Die

Steuerspannung der Ventile $x_{V(i,j)}$ ergibt sich dann in Abhängigkeit von der Regeldifferenz und der Druckdifferenz vor und hinter dem Ventil:

$$x_{V(i,j)}[k] = \begin{cases} 1 = \text{Ventil offen} & \text{für} \quad e^*_{(i,j)}[k] > 0 \wedge (p_{VB} > p_{A(i,j)}) \\ & \vee \quad e^*_{(i,j)}[k] < 0 \wedge (p_{VB} < p_{A(i,j)}) \\ 0 = \text{Ventil geschlossen} & \text{sonst.} \end{cases} \tag{4.36}$$

Bei Handvarianten ohne integrierte Druck- und Kraftsensorik ist zwar eine Positionsregelung möglich, die Leistungsfähigkeit bei der Kraft-Momentenregelung und Kontakterkennung aber sehr eingeschränkt. Hier besteht lediglich die Möglichkeit, entsprechende Grobinformationen aus den zeitlichen Winkelverläufen bei bekannten Pumpen und Ventilzuständen zu extrahieren.

Pneumatik Die Aufgabe des Pneumatik-Regelungsalgorithmus ist es, aus der berechneten Regeldifferenz die binären Steuerspannungen der Einlass- und Auslassventile zu generieren. Da die Stellgrößen im pneumatischen Betrieb binär sind, eignet sich bereits ein einfacher Zweipunkt-Regelungsalgorithmus mit Totzonenbereich [106]. Ist die Regeldifferenz $e_{(i,j)}$ größer als der vorgegebene Totzonenbereich $e_{tot_{aus}}$ wird das Einlassventil geöffnet und das Auslassventil geschlossen. Ist die Regeldifferenz $e_{(i,j)}$ dagegen kleiner als der negative Totzonenbereich $-e_{tot_{aus}}$, dann wird das Einlassventil geschlossen und das Auslassventil geöffnet.

$$x_{EV(i,j)}[k] = \begin{cases} 1 = \text{Ventil offen} & \text{für} \quad e_{(i,j)}[k] > e_{tot_{aus}} \\ 0 = \text{Ventil geschlossen} & \text{sonst.} \end{cases} \tag{4.37}$$

$$x_{AV(i,j)}[k] = \begin{cases} 1 = \text{Ventil offen} & \text{für} \quad e_{(i,j)}[k] < -e_{tot_{aus}} \\ 0 = \text{Ventil geschlossen} & \text{sonst.} \end{cases} \tag{4.38}$$

4.2.3 Ergebnisse

Um die Funktionalität des vorgeschlagenen Entwurfskonzepts am Beispiel des Robotergreifers zu validieren, wird im Weiteren Aufgabenwissen in Form von hierarchisch strukturierten Petri-Netzen (Koordinierungsnetzen) generiert. Bei den beiden beispielhaft mit dem Robotergreifer umgesetzten Aufgabentypen, handelt es sich um rein positionsgeregelte Zeigegesten sowie deutlich komplexere kraft- und positionsgeregelte Greifbewegungen. Die Evaluierung des Konzepts erfolgt in diesem Abschnitt rein simulativ auf der Entwicklungsumgebung, da der reale Robotergreifer derzeit noch nicht über die notwendige sensorische Ausstattung verfügt. Der Robotergreifer und die zu greifenden Objekte werden dazu mit den im Abschnitt 4.2.1 vorgestellten Modellen simuliert.

4.2.3.1 Greifergesten

Gesten sind allgemein zeichenhafte Bewegungen bestimmter Körperteile (v. a. Hand und Kopf) zum Zwecke der nonverbalen Kommunikation [145]. Weltweit bekannte Gesten, die mit der Hand ausgeführt werden, sind beispielsweise die Zeigegeste, das Victoryzeichen, die Faust oder das Winken. Um eine solche Bewegung mit dem vorgestellten petrinetzbasierten Konzept umzusetzen, sind zunächst aufgaben- und teilkomponentenspezifische Elementaraktionen $EA_{i,u}$ zu entwerfen, zu hierarchisch strukturierten Petri-Netzen (Koordinierungsnetze) zusammenzufassen und im Aufgabenwissen der Roboter-Steuerungsarchitektur zu speichern.

Die für die Ausführung einer Gestenbewegung notwendige Öffnungs- bzw. Schließbewegungen des Robotergreifers können mit der Elementaraktion 'Fahren' aus Tabelle 2.1 umgesetzt werden. Deshalb besteht das Koordinierungsnetz der hier beispielhaft vorgestellten Zeigegeste auch nur aus einem diskreten Platz. Zur vollständigen Definition einer Elementaraktion sind nach Abschnitt 2.2.1 Funktionen zur Trajektoriengenerierung und Regelung zu entwerfen. Die in diesem Beispiel implementierte Trajektoriengenerierung generiert rampenförmige Referenztrajektorien für jeden Freiheitsgrad (DOF) des Robotergreifers. Die entsprechenden Sollpositionen der Freiheitsgrade des Robotergreifers sind in Tabelle 4.3 dargestellt. Die Trajektorienart und die Sollpositionen der Freiheitsgrade sind dabei so gewählt, dass der Greifer eine möglichst menschenähnlich aussehende Zeigebewegung ausführt.

Finger i	Sollpositionen der Freiheitsgrade j
Kleiner Finger	$\varphi_{Soll(1,1)}, \varphi_{Soll(1,2)} = 70^\circ$
Ringfinger	$\varphi_{Soll(2,1)}, \varphi_{Soll(2,2)} = 70^\circ$
Mittelfinger	$\varphi_{Soll(3,1)}, \varphi_{Soll(3,2)} = 70^\circ$
Zeigefinger	$\varphi_{Soll(4,1)}, \varphi_{Soll(4,2)} = 0^\circ$
Daumen	$\varphi_{Soll(5,1)}, \varphi_{Soll(5,2)}, \varphi_{Soll(5,3)} = 70^\circ$

Tabelle 4.3: Sollpositionen der Freiheitsgrade für die Zeigegeste

Zur Berechnung der Regeldifferenz wird das Admittanzkonzept nach Gleichung (4.21) verwendet, die Stellgrößen werden nach Abschnitt 4.2.2 in Abhängigkeit von der Betriebsart entweder mit dem Zweipunkt- (Pneumatik) oder dem hybriden diskret-kontinuierlichen Regelungsalgorithmus (Hydraulik) berechnet. Das auf diese Weise generierte Koordinierungsnetz mit unterlagerten Funktionen zur Trajektoriengenerierung und Regelung wird in der Datenbank abgelegt und ist somit Bestandteil des globalen Aufgabenwissens des Robotersystems. Um die Zeigegeste mit dem Robotersystem auszuführen, wird die Aufgabe über das Bediener-Interface (GUI) an das Robotersystem kommandiert. Dazu muss sich das System im Zustand 'Bereit' befinden. Ist die gestellte Aufgabe dem Robotersystem bekannt, d. h. exisitiert Aufgabenwissen in Form von Elementaraktionen, erfolgt die Weiterleitung in den Aufgabenspeicher auf der mittleren Ebene der Steuerungsarchitektur. Bei Verfügbarkeit der notwendigen Hardwareressourcen wird die Ausführung der Aufgabe durch die Ressourcenverwaltung freigegeben. Das entsprechende Koordinierungsnetz wird aus der Datenbank geladen und mit den parametrierten Funktionen zur Trajektoriengenerierung und Regelung ausgeführt.

Im Weiteren wird die hier beispielhaft betrachtete Zeigegeste sowohl mit dem pneumatisch (Szenario 1) als auch mit dem hydraulisch (Szenario 2) angetriebenen Robotergreifer simulativ umgesetzt. Für das Szenario 1 sind als Ergebnis die Verläufe der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel $\varphi_{Soll(i,j)}$ und $\varphi_{(i,j)}$ sowie die Verläufe der Steuerspannungen der Einlass- und der Auslassventile für die 11 Freiheitsgrade des Robotergreifers über der Zeit abgebildet (Bild 4.11). Um die beiden Ventilarten pro Freiheitsgrad in einem Bild darzustellen, erfolgt die Repräsentation geöffneter Einlassventile in Form von positiven Steuerspannungen $x_{V(i,j)} = 1$ und Repräsentation geöffneter Auslassventile in Form von negativen Steuerspannungen $x_{V(i,j)} = -1$ (Bild 4.11 b).

Prinzipiell ermöglicht der pneumatische Betrieb gegenläufige Gelenkbewegungen (hier Strecken der Zeigefingergelenke bei gleichzeitigem Beugen aller anderen Fingergelenke). Da die konstruktiv vorgegebene Gleichgewichtslage der verwendeten Fluidaktoren bei Umgebungsdruck p_0 allerdings $\varphi_{0(i,j)} \approx 10^\circ$ beträgt und bei pneumatischen Betrieb nur gegen den Umgebungsdruck ausgeblasen wird, ist der Regler, trotz Ansteuerung der entsprechenden Auslassventile ($x_{V(4,1)} = -1$ und $x_{V(4,2)} = -1$, Bild 4.11 b) nicht in der Lage, die geforderte Überstreckung der Fingergelenke des Zeigefingers ($\varphi_{Soll(4,1)} = 0$ und $\varphi_{Soll(4,2)} = 0^\circ$) zu realisieren (Bild 4.11 a).

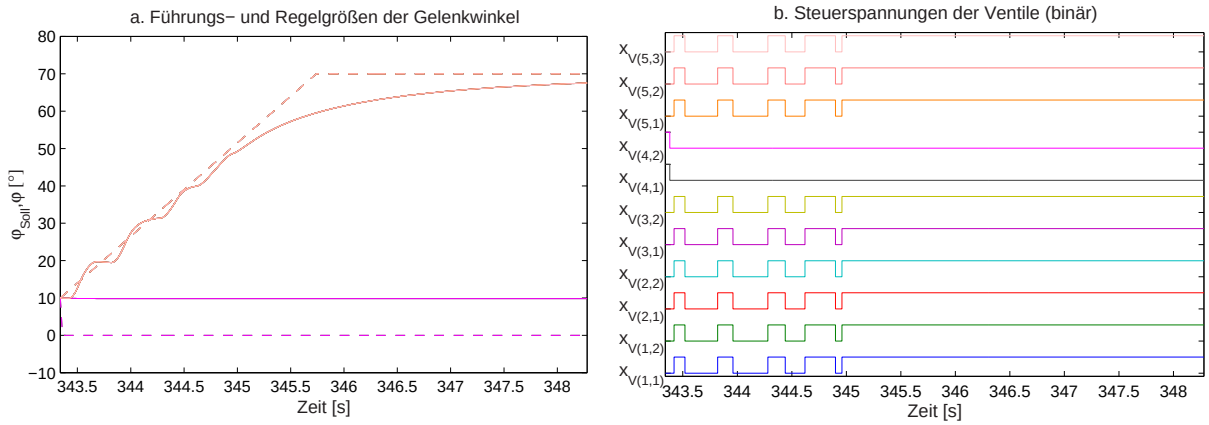


Bild 4.11: Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel (a) sowie binäre Steuerspannungen der Einlass- und Auslassventile (b) über der Zeit (Szenario 1)

Für das Szenario 2 (hydraulischer Betrieb) sind als Ergebnis die Verläufe der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel $\varphi_{Soll(i,j)}$ und $\varphi_{(i,j)}$ sowie die Verläufe der Steuerspannung der Pumpe x_P und der Ventile $x_{V(i,j)}$ für die 11 Freiheitsgrade des Robotergreifers über der Zeit abgebildet (Bild 4.12).

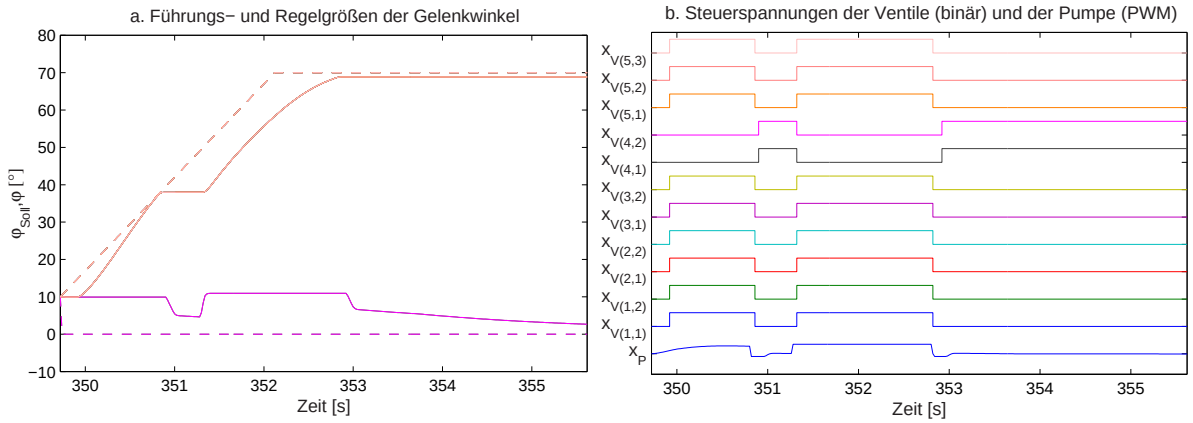


Bild 4.12: Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel (a) sowie Steuerspannungen der Ventile und der Pumpe (PWM) (b) über der Zeit (Szenario 2)

Deutlich zu erkennen sind die bereits in den Abschnitten 4.2.1 und 4.2.2 angesprochenen Probleme aufgrund der Ausführung der hydraulischen Fluidversorgung. Da für die Regelung *mehrerer* Gelenke nur *eine* unabhängige Druckversorgung zur Verfügung steht, kann sich der diskrete Regelungsalgorithmus (Abschnitt 4.2.2) pro Abtastintervall nur für eine Pumpenrichtung und damit für ein entsprechendes pulsweitenmoduliertes Steuersignal x_P entscheiden und ist damit nicht in der Lage, in einem Abtastintervall gegenläufige Bewegungen verschiedener Fingergelenke zu realisieren. Allerdings kann im Gegensatz zum pneumatischen Betrieb die geforderte Überstreckung der Fingergelenke des Zeigefingers erreicht werden (Pumpe erzeugt Unterdruck im Aktor).

Gemäß Bild 4.12 überwiegt zunächst die Summe der negativen Regeldifferenzen (Freiheitsgrade deren Sollposition $\varphi_{Soll(i,j)} = 70°$ beträgt). Der Regler generiert eine positive Steuerspannung für die Pumpe ($x_P > 0$) und öffnet die Ventile der entsprechenden Freiheitsgrade (Bild 4.12 b). Die Ventile der Freiheitsgrade mit positiver Regeldifferenz (Zeigefinger vgl. Tabelle 4.3) bleiben geschlossen ($x_{V(4,1)} = 0$

und $x_{V(4,2)} = 0$). Ab dem Simulationszeitpunkt $t = 350.8$ s ist die Summe aller positiven Regeldifferenzen größer als die Summe der negativen Regeldifferenzen. Der Regler entscheidet folgerichtig, die Pumpe in den Rückwärtsbetrieb umzuschalten und die Ventile dieser Freiheitsgrade zu öffnen. Die Ventile der Freiheitsgrade mit negativer Regeldifferenz werden entsprechend geschlossen (Bild 4.12 b). In diesem Fall wird dadurch die begonnene Schließbewegung unterbrochen und zunächst eine gegenläufige Öffnungsbewegung ausgeführt. Zum Simulationszeitpunkt $t = 353.6$ s ist die Summe der positiven Regeldifferenzen weitgehend abgebaut, so dass wiederum die Summe aller negativen Regeldifferenzen überwiegt. Entsprechend erfolgt ein abermaliges Umschalten von Pumpe und Ventilen. Obwohl schließlich alle Freiheitsgrade die vorgegebenen Sollpositionen erreichen, ist das Resultat der unkoordinierten, in diesem Fall aber nicht zu verhindernden Pumpen- und Ventilumschaltungen, eine nicht menschlich aussehende Zeigegestenbewegung des Robotergreifers.

Eine Möglichkeit diese Problematik zu umgehen, ist die Aufteilung der Gestenbewegung in fingerspezifische Elementaraktionen. Analog zu dem bereits beschriebenen Vorgehen werden jetzt alle Freiheitsgrade, die dieselben Anforderungen an die Pumpe haben, zu einem separaten Koordinierungsnetz, wiederum in Form einer Elementaraktion, zusammengefasst. Anstatt *einer* alle Freiheitsgrade des Robotergreifers steuernden Elementaraktion werden in diesem Beispiel somit drei fingerspezifische Elementaraktionen erzeugt und sequentiell ausgeführt. Erst wenn die vorausgegangene Elementaraktion erfolgreich abgearbeitet (Regelungsziel erreicht) ist, sind die notwendigen Hardwareressourcen verfügbar (Platz A5 frei) und die Ausführung einer weiteren Elementaraktion wird durch die Ressourcenverwaltung freigegeben.

Bild 4.13 zeigt als Ergebnis die Durchführung der Zeigegeste mit dem hydraulisch betriebenen Robotergreifer. Abgebildet sind wiederum die Verläufe der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel $\varphi_{Soll(i,j)}$ und $\varphi_{(i,j)}$ sowie die Verläufe der Steuerspannungen der Pumpe x_P und der Ventile $x_{V(i,j)}$ für die 11 Freiheitsgrade des Robotergreifers über der Zeit.

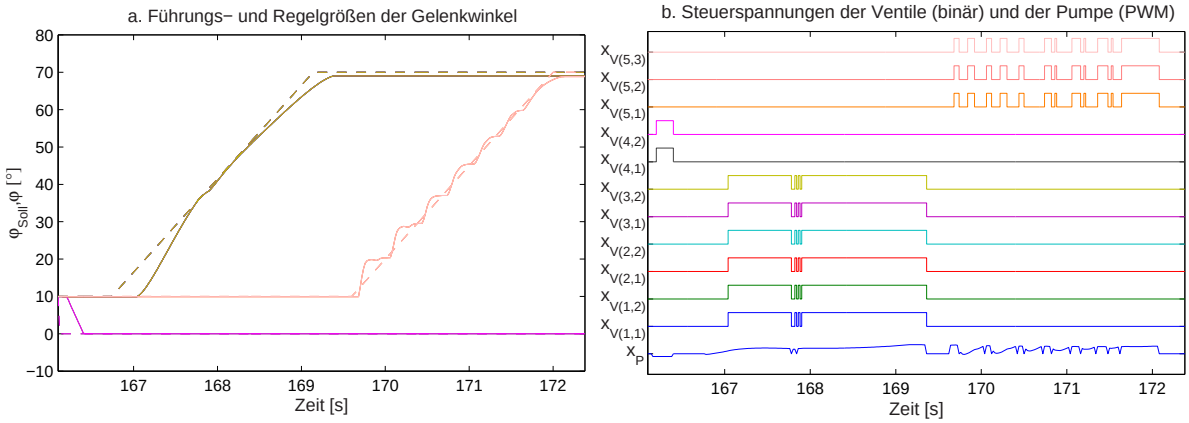


Bild 4.13: Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel (a) sowie Steuerspannungen der Ventile und der Pumpe (PWM) (b) über der Zeit (Szenario 3))

Die erste ausgeführte Elementaraktion hat die Aufgabe, die Freiheitsgrade des Robotergreifers mit positiver Regeldifferenz (Zeigefinger vgl. Tabelle 4.3) zu steuern. Deshalb wird die Pumpe im Rückwärtslauf betrieben und die dazugehörenden Ventile werden geöffnet. Alle anderen Ventile sind geschlossen (Bild 4.13 b). Ist die Regeldifferenz für einen bestimmten Zeitraum innerhalb eines vorgegebenen Sollbereichs (Regelziel), dann wertet das entsprechende Verwaltungsmodul der Steuerungsarchitektur des Robotersystems dies als erfolgreich durchgeführte Elementaraktion und aktiviert die Transition T12 des Statusnetzes. Erst dadurch wird der Platz A5 'Aktiv' des Statusnetzes frei und die Ressourcenverwaltung kann die Ausführung der nächsten Elementaroperation freigeben (vgl. Abschnitt 3.3). Die Ausfüh-

rung der zweiten Elementaraktion führt zu einem Abwinkeln aller Freiheitsgrade von Zeige-, Mittel- und Ringfinger. Nach erfolgreicher Beendigung wird die dritte Elementaraktion ausgeführt. Diese führt zu einem Abwinkeln aller Freiheitsgrade des Daumens. Durch diese mit dem vorgeschlagenen neuen Konzept sehr leicht zu realisierende, zeitliche Koordination ist eine deutlich menschlichere Zeigegestenbewegung mit dem hydraulisch betriebenen Robotergreifer möglich. Statt der hier vorgestellten Umsetzung mit sequentiell ausgeführten, unabhängigen Elementaraktionen, ist auch ein einzelnes allerdings komplexeres Koordinierungsnetz denkbar, welches die vorgestellten fingerspezifischen Elementaraktionen enthält. Nachteilig sind dabei aber eine deutlich höherer Entwurfs- und Implementierungsaufwand vor allem bei den Funktionen zur Trajektoriengenerierung.

4.2.3.2 Greifvorgänge

In diesem Abschnitt erfolgt die Umsetzung und Evaluierung des vorgeschlagenen Gesamtkonzepts anhand von simulierten Greifvorgängen. Da bei Greifbewegungen im Normalfall keine gegenläufigen Bewegungen auftreten, sind beide Antriebsvarianten grundsätzlich in der Lage, die vorgegebenen Bewegungen erfolgreich durchzuführen. Allerdings zeigen Vergleichssimulationen, dass bei der gleichzeitigen Ansteuerung mehrerer Aktoren die hydraulische Variante systembedingt Nachteile aufweist. Aufgrund der höheren regelungstechnischen Herausforderung wird im Weiteren deshalb der hydraulisch angetriebene Robotergreifer als Referenzsystem betrachtet. Die Umsetzung eines Greifvorgangs erfolgt wiederum mit der in Kapitel 3.3 vorgestellten Implementierung der Roboter-Steuerungsarchitektur auf der Entwicklungsplattform (Bild 4.14).

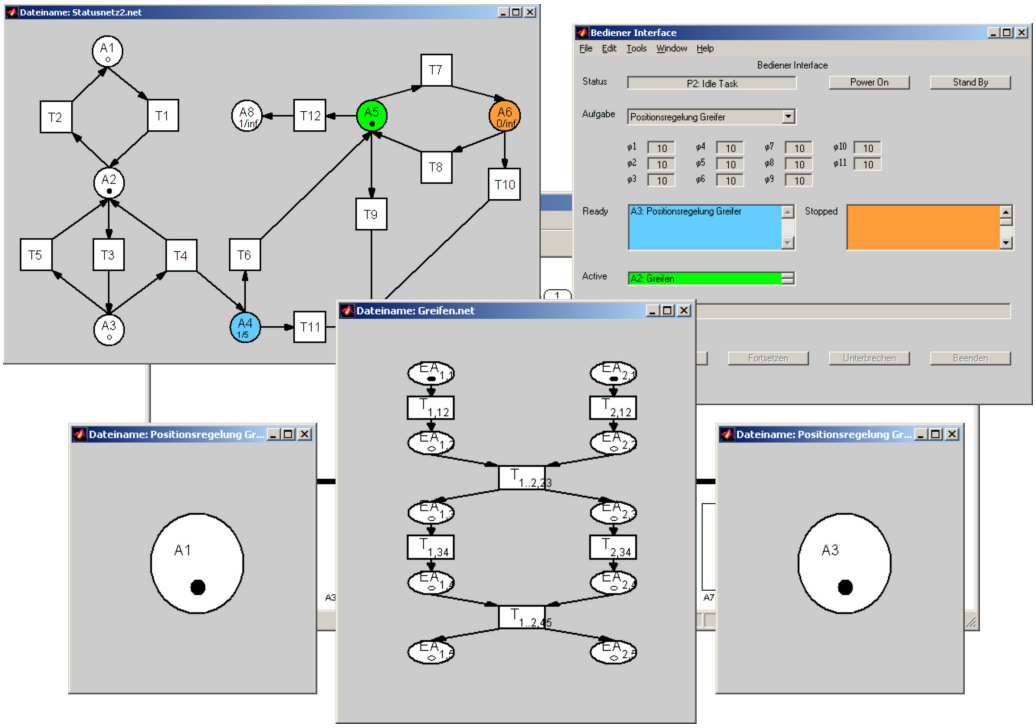


Bild 4.14: Umsetzung eines Präzisionsgriffs auf der Entwicklungsplattform. Das Beispiel zeigt die Realisierung der Roboter-Steuerungsarchitektur nach Kapitel 3.3 (oben links und rechts) sowie die Koordinierungsnetze für die 'Preshape-Pose' (unten links), die 'Greifbewegung' (unten Mitte) und die 'Initial-Pose' (unten rechts)

Vor der Ausführung der eigentlichen 'Greifbewegung' muss der Robotergreifer aus der 'Initial-Pose' in die so genannte 'Preshape-Pose' gebracht werden. Nach der Durchführung der 'Greifbewegung' erfolgt die Rückkehr in die 'Initial-Pose', um bereit für die Ausführung einer neuen Aufgabe zu sein. Für die Preshape- und die Initial-Pose besteht das Koordinierungsnetz aus einem diskreten Platz, beschreibbar durch eine Instanz der Elementaraktion 'Fahren' (Bild 4.14 unten links und rechts). Eine 'Greifbewegung' setzt sich im Gegensatz dazu aus mehreren Elementaraktionen zusammen. So sind für die erfolgreiche Ausführung einer Greifbewegung mindestens die Elementaraktionen 'Fahren', 'Zugreifen' und 'Halten' zu definieren. Weiterhin hat es sich als wirksam erwiesen, die 'Wartefunktion' als zusätzliche Elementaraktion mit in die Greifbewegung zu integrieren, um damit die Fingerbewegungen zu koordinieren und dadurch mögliche Planungsfehler auszugleichen [174]. Somit entsteht ein deutlich komplexeres Koordinierungsnetz (Bild 4.14 unten Mitte).

Die an dieser Stelle umgesetzte 'Greifbewegung' ist ein Präzisionsgriff. Bei diesem Griff erfolgt die Ausführung der Greifbewegung durch den Daumen und den Zeigefinger ($n_G = 2$). Beispiele für die Anwendung von Präzisionsgriffen sind das Zeichnen, das Nähen oder das Essen [241]. Bild 4.14 (unten Mitte) zeigt die Umsetzung des entsprechenden Koordinierungsnetzes auf der Entwicklungsplattform. Zunächst nähern sich beide am Griff beteiligten Finger dem zu greifenden Objekt an $EA_{1..2,1}$. Ergibt sich für einen der beiden Finger vorzeitig Kontakt mit dem zu greifenden Objekt, erfolgt für diesen Finger n_1 ein Übergang $T_{n_1,12}$ zum Platz $EA_{n_1,2}$. Dort wartet dieser Finger, bis der zweite am Griff beteiligte Finger Kontakt hat $T_{n_2,2}$ und eine koordinierte Greifbewegung ausgeführt werden kann (Markenfluss zu den Plätzen $EA_{n_1..n_2,3}$). Analog dazu wartet nach Ende der Greifbewegung ein evtl. vorzeitig bereiter Finger n_1 auf dem Platz $EA_{n_1,4}$ bis der zweite am Griff beteiligte Finger n_2 den Kontakt zum Objekt verloren hat $T_{n_2,34)}$ und die koordinierte Absetzbewegung (Transition $T_{1..2,45}$) beginnt. Mit dem Erreichen der Plätze $EA_{(1..2),5}$ ist die Aktionsfolge 'Greifbewegung' für den Präzisionsgriff erfolgreich abgearbeitet. Tabelle 4.4 erläutert alle zu diesem Netz gehörenden Stellen und Transitionen.

Stelle	Bedeutung
$EA_{1..2,1}$	Annähern an das zu greifende Objekt
$EA_{1..2,2}$	Warten bis alle Finger Kontakt
$EA_{1..2,3}$	Greifbewegung ausführen
$EA_{1..2,4}$	Warten bis alle Finger ohne Kontakt
$EA_{1..2,5}$	Entfernen
Transition	**Bedeutung**
$T_{n_1,12}$	Objektkontakt Finger n_1
$T_{1..2,23}$	Objektkontakt für beide Finger
$T_{n_2,34}$	kein Objektkontakt Finger n_2
$T_{1..2,45}$	kein Objektkontakt für beide Finger

Tabelle 4.4: Plätze und Transitionen der Aktionsfolge 'Greifbewegung' für den Präzisionsgriff

Die Transitionen des Koordinierungsnetzes hängen in diesem Beispiel allein von der derzeitigen Kontaktsituation der Finger des Robotergreifers ab (vgl. Tabelle 4.4). Die Entscheidung, ob Objektkontakt vorliegt, erfolgt dabei sensoriell (mit taktiler Sensorik) oder bei fehlender Sensorik, modellbasiert mit Hilfe von Kontaktzustandsschätzungen [38].

Bild 4.15 zeigt als Ergebnis die Verläufe der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) für die Gelenkwinkel (a) und die Gelenkmomente (c), die Ventil- und Pumpenzustände (b) sowie die Kontaktsituation (d) über der Zeit für die 11 Freiheitsgrade des Robotergreifers.

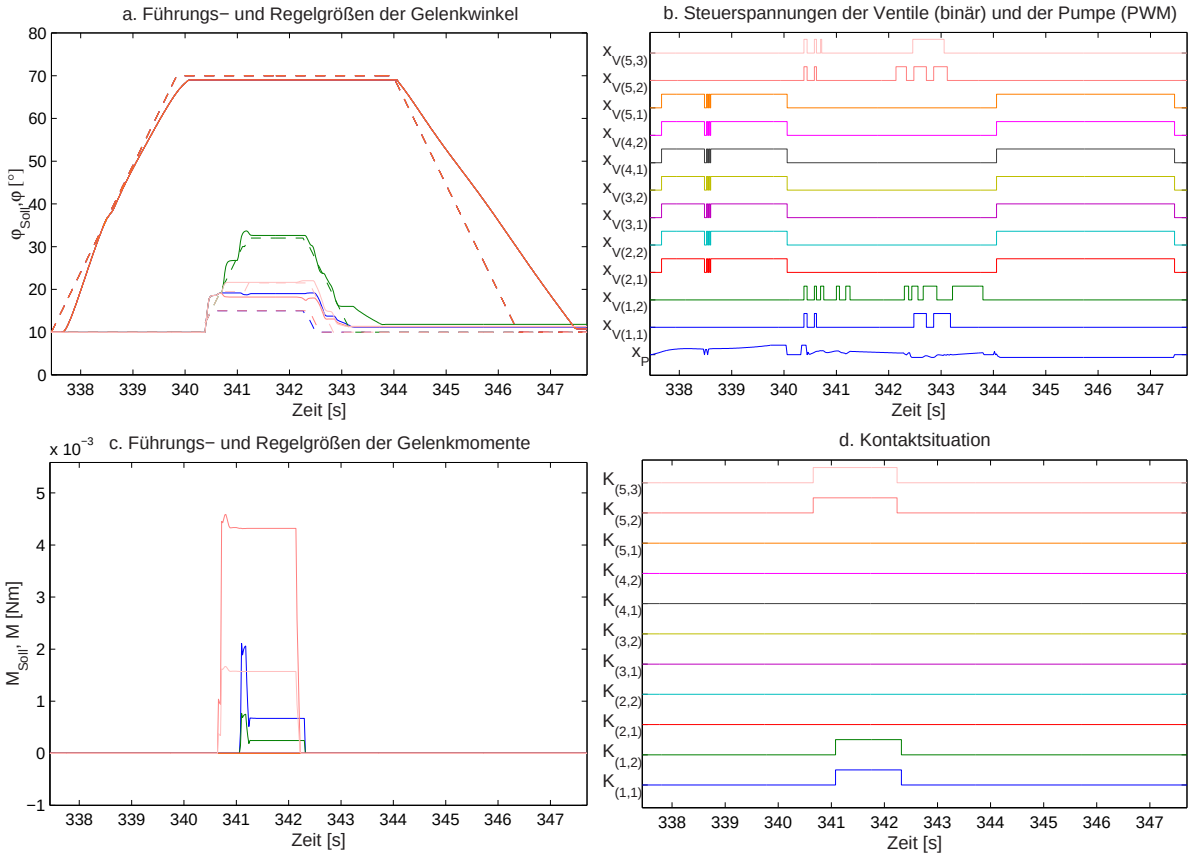


Bild 4.15: Führungs- (gestrichelt) und Regelgrößen (durchgezogen) für Gelenkwinkel (a) und Gelenkmomente (c), Ventil- und Pumpenzustände (b) sowie Kontaktsituation (d) über der Zeit

Bis zum Simulationszeitpunkt $t = 340.3$ s wird die erste Aktion 'Preshape-Pose' ausgeführt. Dazu werden alle Freiheitsgrade des kleinen Fingers, des Ringfingers und des Mittelfingers sowie der erste Freiheitsgrad des Daumens auf $\varphi_{Soll(i,j)} = 70°$ geregelt. Dies bewirkt, dass sich die drei nicht an der Ausführung der Greifbewegung beteiligten Finger schließen und zusätzlich der Daumen in die Oppositionsstellung gebracht wird (Bild 4.15).

Nach dem erfolgreichen Abschluss der 'Preshape-Pose' zum Simulationszeitpunkt $t > 340.3$ s (regelbasiert erkannt durch das entsprechende Verwaltungsmodul der Koordinierungsebene[3], vgl. Abschnitt 3.3) ist der Platz A5 des Statusnetzes nicht mehr belegt und die Ressourcenverwaltung der Steuerungsarchitektur gibt nach Prüfung aller erforderlichen Nebenbedingungen (Robotergreifer in der für den geplanten Griff notwendigen Greifkonfiguration (Preshape-Pose) an der vorausberechneten Greifposition angekommen) die nächste Aktion ('Greifbewegung') frei (vgl. Abschnitt 2.3.3). Dazu wird das entsprechende Koordinierungsnetz mit seiner vorgegebenen initialen Markenbelegung geladen und ausgeführt (Bild 4.14 unten Mitte). Beide Finger nähern sich dem zu greifenden Objekt. Der Regler generiert eine positive Steuerspannung für die Pumpe ($x_P > 0$) und öffnet die Ventile der entsprechenden Freiheitsgrade (Bild 4.15 b). Da in diesem Beispiel die vorausberechnete Objektposition nicht exakt mit der realen Objektposition übereinstimmt, ergibt sich für den Daumen vorzeitig ($t = 340.8$ s) Kontakt mit dem zu greifenden Objekt (Bild 4.15 d). Dieser Finger wartet, bis auch der Zeigefinger Objektkontakt erreicht hat ($t = 341.0$ s) und der koordinierte Greifvorgang beginnt. Dazu werden durch die über den Steifig-

[3] alternativ über einen zusätzlichen Platz im Koordinierungsnetz realisierbar

keitsparameter K und den Dämpfungsparameter D eingestellte Admittanz Gelenkmomente erzeugt, um so die Greifkraft auf das Objekt zu übertragen (Bild 4.15 c). Beim Loslassen verliert der Zeigefinger als erster ($t = 342.2$ s) den Kontakt zum Objekt (Bild 4.15 d) und wartet entsprechend, bis auch der Daumen den Kontakt zum Objekt verliert und die koordinierte Loslassbewegung beginnen kann ($t = 342.3$ s). Dazu generiert der Regler eine negative Steuerspannung für die Pumpe ($x_P < 0$) und öffnet die Ventile der entsprechenden Freiheitsgrade (Bild 4.15 b).

Zum Simulationszeitpunkt $t > 344.1$ s ist die 'Greifbewegung' erfolgreich beendet und somit der Platz A5 des Statusnetzes wiederum frei. Die Ressourcenverwaltung führt nach Prüfung aller erforderlichen Nebenbedingungen die nächste Aktion ('Initial-Pose') aus. Dazu werden alle Freiheitsgrade des Robotergreifers in ihre Gleichgewichtslage bei $\varphi_{Soll(i,j)} \approx 10^\circ$ zurück geregelt (Bild 4.15 a). Nach dem Erreichen der Gleichgewichtslage ist die Initial-Pose erfolgreich ausgeführt und die Transition T12 im Statusnetz wird von der Steuerungsarchitektur aktiviert. Damit ist zum Simulationszeitpunkt $t > 347.4$ s der komplette Greifvorgang erfolgreich abgeschlossen und der Robotergreifer ist bereit für die Ausführung einer neuen Aufgabe.

4.2.4 Bewertung

Zusammenfassend ergibt sich, dass sowohl die Sensorausstattung als auch die Antriebsart des fluidisch angetriebenen Robotergreifers großen Einfluss auf die Anwendbarkeit von Regelungskonzepten und damit die Greiferperformance haben. Zwar ergeben sich durch die Flexibilität der Fluidaktoren (passive Nachgiebigkeit, vgl. Abschnitt 4.2.2) Vorteile in bestimmten Greifsituationen, allerdings ist für ein sicheres Greifen vor allem in unbekannten Alltagssituationen Winkel- und taktile Sensorik unerlässlich (Drucksensorik optional). Beim Vergleich von pneumatischem und hydraulischem Antrieb ergeben sich bei der Hydraulik Vorteile im Hinblick auf die Mobilität des Robotergreifers, da kein externer Kompressor zur Druckerzeugung notwendig ist. Nachteilig ist dagegen, dass zur Regelung *aller* unabhängigen Freiheitsgrade des Robotergreifers nur *eine* Druckversorgung (Pumpe) zur Verfügung steht. Dies führt wiederum zu Restriktionen bei der Umsetzung von Regelungskonzepten (Abschnitt 4.2.2).

Wie in den vorigen Abschnitten gezeigt, hat neben dem System auch die Entwurfs- und Implementierungsflexibiliät des hier vorgestellten neuen Konzepts zur Generierung von Aufgabenwissen mit Petri-Netzen einen entscheidenden Einfluss auf die Gesamtperformance des Robotergreifers. So können beispielsweise durch geschicktes Aufteilen einer komplexen Gestenbewegung in kleinere fingerspezifische Teilbewegungen (Elementaraktionen) Restriktionen der Mechanik umgangen werden (Beispiel: Gestenbewegung).

4.3 Anthropomorphe Halseinheit

4.3.1 Modellbildung

In diesem Abschnitt erfolgt die mathematische Modellierung der einzelnen Komponenten der Roboterhalseinheit. Dazu werden die Funktionalbeziehungen der durch Konstruktion vorgegebenen Kinematik und der Teilkomponenten des Antriebs (Motor, Getriebe und Zahnriemen) ermittelt und zu einem dynamischen Gesamtsystem zusammengeführt (Dynamik). Die mathematische Modellierung erfolgt dabei wiederum (vgl. Abschnitt 4.2.1) durch Ableiten der Systemgleichungen aus physikalischen Grundgesetzen (theoretische Modellbildung [90]) und Vergleich mit Messungen am realen System (experimentelle Modellbildung) [130, 226, 267].

4.3.1.1 Kinematik

Beim Menschen werden die Halsbewegungen über jeweils drei Freiheitsgrade in sieben Halswirbeln realisiert. Damit ergeben sich insgesamt 21 Freiheitsgrade [231]. Wünschenswert ist eine Bewegungsfreiheit der Roboterhalseinheit vergleichbar der des Menschen (u. A. bei der Mensch-Roboter-Kooperation). Dies ist technisch derzeit jedoch nicht realisierbar. Deshalb werden die Halsbewegungen auf vier Freiheitsgrade reduziert (Bild 4.16):

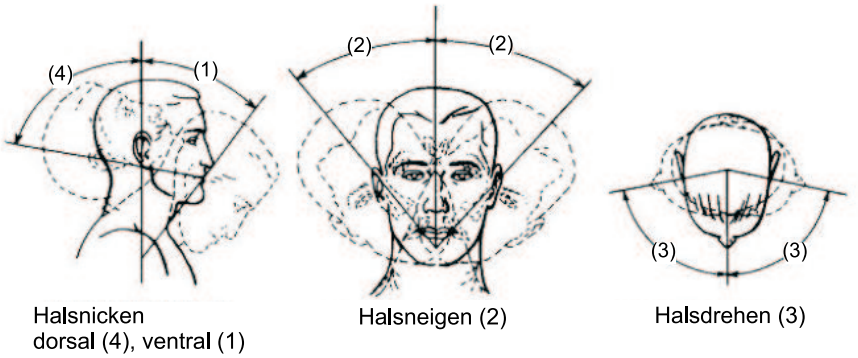


Bild 4.16: Bewegungen des menschlichen Halses [231]

- unteres Nicken (Bild 4.16 links, ventral (1)),
- Halsneigen (Bild 4.16 Mitte (2)),
- Halsdrehen (Bild 4.16 rechts (3)) sowie
- oberes Nicken (Bild 4.16 links, dorsal (4)).

Bild 4.17 zeigt die schematische Darstellung des sich aus diesen Anforderungen und Restriktionen ergebenden kinematischen Grundkonzepts der Roboterhalseinheit.

In Tabelle 4.5 sind die Winkelbeweglichkeiten der Roboterhalseinheit im Vergleich mit denen des menschlichen Halses sowie die konstruktiven Grenzen dargestellt. Die Einhaltung anthropomorpher Bewegungsabläufe, wie beispielsweise die Begrenzung der Halsdrehung, muss sofern nicht konstruktiv vorgegeben, über die Regelung (Abschnitt 4.3.2) realisiert werden.

Winkelbeweglichkeiten			
	menschlicher Hals	Roboterhalseinheit	konstruktive Grenzen
unteres Nicken (1)	$60°$	$\pm 30°$	$\pm 45°$
Halsneigen (2)	$\pm 41°$	$\pm 30°$	$\pm 45°$
Halsdrehen (3)	$\pm 79°$	$\pm 90°$	∞
oberes Nicken (4)	$61°$	$\pm 30°$	$\pm 90°$

Tabelle 4.5: Vergleich zwischen menschlicher Halsbeweglichkeit und Roboterbeweglichkeit sowie Darstellung der konstruktiven Grenzen [231]

Die Ansteuerung der Gelenkachsen erfolgt über Antriebseinheiten, bestehend aus Motor und Getriebe. Für die Bewegung des unteren Nickens (1) und des Neigens (2) sind zusätzlich Zahnriemen vorhanden (in Bild 4.17 nicht dargestellt). Im Folgenden werden die Funktionalbeziehungen dieser Teilsysteme mathematisch ermittelt und zu einem dynamischen Gesamtsystem zusammengeführt.

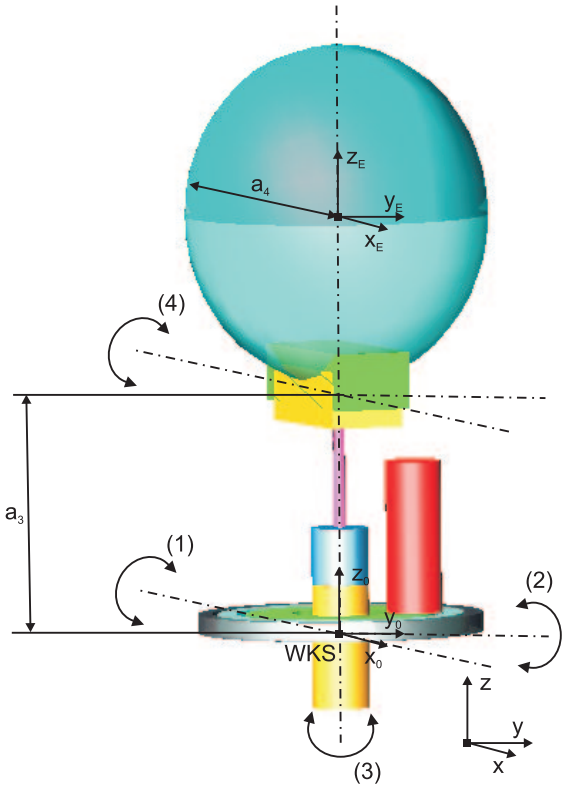


Bild 4.17: Schematische Darstellung des kinematischen Grundkonzepts der Roboterhalseinheit [231]

4.3.1.2 Teilsysteme des Antriebs

Elektromotor Die verwendeten Elektromotoren haben die Aufgabe, Energie auf die Bewegungsachsen zu übertragen, um damit die Gelenke der Halseinheit direkt oder indirekt über Getriebeeinheiten anzutreiben. Die Motoren wandeln dazu elektrische Energie (Strom, Spannung) in mechanische Energie (Drehzahl, Drehmoment). Für alle Freiheitsgrade der Roboterhalseinheit wird ein Faulhaber Gleichstrommotor der Serie 2342 024 CR mit 24V Nennspannung verwendet. Die Parameter des Motors sind in Tabelle C.4 zusammengefasst. Für die Bestimmung des dynamischen Übertragungsverhaltens müssen die Funktionalbeziehungen der zeitveränderlichen Größen aufgestellt werden. Ausgangspunkt hierfür ist das in Abbildung 4.18 dargestellte Schaltbild eines Elektromotors mit Last.

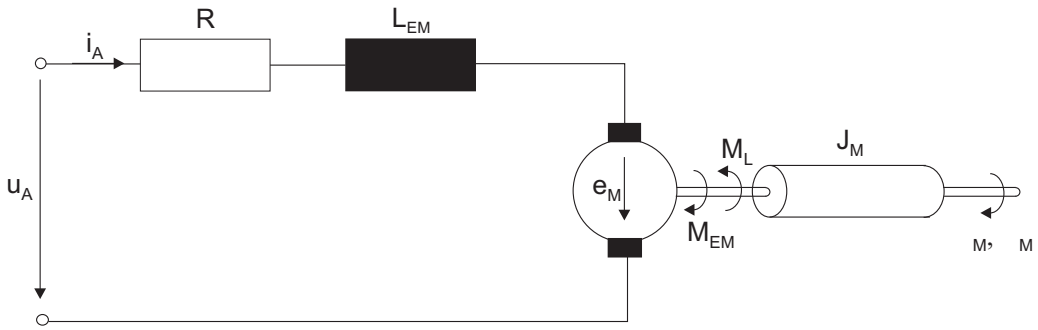


Bild 4.18: Ersatzschaltbild des Motors mit Last [90]

Der Gleichstrommotor wird durch die Ankerspannung u_A angesteuert und erzeugt das Motormoment M_{EM}. Dem Motormoment wirkt ein Lastmoment M_L entgegen. Ausgangsgrößen sind die Winkelgeschwindigkeit ω_M und die Achsposition φ_M des Motors.

Bei angelegter Ankerspannung u_A am Motor gilt:

$$u_A(t) = i_A(t)R + L_{EM}\frac{di_A(t)}{dt} + e_M. \tag{4.39}$$

wobei R den Anschlusswiderstand, L_{EM} die Anschlussinduktivität, i_A den Ankerstrom und e_M die induzierte Spannung darstellen. Ausgehend von konstanter Felderregung ist die induzierte Spannung e_M der Winkelgeschwindigkeit ω_M proportional:

$$e_M(t) = k_M\omega_M(t). \tag{4.40}$$

Das Motormoment M_{EM} ist dem Ankerstrom i_A proportional, wobei k_M die Drehmomentkonstante darstellt:

$$M_{EM}(t) = k_M i_A(t). \tag{4.41}$$

Das auf die Motorwelle reduzierte Lastmoment $M_{L,Red}$ verringert das Motormoment, so dass für das antreibende Moment gilt:

$$M(t) = M_{EM}(t) - M_{L,Red}(t). \tag{4.42}$$

Die Winkelbeschleunigung $\frac{\omega_M}{dt}$ wird über das auf die Motorwelle reduzierte Massenträgheitsmoment J_{red} zu

$$\frac{d\omega_M(t)}{dt} = \frac{M(t)}{J_{Red}(t)}. \tag{4.43}$$

Die zeitabhängigen Größen $M_{L,Red}(t)$ und $J_{Red}(t)$ werden im Abschnitt 4.3.1.2 hergeleitet. Die Änderung der Achsposition φ_M über der Zeit ergibt die Winkelgeschwindigkeit des Motors ω_M:

$$\frac{d\varphi_M(t)}{dt} = \omega_M(t). \tag{4.44}$$

Die Gleichungen (4.39)-(4.44) stellen das mathematische Modell des Gleichstrommotors mit Last dar.

Harmonic Drive Getriebe Die Aufgabe des Getriebes ist die Übertragung und Wandlung von Drehmoment und Drehzahl. Wesentliche Merkmale des Harmonic Drive Getriebes sind das präzise und spielfreie Übertragungsverhalten, ein hoher Wirkungsgrad und eine hohe Torsionssteifigkeit. Ein Harmonic Drive Getriebe besteht prinzipiell aus drei Teilen. Der Circular Spline, der Flexspline und dem Wave Generator [208].

- Circular Spline: Massiver zylindrischer Stahlring, der eine innen liegende Verzahnung besitzt,
- Flexspline: Zylindrische, verformbare Stahlbüchse mit außen liegender Verzahnung,
- Wave Generator: Elliptische Stahlscheibe mit zentrischer Nabe und einem elliptisch verformbaren Spezialkugellager.

Für jeden Freiheitsgrad der Halseinheit ist ein Harmonic Drive Getriebe HFUC-8-100-2A mit einer Untersetzung von $i_G = 100$ vorgesehen.

Trotz hoher Torsionssteifigkeit sind im realen System Elastizitäten vorhanden, deren mathematische Modellierung allerdings extrem kompliziert ist. Aus diesem Grund wurde in [19] eine Abschätzung des durch die Vernachlässigung dieser Elastizitäten entstehenden Fehlers durchgeführt. Demnach treten die

maximalen Abweichungen beim unterem Nicken (1) und beim Neigen (2) auf und entsprechen maximal $0.25°$. Aufgrund der geringen Auswirkung der Elastizitäten auf die Genauigkeit werden diese für das hier verwendete mathematische Modell vernachlässigt (”starres Getriebe”). Für diesen Fall folgt ein direktes Verhältnis des Motorwinkels zum Lastwinkel. Der Proportionalitätsfaktor ist das Untersetzungsverhältnis i_G. Mit $\varphi_{GE} = \varphi_M$ (φ_{GE}=Achsposition des Getriebeeingangs, φ_M=Achsposition des Motors, vgl. Abschnitt 4.3.1.2) und $\varphi_{GA} = \varphi_L$ (φ_{GA}=Achsposition des Getriebeausgangs, φ_L=Achsposition der Last) folgt:

$$i_1 = i_4 = i_G = \frac{\varphi_M}{\varphi_L}. \tag{4.45}$$

Gleichung (4.45) gilt für das Drehen und das obere Nicken mit $i_G = 100$. Für die anderen beiden Freiheitsgrade muss zusätzlich der Zahnriementrieb berücksichtigt werden.

Zahnriementrieb Der Zahnriementrieb ist ein formschlüssiges Zugmittelgetriebe. Zugmittelgetriebe werden zur Wandlung von Drehzahlen und Drehmomenten eingesetzt, wobei die Wandlung zwischen zwei oder mehreren, nichtkoaxialen Wellen stattfindet. Dabei sind größere Achsabstände möglich. Als Zugmittel findet der Zahnriemen Verwendung, der die Riemenscheibe von An- und Abtriebswelle umschlingt. Die Aufgabe des Zahnriemens ist, die auftretenden Umfangskräfte formschlüssig und ohne Schlupf zu übertragen [109]. Dabei sind auf der Innenseite des Riemens Zähne aus Neopren oder Polyurethan angeordnet, die in ein Zahnrad eingreifen. Vorteile des Zahnriementriebes sind geringer Platzbedarf sowie Schlupf- und Wartungsfreiheit. Typische Einsatzorte von Zahnriementrieben sind der Nockenwellenantrieb in Kolbenmotoren oder der Antrieb von Positioniersystemen in der Robotik.

Für die Bewegung des seitlichen Neigens und des unteren Nickens werden identische AT5-Leistungsprofil-Zahnriemen der Fa. Mulco eingesetzt. Mit antriebs- und abtriebsseitigen Zähnezahlen von $z_1 = 15$ und $z_2 = 32$ ergibt sich ein Übersetzungsverhältnis von $i_Z = \frac{z_2}{z_1} = 2.1\overline{33}$. Die Gesamtübersetzung für das seitliche Neigen und das untere Nicken ergibt sich durch Multiplikation der Übersetzungsverhältnisse des Harmonic Drive Getriebes und des Zahnriementriebes:

$$i_2 = i_3 = i_G i_Z = 213,\overline{33}. \tag{4.46}$$

4.3.1.3 Dynamik

Die Bewegung eines n-achsigen Roboters ist allgemein durch Gleichung (4.5) beschrieben. Zur Herleitung der Bewegungsgleichung für ein spezielles System kann zum Beispiel das rekursive Verfahren von Newton-Euler, der Lagrange-Formalismus oder das Prinzip von d’Alembert verwendet werden.

Für die Roboterhalseinheit wird das Prinzip von d’Alembert benutzt. Dabei wird das Gesamtsystem in Teilsysteme (Motor, Getriebe, Zahnriemen, Last) unterteilt und an den Schnittstellen werden die Schnittreaktionen (Schnittkräfte bzw. -momente) eingetragen. Jedes Teilsystem bildet dann für sich ein Gleichgewichtssystem. Die dynamischen Gleichungen werden aus der mechanischen Bewegung des Motorankers mit angekoppelter Last erhalten (Gleichung (4.43)). Diese Gleichung genügt dem 2. Newtonschen Axiom und stellt die Basis für das Gesamtsystem dar, in welches sämtliche Beziehungen einfließen.

Die Last-Massenträgheitsmomente ergeben sich durch Anwendung des Steinerschen Satzes und berechnen sich mit den kartesischen Koordinaten (x_j, y_j, z_j) des Kopf-Schwerpunkts auf dem j-ten Koordinatensystem für die vier Bewegungsfreiheitsgrade $j = [1 \dots 4]$ wie folgt [19]:

$$\mathbf{J_L}(\mathbf{t}) = \begin{pmatrix} J_{L,1} \\ J_{L,2} \\ J_{L,3} \\ J_{L,4} \end{pmatrix} = \begin{pmatrix} J_S \\ J_S \\ J_S \\ J_S \end{pmatrix} + m_K \begin{pmatrix} x_0^2 + y_0^2 \\ x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ x_3^2 + y_3^2 \end{pmatrix}. \tag{4.47}$$

Die Lastmomente $M_{L,j}$ resultieren aus der Gravitation, die senkrecht, entlang der negativen z-Achse, auf das Weltkoordinatensystem wirkt. Auftretende Zentripetal- und Corioliskräfte werden aufgrund anthropomorpher Geschwindigkeiten vernachlässigt. Die auf die mitbewegten Motorwellen auftretenden Lastmomente berechnen sich wiederum mit Hilfe der kartesischen Koordinaten (x_j, y_j, z_j) des Kopf-Schwerpunkts zu:

$$\mathbf{M_L}(\mathbf{t}) = \begin{pmatrix} M_{L,1} \\ M_{L,2} \\ M_{L,3} \\ M_{L,4} \end{pmatrix} = -m_K g \begin{pmatrix} x_0 \\ y_1 \\ 0 \\ x_3 \end{pmatrix}. \tag{4.48}$$

wobei das negative Vorzeichen aus Gleichung (4.42) stammt, da die Last aus der Null-Lage mit der Bewegung wirkt (d. h. additiv zum Motormoment).

Um die fehlenden Parameter $M_{L,j,Red}$ und $J_{Red,j}$ für die vier Freiheitsgrade zu berechnen, wird im Folgenden der Einfluss des Getriebes auf das dynamische Verhalten des Gesamtsystems untersucht. Zur Reduktion des Lastmomentes wird eine Leistungsbilanz verwendet:

$$P = \frac{dE_{kin}}{dt} = M_{L,Red}\,\dot{\varphi}_M = M_L\,\dot{\varphi}_{GA}. \tag{4.49}$$

Das reduzierte Lastmoment $M_{L,j,Red}$ auf die Motorwelle lautet damit:

$$M_{L,j,red} = M_{L,j}\frac{\dot{\varphi}_{GA}}{\dot{\varphi}_M} = M_{L,j}\frac{1}{i_j}. \tag{4.50}$$

Eine analoge Betrachtung zur Reduktion der Massenträgheitsmomente lastseitig mit

$$E_{kin} = \frac{1}{2}J_{LS,GE}\,\dot{\varphi}_M^2 = \frac{1}{2}J_{LS}\,\dot{\varphi}_{GA}^2 \tag{4.51}$$

ergibt

$$J_{LS,GE} = J_{LS}\frac{\dot{\varphi}_{GA}^2}{\dot{\varphi}_M^2} = J_{LS}\frac{1}{i_j^2} \tag{4.52}$$

wobei

$$J_{LS} = J_L + J_{GA} \tag{4.53}$$

gilt [110].

Das auf die Motorwelle reduzierte Massenträgheitsmoment, das der Beschleunigung der Motorwelle entgegenwirkt, lautet schließlich:

$$J_{Red,j}(t) = J_M + J_{GE} + (J_{L,j}(t) + J_{GA})\frac{1}{i_j^2}. \tag{4.54}$$

Werden die Gleichungen (4.41) und (4.50) in die Gleichungen (4.42) und (4.54) eingesetzt, sind alle Parameter des Dynamik-Modells bestimmt und es ergibt sich:

$$\frac{d\omega_{M,j}(t)}{dt} = \frac{M_j(t)}{J_{Red,j}(t)}. \tag{4.55}$$

Das mathematische Modell des Gesamtsystems wird im Weiteren simulativ und experimentell evaluiert. Die Bilder 4.19-4.22 zeigen den Verlauf der pulsweitenmodulierten Steuerspannung x_{PWM} der Elektromotoren sowie den sich ergebenden Gelenkwinkelverlauf für Messung (jeweils schwarz durchgezogen)

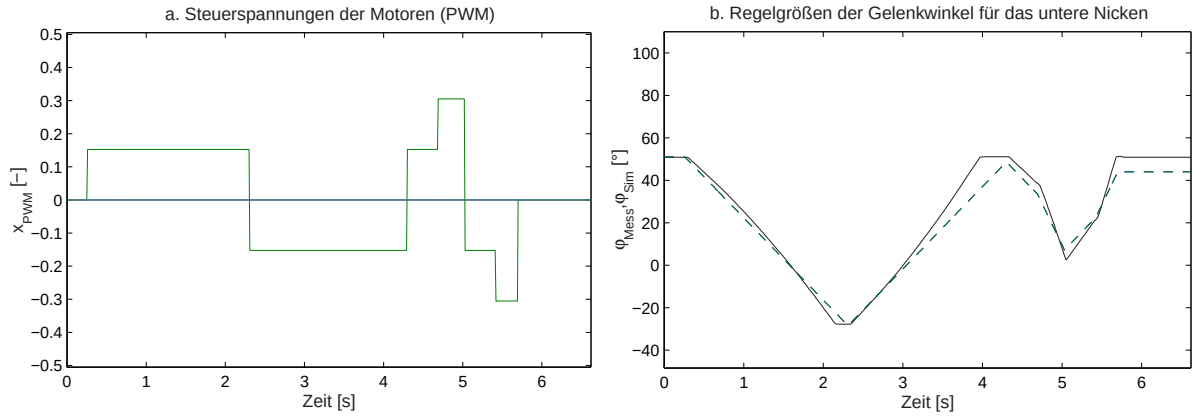


Bild 4.19: Steuerspannungen der Motoren (a) sowie Vergleich der Gelenkwinkelverläufe von Messung (durchgezogen) und Simulation (gestrichelt) (b) über der Zeit für das untere Nicken

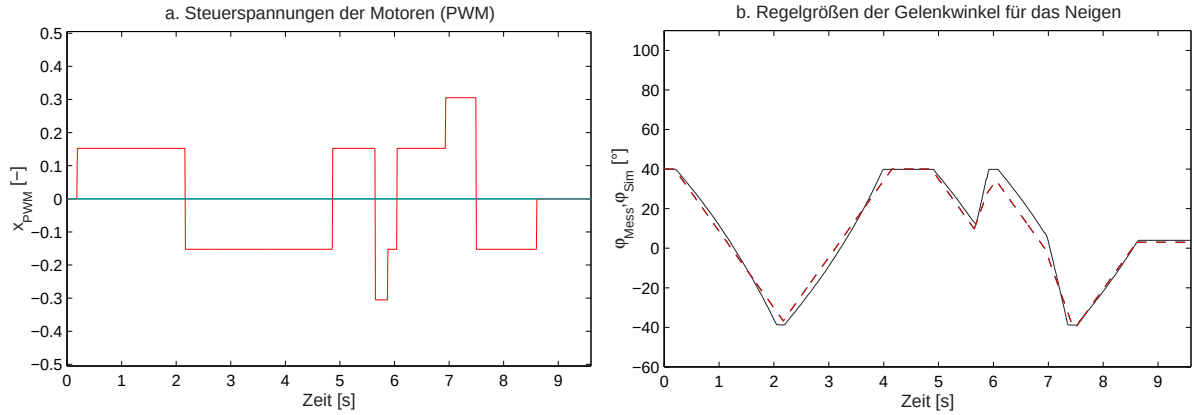


Bild 4.20: Steuerspannungen der Motoren (a) sowie Vergleich der Gelenkwinkelverläufe von Messung (durchgezogen) und Simulation (gestrichelt) (b) über der Zeit für das Neigen

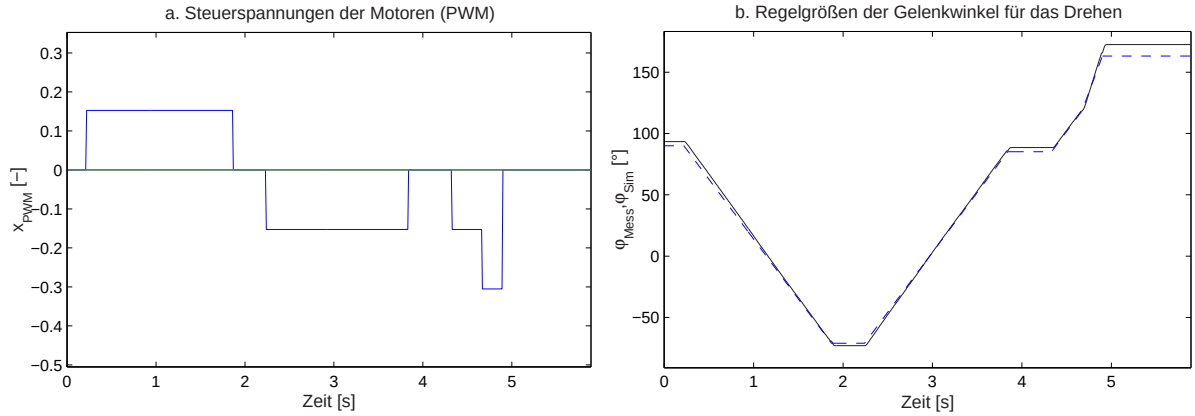


Bild 4.21: Steuerspannungen der Motoren (a) sowie Vergleich der Gelenkwinkelverläufe von Messung (durchgezogen) und Simulation (gestrichelt) (b) über der Zeit für das Drehen

und Simulation (jeweils gestrichelt) für die unabhängigen Freiheitsgrad der Roboterhalseinheit (unteres Nicken (1), Neigen (2), Drehen (3) und oberes Nicken (4)) über der Zeit.

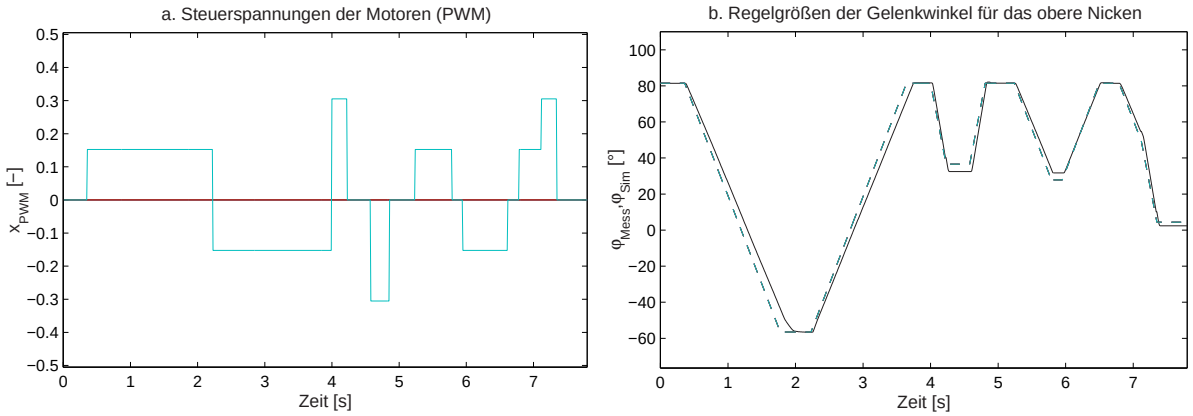


Bild 4.22: Steuerspannungen der Motoren (a) sowie Vergleich der Gelenkwinkelverläufe von Messung (durchgezogen) und Simulation (gestrichelt) (b) über der Zeit für das obere Nicken

Wie in den Bildern 4.19-4.22 zu erkennen, bildet das entwickelte Modell der Roboterhalseinheit für alle hier dargestellten Bewegungsverläufe das reale Systemverhalten ausreichend genau nach. Die hierfür notwendigen Parameteradaptionen beschränkten sich auf die Anpassung der Kopfmasse (Kopfmasse $m_K = 0$ kg bei der realen Halseinheit, da zum Zeitpunkt der Messung noch kein Stereo-Kamerasystem montiert war) sowie die gelenkspezifischen Einstellungen von Bewegungs-Anschlägen und Reibungsparametern. Eine weitere Verbesserung des Simulationsmodells kann durch den Einsatz von numerischen Optimierungsmethoden erreicht werden, ist aber auf Grund der dargestellten Modellgüte nicht notwendig.

4.3.1.4 Kamerasystem

Um das neue Konzept zur Generierung von petrinetzbasiertem Aufgabenwissen simulativ in komplexeren Szenarien mit der Roboterhalseinheit evaluieren zu können, wird neben der Kinematik, der Dynamik und den Teilsystemen des Antriebs zusätzlich ein Stereo-Kamerasystem modelliert und in das bestehende Modell der Roboterhalseinheit integriert. Hierzu wird die frei erhältliche Matlab Epipolar Geometry Toolbox (EGT) verwendet. Mit dieser Toolbox ist es möglich, Multi-Kamera Szenarios zu entwerfen und zu visualisieren. Die Funktionalität umfasst dazu u. a. ein Lochkamera- und ein katadioptrisches Kameramodell [182]. Das in dieser Arbeit verwendete Lochkamera-Modell wird im Weiteren kurz vorgestellt (Bild 4.23).

Gemäß Bild 4.23 beschreibt das Modell den Zusammenhang zwischen einem in homogenen Koordinaten dargestellten Punkt $\mathbf{X}_0 = [x \quad y \quad z \quad 1]^T$ im Weltkoordinatensystem (Ursprung O_0) und seiner Projektion $\mathbf{m} = [u \quad v \quad 1]^T$ auf der Bildebene:

$$\mathbf{m} = \mathbf{K}_K \, {}^0\mathbf{T}_K \, \mathbf{X}_0 \tag{4.56}$$

wobei die $\mathbb{R}^{3\times3}$ Matrix $\mathbf{K}_K$ der internen Kameraparameter durch

$$\mathbf{K}_K = \begin{pmatrix} k_u f & \gamma & u_0 \\ 0 & k_u f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

gegeben ist. Die Koordinaten (u_0, v_0) repräsentieren den Schnittpunkt zwischen der optischen Achse z_K und der Bildebene, k_u und k_v geben die Auflösung in u- und v-Richtung an, f ist die Brennweite und γ bestimmt den Orthogonalitätsfaktor der Bildachsen (u, v).

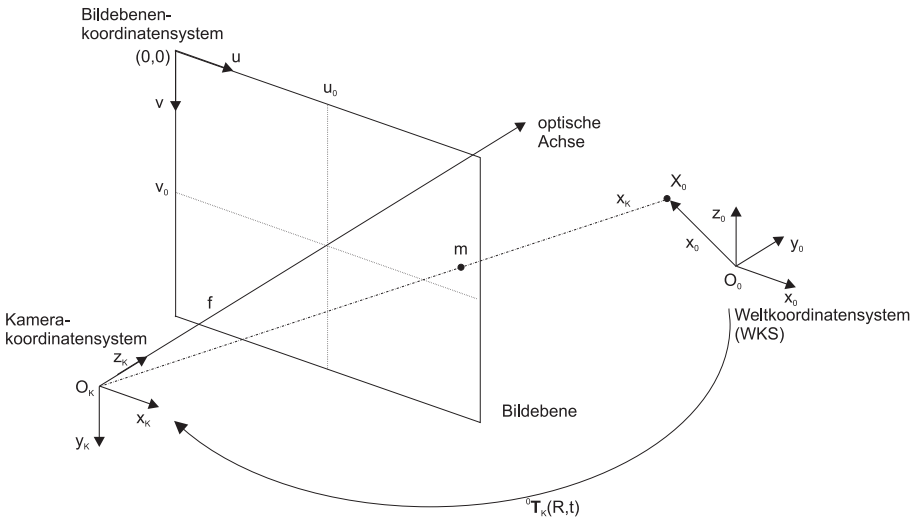


Bild 4.23: Modell der Lochkamera nach [182]

${}^0\mathbf{T}_K(\mathbf{R},\mathbf{t}) \in \mathbb{R}^{3\times 4}$ ist die in homogenen Koordinaten dargestellte Matrix der externen Kameraparameter und beschreibt die Rotation $\mathbf{R}$ und Translation $\mathbf{t}$ zwischen Welt- und Kamerakoordinatensystem. Wird das Modell der Kamera mit dem Modell der Halseinheit aus den vorigen Abschnitten kombiniert, dann wird die Transformationsmatrix ${}^0\mathbf{T}_K$ durch die Vorwärtskinematik der Halseinheit (vgl. Anhang B) bestimmt.

4.3.2 Modellbasierte Regelung

Auf Basis der in den vorigen Abschnitten vorgestellten mathematischen Modellen werden in diesem Abschnitt Algorithmen zur Positionsregelung (Abschnitt 4.3.2.1) bzw. zur bildbasierten Positionsregelung (Abschnitt 4.3.2.2) der Freiheitsgrade der Roboterhalseinheit vorgestellt. Ziel ist es dabei, die Elektromotoren derart anzusteuern, dass trotz Störgrößen wie wechselnder Lasten und/oder Kopplungen der Gelenke ein hinreichend genaues Bewegungsverhalten garantiert werden kann.

Prinzipiell kann bei Roboterregelungen eine Unterteilung nach der Art der Führungsgröße erfolgen. So wird zwischen Regelungskonzepten im Gelenkwinkelraum und im kartesischen Weltkoordinatensystem unterschieden (Abschnitt 1.2.2). Die Umsetzung der Positionsregelung (Abschnitt 4.3.2.1) ist im Gelenkwinkelraum realisiert, während bei der bildbasierten Positionsregelung (Abschnitt 4.3.2.2) die Führungsgrößenvorgabe im kartesischen Weltkoordinaten erfolgt.

4.3.2.1 Positionsregelung

Die Positionsregelung im Gelenkwinkelraum beinhaltet für jedes Gelenk der Roboterhalseinheit einen separaten Eingrößenregler. Diese Regler sind allerdings durch die Trägheit der Mechanik und den auftretenden Lastmomenten miteinander gekoppelt. Bild 4.24 zeigt die sich ergebende Struktur von Regler und Regelstrecke (Teilsysteme des Antriebs und mechanisches System) für die elektrisch angetriebene Roboterhalseinheit.

Eingangsgrößen des Reglers sind die Führungsgrößenvorgaben $\boldsymbol{\varphi}_{Soll}$, die Regelgrößen $\boldsymbol{\varphi}$ und deren Ableitung $\boldsymbol{d\varphi}$ jedes Freiheitsgrades der Roboterhalseinheit. Ausgangsgrößen sind die Stellgrößen des Regelungsalgorithmus in Form der pulsweitenmodulierten Spannungen der Elektromotoren $\mathbf{x}_{PWM}$.

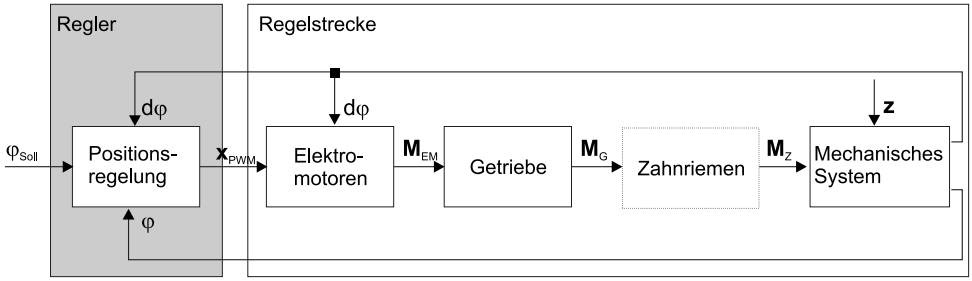


Bild 4.24: Schematische Darstellung von Regler und Regelstrecke (Teilsysteme des Antriebs und mechanisches System) für die elektrisch angetriebene Roboterhalseinheit

Für die erfolgreiche Positionsregelung der Gelenke können prinzipiell verschiedene Regelungskonzepte angewendet werden. Klassische Konzepte sind die Realisierung mit linearen Reglern (z. B. P-, PI-, PD und PID-Regler). Basierend auf dem aus der Literatur bekannten digitalen PID-Stellalgorithmus [129]

$$\mathbf{x}_{PWM}[k] = K_r \left(\mathbf{e}[k] + \frac{t_A}{T_n} \sum_{v=0}^{k-1} \mathbf{e}[v] + \frac{T_v}{t_A} (\mathbf{e}[k] - \mathbf{e}[k-1]) \right), \; k = 0, 1, 2, ... \tag{4.57}$$

lassen sich dabei durch geeignete Kombination (d. h. Weglassen einzelner additiver Terme in Gleichung (4.57) einfach die verschiedenen linearen Reglertypen umsetzen. Vorteile linearer Reglerstrukturen sind die Eignung für nahezu alle Prozesstypen, die Robustheit und der geringe Realisierungsaufwand. In [19] wurden die beschriebenen Algorithmen zur Regelung der Freiheitsgrade der Roboterhalseinheit sowohl für einfache als auch gekoppelte Bewegungsvorgänge umgesetzt und miteinander verglichen. Dabei hat sich der P-Regler als am besten geeigneter Regler erwiesen, da nur ein Parameter einzustellen ist und zudem ein vergleichbar genaues, schnelles und robustes Regelverhalten erreicht werden kann. Am Forschungszentrum für Informatik (Karlsruhe) wurde ein solcher P-Regler in Form eines MCA-Moduls umgesetzt, entsprechend parametriert und zur Positionsregelung der realen Roboterhalseinheit eingesetzt. Die erzielten Ergebnisse der experimentellen Validierung sind in den Bildern 4.25 und 4.26 dargestellt.

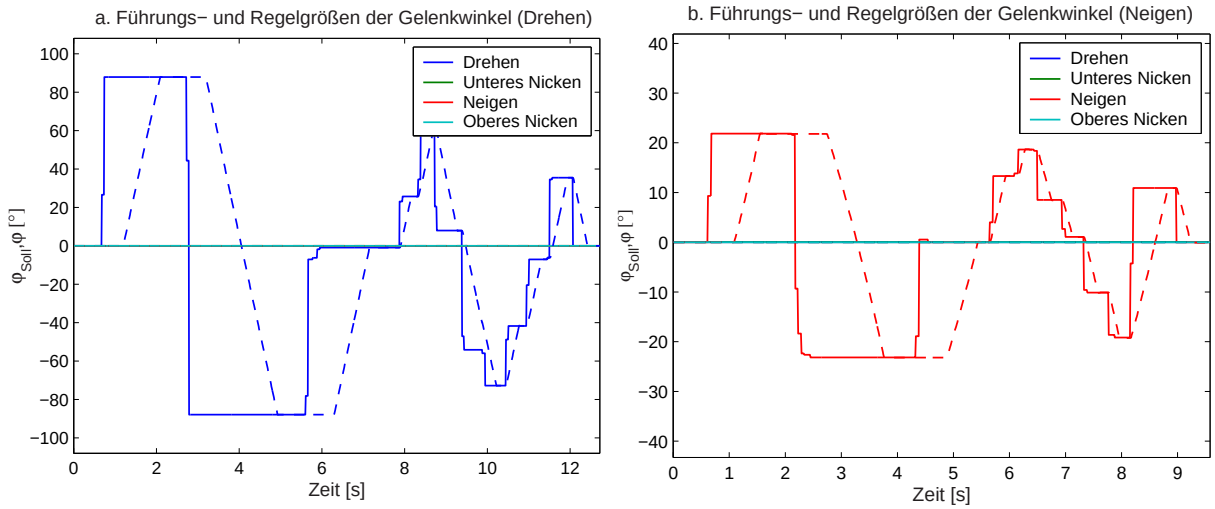


Bild 4.25: Gemessene Führungs- (durchgezogen) und Regelgrößenverläufe (gestrichelt) am Beispiel von Bewegungen eines einzelnen Freiheitsgrades (links Drehen, rechts Neigen)

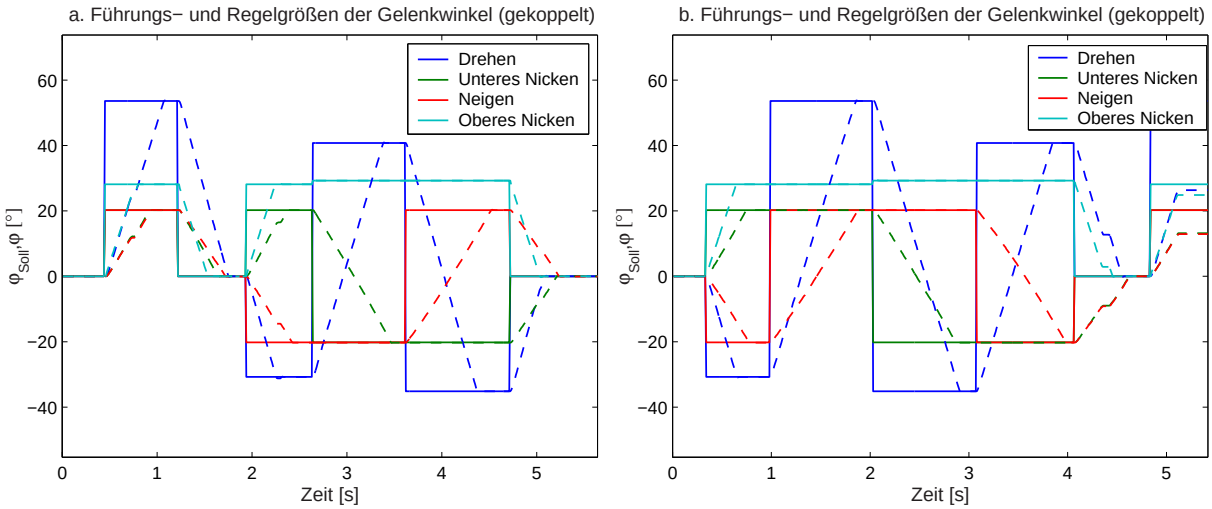


Bild 4.26: Gemessene Führungs- (durchgezogen) und Regelgrößenverläufe (gestrichelt) am Beispiel von gekoppelten Bewegungen aller vier Freiheitsgrade

Sowohl für die Regelung eines einzelnen Freiheitsgrades (Bild 4.25) als auch für die gekoppelte Regelung aller vier Freiheitsgrade (Bild 4.26) ist mit dem simulativ in [19] entworfenen Regler ein *gutes* Regelverhalten zu erzielen (kein Überschwingen, stationäre Endlage erreicht, ausreichende Schnelligkeit). Das hauptsächliche Ziel der hier dargestellten sehr schnellen Verläufe bestand dabei darin, die maximale Halsdynamik zu testen. Die Stellglieder befinden sich dabei größtenteils in den Stellbegrenzungen.

4.3.2.2 Bildbasierte Positionsregelung

Bei der Struktur der bildbasierten Positionsregelung handelt es sich um eine Kaskadenregelung. Eine Kaskadenregelung entsteht dadurch, dass zusätzlich zur Regelgröße weitere Systemgrößen messtechnisch erfasst und diese als unterlagerte Regelkreise zurückführt werden. Zur Regelung innerhalb der unterlagerten Regelkreise können wiederum lineare Regler verwendet werden. Vorteil bei Verwendung einer solchen Struktur ist, bei geeigneter Wahl der Reglerparameter, ein verbessertes dynamisches Verhalten des gesamten Regelkreises. Diese Struktur ist derzeit Standard im Bereich der Industrierobotik [6, 7, 168, 254]. Bild 4.27 zeigt die Struktur von Regler und Regelstrecke (Halseinheit und Kamerasystem) für die bildbasierte Positionsregelung der Roboterhalseinheit.

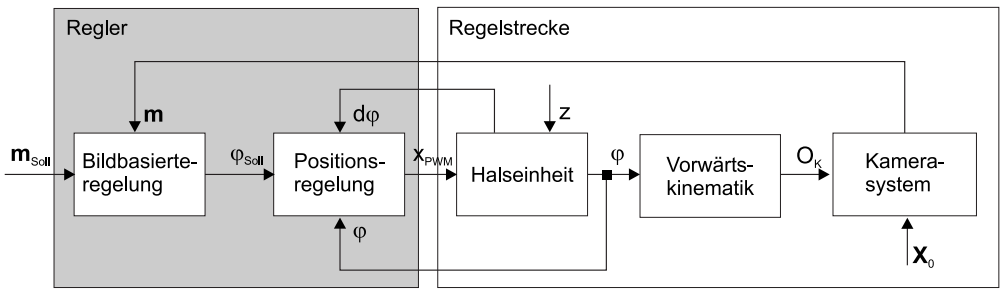


Bild 4.27: Schematische Darstellung von Regler und Regelstrecke (Halseinheit und Kamerasystem) für die bildbasierten Positionsregelung der Roboterhalseinheit

Allgemeines Ziel der bildbasierten Positionsregelung ist es, die Roboterhalseinheit erkannten, sich u. U. frei im Raum bewegenden Objekten oder Personen nachzuführen (engl.: tracken). Dies ist vor allem bei der Roboter-Mensch-Interaktion und -Kommunikation von großer Bedeutung (Stichwort Aufmerksamkeitssteuerung).

Um den hier vorgestellten bildbasierten Regelungsalgorithmus zu testen, wird im Weiteren davon ausgegangen, dass die zu verfolgenden Objekte $\mathbf{X}_0$ vom Kamerasystem und der entsprechenden Bilderkennungssoftware eindeutig zugeordnet werden können. Unter dieser Voraussetzung hat der bildbasierte Regler die Aufgabe, die von der Kamera berechnete Projektion $\mathbf{m} = [u \quad v \quad 1]^T$ des zu verfolgenden Objekts auf der Bildebene in eine hinreichend genaue Umgebung der gewünschten Position $\mathbf{m}_{Soll} = [u_{Soll} \quad v_{Soll} \quad 1]^T$ zu überführen. Die dazu zur Verfügung stehenden Stellgrößen sind die Gelenkwinkel-Führungsgrößen $\boldsymbol{\varphi}_{Soll}$ des Positionsregelungsalgorithmus aus dem vorigen Abschnitt. Dabei hat es sich für alltägliche Aufgaben als ausreichend erwiesen, nur die Freiheitsgrade für das Drehen (3) und das obere Nicken (4) des Halses durch den bildbasierten Regler anzusteuern. Die Berechnung der Stellgrößen dieser Freiheitsgrade erfolgt wiederum durch einen einfachen P-Stellalgorithmus:

$$\varphi^*_{Soll(j)}[k] = \begin{cases} -K_{BP}(u_{Soll}[k] - u[k]) & \text{für} \quad j = 3 \\ -K_{BP}(v_{Soll}[k] - v[k]) & \text{für} \quad j = 4 \\ 0 & \text{sonst.} \end{cases} \tag{4.58}$$

Um die Regleraktivität zu dämpfen, werden analog zu Abschnitt 4.2.2.2 Totzonenbereiche $u_{tot_{aus}}$, $v_{tot_{aus}}$ bzw. $u_{tot_{ein}}$, $u_{tot_{ein}}$ definiert. Befindet sich die Projektion des zu verfolgenden Objektes auf der Bildebene innerhalb dieser Totzonenbereiche mit $u_{tot_{aus}} > u_{tot_{ein}}$ und $v_{tot_{aus}} > v_{tot_{ein}}$ (Hysteresen), dann ist das Regelziel erreicht und die Stellgrößen werden eingefroren. Erst nach Verlassen der äußeren Totzonenbereiche $u_{tot_{aus}}$ bzw. $v_{tot_{aus}}$ wird die bildbasierte Regelung wieder aktiviert [189].

$$\varphi_{Soll(j)}[k] = \begin{cases} \varphi_{Soll(j)}[k-1] & \text{für} \quad (|u[k]| < u_{tot_{ein}}) \wedge (|v[k]| < v_{tot_{ein}}) \\ & \vee \quad (|u[k]| < u_{tot_{aus}} \wedge |u[k-1]| < u_{tot_{ein}}) \\ & \wedge \quad (|v[k]| < v_{tot_{aus}} \wedge |v[k-1]| < v_{tot_{ein}}) \\ \varphi^*_{Soll(j)}[k] & \text{sonst.} \end{cases} \tag{4.59}$$

Besteht das visuelle System des Roboters aus einem Stereo-Kamerasystem, können durch die unterschiedlichen Positionen des abgebildeten Objekts auf den Bildebenen der beiden Kameras widersprüchliche Stellgrößenanforderungen an den Positionsregler auftreten. Der einfachste Weg dies zu vermeiden ist, nur eine Kamera aktiv zur bildbasierten Regelung einzusetzen.

4.3.3 Ergebnisse

Um die Funktionalität des vorgeschlagenen Entwurfskonzepts am Beispiel der Roboterhalseinheit nachzuweisen, wird in den folgenden Abschnitten Aufgabenwissen für rein positionsgeregelte Kopfgesten und eine bildbasiert geregelte 'Aufmerksamkeitssteuerung' (Objekt im Raum suchen und wenn gefunden, dann mit der Halseinheit bildbasiert verfolgen) generiert. Der Entwurf erfolgt für beide Beispiele auf der Entwicklungsumgebung. Die Roboterhalseinheit und das verwendete Kamerasystem werden dazu mit den im Abschnitt 4.3.1 vorgestellten Modellen simuliert.

4.3.3.1 Kopfgesten

Gestenbewegungen der Roboterhalseinheit wie z. B. das Verneinen (Kopfschütteln), das Zustimmmen (Kopfnicken) oder das Abwägen (Kopfneigen) lassen sich, wie die bereits in Abschnitt 4.2.3 vorgestellten Greifergesten, durch eine Instanz der Basis-Elementaroperationen $EA_{i,u}$ 'Fahren' darstellen. Das

entsprechende Koordinierungsnetz besteht deshalb aus einem einzigen diskreten Platz. Die Umsetzung erfolgt mit der in Kapitel 3.3 vorgestellten Implementierung der Roboter-Steuerungsarchitektur auf der Entwicklungsplattform. In diesem Beispiel werden drei Kopfgesten (’Verneinen’ (Kopfschütteln), ’Zustimmen’ (Kopfnicken) und ’Abwägen’ (Kopfneigen)) sequentiell ausgeführt. Die für die Generierung der rampenförmigen Führungsgrößentrajektorien erforderlichen maximalen bzw. minimalen Sollpositionen $\varphi_{Soll(Max(j))}$ bzw. $\varphi_{Soll(Min(j))}$ der Führungsgrößentrajektorien sind für die drei Kopfgesten in Tabelle 4.6 dargestellt. Es handelt sich bei den vorgestellten Gesten prinzipiell um nicht gekoppelte Bewegungen, d. h. es wird nur ein Freiheitsgrad der Roboterhalseinheit aktiv angesteuert. Zur Ausführung einer verneinenden Geste ist der dritte Freiheitsgrad (Drehbewegung), für eine zustimmende oder abwägende Kopfgeste sind dagegen der vierte (oberes Nicken) bzw. der zweite (Neigen) Freiheitsgrad der Halseinheit anzusteuern. Ungewollte Bewegungen der anderen drei Freiheitsgrade ergeben sich in Folge der dynamischen Kopplung (Abschnitt 4.3.1) und müssen vom Regelungsalgorithmus ausgeregelt werden. Die Trajektorienart und die Sollpositionen der Freiheitsgrade sind wiederum so gewählt, dass das Robotersystem möglichst menschenähnlich aussehende Bewegungen (hier Kopfgesten) ausführt.

Kopfgeste (angesteuerter DOF)	Sollpositionen der Freiheitsgrade (alternierend)
Verneinen (Drehen)	$\varphi_{Soll(Max(3))} = 10°$ und $\varphi_{Soll(Min(3))} = -10°$
Zustimmen (Nicken)	$\varphi_{Soll(Max(4))} = 10°$ und $\varphi_{Soll(Min(4))} = -10°$
Abwägen (Neigen)	$\varphi_{Soll(Max(2))} = 10°$ und $\varphi_{Soll(Min(2))} = -10°$

Tabelle 4.6: Maximale bzw. minimale Sollpositionen der jeweiligen Freiheitsgrade für die drei untersuchten Kopfgesten

Die Regelung erfolgt mit dem diskreten PID-Stellalgorithmus nach Gleichung (4.57), der als reiner P-Regler ausgeführt ist.

Bild 4.28 zeigt als Ergebnis die Durchführung der drei in diesem Beispiel ausgeführten Kopfgesten. Abgebildet sind der Verlauf der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel $\varphi_{Soll(j)}$, $\varphi_{(j)}$ (a) und der Verlauf der Lastmomente $M_{L(j)}$ (b) über der Zeit.

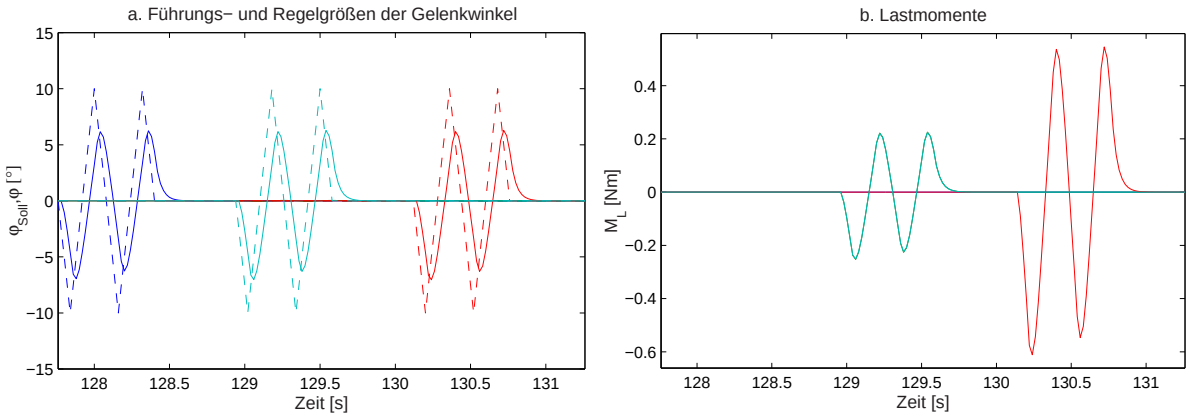


Bild 4.28: Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel (a) sowie Lastmomente (b) über der Zeit für die drei untersuchten Kopfgesten

Bis zum Simulationszeitpunkt $t = 128.6$ s wird die erste Elementaraktion (’Verneinen’) aktiv ausgeführt. Dazu wird der dritte Freiheitsgrad der Roboterhalseinheit angesteuert (Tabelle 4.6). Die Trajektorienge-

nerierung erzeugt alternierende Führungsgrößentrajektorien $\varphi_{Soll(Max(3))} = 10°$ bzw. $\varphi_{Soll(Min(3))} = -10°$ mit einer Steigung von $\Delta\varphi = 2.5°/\text{ta}$ (Bild 4.28 a). Der Regelungsalgorithmus hat die Aufgabe die entstehenden Regeldifferenzen zu minimieren. Deutlich zu erkennen ist, dass bei der Drehbewegung keine Lastmomente auf die Roboterhalseinheit wirken (Bild 4.28 b), da sich der Schwerpunkt des Roboterkopfes immer auf der Drehachse befindet [19].

Die übergeordnete Verarbeitung erfolgt wiederum mit dem bekannten Statusnetz. Zum Simulationszeitpunkt $t = 129.0$ s hat die erste Elementaraktion das vorgegebene Regelungsziel erreicht. Die Roboter-Steuerungsarchitektur aktiviert eigenverantwortlich die Transition T12 im Statusnetz und der Platz A5 'Active' wird frei. Die Ressourcenverwaltung überprüft die zweite im Aufgabenspeicher hinterlegte Elementaraktion (hier 'Zustimmmen') und gibt diese zum Simulationszeitpunkt $t = 129.0$ s zur Ausführung frei. Die Trajektoriengenerierung erzeugt wiederum alternierende Führungsgrößentrajektorien für den entsprechenden Freiheitsgrad (oberes Nicken). Trotz der bei dieser Bewegung auftretenden Lastmomente (Schwerpunktachse befindet sich bei Auslenkung außerhalb der Drehachse, Bild 4.28 b), ist der Regler in der Lage, die vorgegebenen Führungsgrößentrajektorien einzuregeln (Bild 4.28 a) und damit die Elementaraktion zum Simulationszeitpunkt $t = 130.0$ s erfolgreich abzuschließen.

Bei der dritten Elementaraktion 'Abwägen' (aktive Ausführung ab dem Simulationszeitpunkt $t = 130.0$ s, Bild 4.28 a) ergibt sich, aufgrund des größeren Abstands der Schwerpunktachse von der Drehachse bei der Auslenkung, ein nochmals höheres Lastmoment (Bild 4.28 b). Der Regler ist trotzdem in der Lage, das geforderte Sollverhalten zu gewährleisten und zum Simulationszeitpunkt $t = 131.4$ s ist auch die dritte Elementaraktion erfolgreich ausgeführt. Das hauptsächliche Ziel der hier dargestellten sehr schnellen Verläufe bestand dabei darin, die maximale Halsdynamik zu testen. Die Stellglieder befinden sich dabei größtenteils in den Stellbegrenzungen.

Durch diese petrinetzbasierte Beschreibung des übergeordneten Ablaufs bei der Aufgabenverarbeitung kann der Bediener oder die Aufgabenplanung jederzeit Aufgaben in Form von Elementaraktionen an die Roboter-Steuerungsarchitektur kommandieren. Diese werden im Aufgabenspeicher abgelegt und vom Ressourcenmanagement autonom verarbeitet. Weiterer Vorteil dieses flexiblen Konzepts ist, dass auch während der Laufzeit noch Planänderungen durch den Benutzer oder das Robotersystem selbst vorgenommen werden können. Dazu werden die entsprechenden Elementaraktionen aus dem Aufgabenspeicher gelöscht. Neben dem irreversiblen Löschen kann zudem durch eine geeignete Aufgabenpriorisierung die Ausführungsreihenfolge beeinflusst werden. Beispiele hierzu werden in Kapitel 5 ausführlich vorgestellt.

4.3.3.2 Aufmerksamkeitssteuerung

In diesem Abschnitt erfolgt die Umsetzung und Evaluierung des vorgeschlagenen Gesamtkonzepts am Beispiel der 'Aufmerksamkeitssteuerung' des anthropomorphen Roboter-Kopfes. Dazu sind im Rahmen des vorgeschlagenen Konzepts aufgaben- und teilkomponentenspezifische Elementaroperationen $EA_{i,u}$ zu entwerfen und zu hierarchisch strukturierten Petri-Netzen (Koordinierungsnetze) zusammenzufassen. Die Aktion 'Aufmerksamkeitssteuerung' setzt sich dabei aus jeweils einer Instanz der Elementaraktion 'Suchen' und 'Verfolgen' aus Tabelle 2.1 zusammen.

Bild 4.29 (unten Mitte) zeigt die Umsetzung des entsprechenden Koordinierungsnetzes auf der Entwicklungsplattform. Ist das zu verfolgende Objekt (z. B. eine vom Roboter zu greifende Tasse oder der Benutzer) lokalisiert[4], so ist der Platz $EA_{Ko,1}$ der Aktion 'Aufmerksamkeitssteuerung' markiert. Das Ziel der bildbasierten Regelung der Roboter-Halseinheit ist es, den Bildpunkt des zu verfolgenden

[4]Lokalisiert: Im Sichtfeld der Kamera, interpolierte verdeckte Position oder bekannte Lage im Greifer.

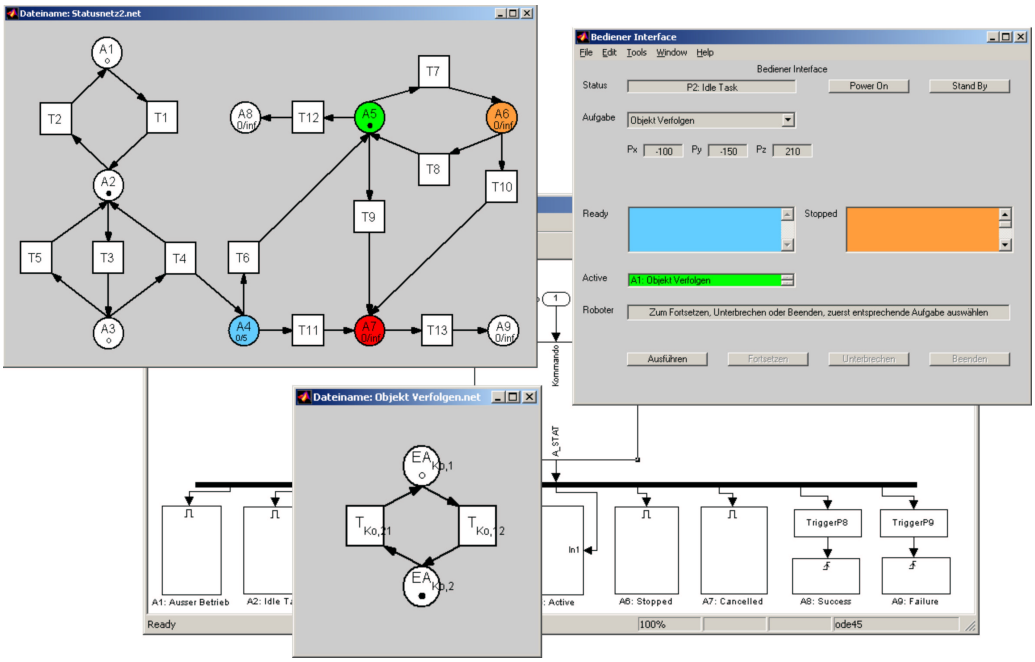


Bild 4.29: Umsetzung der 'Aufmerksamkeitssteuerung' auf der Entwicklungsplattform. Dargestellt sind die Realisierung der Roboter-Steuerungsarchitektur nach Kapitel 3.3 (oben links und rechts) sowie das Koordinierungsnetz für die 'Aufmerksamkeitssteuerung' (unten Mitte)

Objekts innerhalb eines vorgegebenen Totzonenbereichs ($u_{tot_{ein}}$, $v_{tot_{ein}}$ bzw. $u_{tot_{aus}}$, $v_{tot_{aus}}$) im Bildkoordinatensystem einer der beiden Kameras zu positionieren (Abschnitt 4.3.2.2, Gleichung (4.59)). Bei Objektbewegungen im Weltkoordinatensystem (WKS) muss der Regler die Halseinheit mitbewegen, um die Position im Bildkoordinatensystem zu halten. Falls das Objekt von der Kamera nicht mehr lokalisiert werden kann ($T_{Ko,12}$), startet der Kopf einen Suchvorgang (markierte Stelle $EA_{Ko,2}$). Dazu werden alternierende, rampenförmige Führungsgrößentrajektorien mit maximalen bzw. minimalen Sollpositionen $\varphi_{Soll(Max(3))} = 90°$ bzw. $\varphi_{Soll(Min(3))} = -90°$ für den dritten Freiheitsgrad (Drehen) der Roboterhalseinheit generiert und so der Raum horizontal gescannt. Ist das Robotersystem dabei, dass zu verfolgende Objekt durch Kopfbewegungen zu suchen ($EA_{Ko,2}$), so findet bei einem Erfolg ein Markenfluss mit $T_{Ko,21}$ zurück auf die Stelle $EA_{Ko,1}$ statt. Tabelle 4.7 erläutert die entsprechenden Stellen und Transitionen des Koordinierungsnetzes.

Stelle	Bedeutung
$EA_{Ko,1}$	Objekt bildbasiert verfolgen
$EA_{Ko,2}$	Objekt suchen
Transition	**Bedeutung**
$T_{Ko,12}$	Objekt nicht lokalisiert
$T_{Ko,21}$	Objekt lokalisiert

Tabelle 4.7: Plätze und Transitionen der Aktion 'Aufmerksamkeitssteuerung' ohne Ausnahmesituationsbehandlung (vgl. Tabelle 2.8)

Bild 4.30 zeigt als Ergebnis der Simulation die Verläufe der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel $\varphi_{Soll(j)}$, $\varphi_{(j)}$ für den dritten Freiheitsgrad (Drehen), den Verlauf der

Platzbelegung im Koordinierungsnetz über der Zeit sowie die Verläufe der Bildpositionen des zu verfolgenden Objekts auf den beiden Bildebenen des Stereo-Kamerasystems (grünes Kreuz: eingeregelte Endposition).

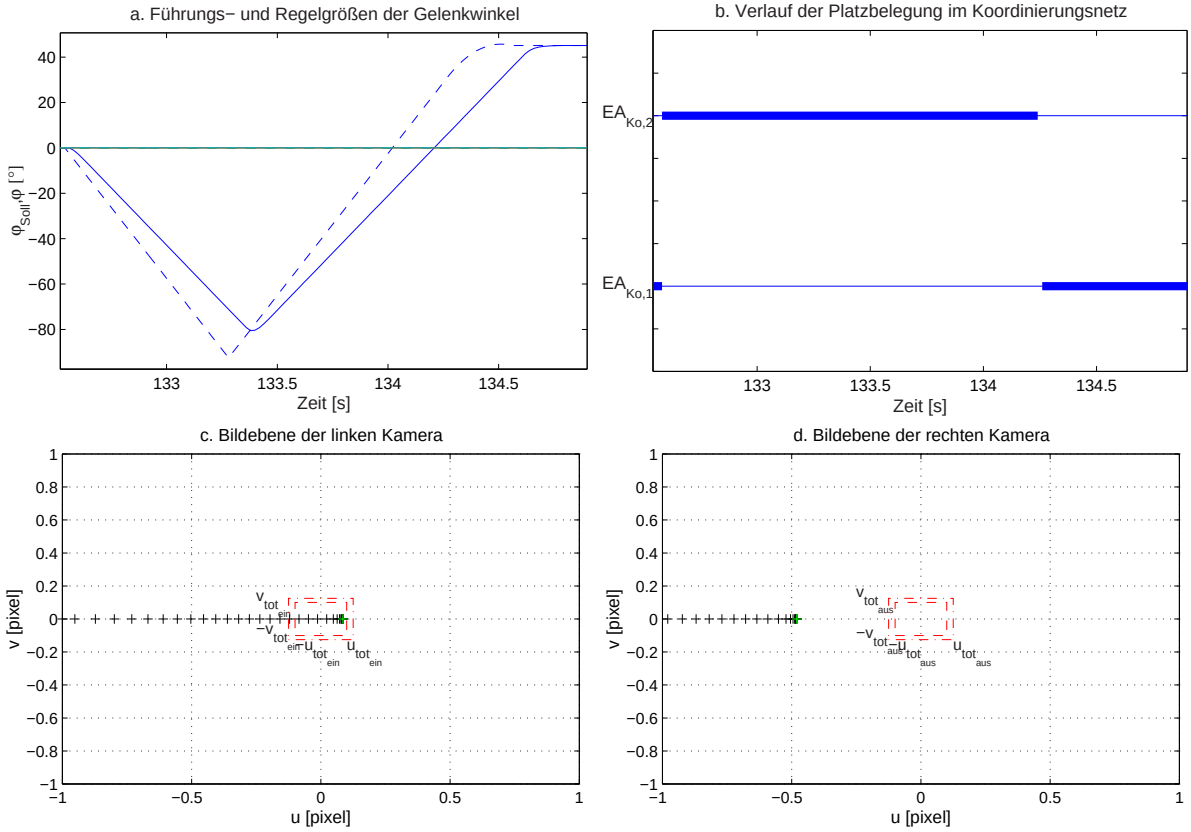


Bild 4.30: Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel (a) und Verlauf der Platzbelegung im Koordinierungsnetz (b) über der Zeit sowie Verläufe der Bildpositionen des zu verfolgenden Objekts auf den beiden Bildebenen des Stereo-Kamerasystems (c und d)

Nach der Freigabe der Aktion durch die Ressourcenverwaltung wird zunächst das zugehörige Koordinierungsnetz mit seiner initialen Platzbelegung geladen und ausgeführt (hier $\mathcal{M} = \{(EA_{Ko,1}, 1)\}$). Ab dem Simulationszeitpunkt $t = 132.6$ s registriert der Regelungsalgorithmus, dass sich das zu verfolgende Objekt außerhalb des Sichtfeldes des Stereo-Kamerasystems befindet und der Platz $EA_{Ko,2}$ der Aktion 'Aufmerksamkeitssteuerung' wird markiert (Bild 4.30 b). Der Roboterkopf startet einen Suchvorgang. Dazu werden alternierende, rampenförmige Führungsgrößentrajektorien für den dritten Freiheitsgrad (Drehen) der Roboterhalseinheit generiert und vom Regler rein positionsgeregelt umgesetzt. Zum Simulationszeitpunkt $t = 133.3$ s hat die Roboterhalseinheit dabei die konstruktiv vorgegebene maximale negative Auslenkung (Linksdrehung) für den dritten Freiheitsgrad $\varphi_{Soll(Min(3))} = -90°$ erreicht (Bild 4.30 a). Danach wird die Suchrichtung umgedreht und der Kopf dreht sich nach rechts. Ab dem Simulationszeitpunkt $t = 134.3$ s wird das zu verfolgende Objekt vom Stereo-Kamerasystem lokalisiert. Durch Aktivieren der Transition $T_{Ko,21}$ wird der Platz $EA_{Ko,2}$ des Koordinierungsnetzes markiert (Bild 4.30 b). Jetzt ist es das Ziel der Regelung, den Bildpunkt des zu verfolgenden bzw. zu erfassenden Objekts innerhalb des vorgegebenen Totzonenbereichs ($u_{tot_{ein}}$, $v_{tot_{ein}}$ bzw. $u_{tot_{aus}}$, $v_{tot_{aus}}$) um den Mittelpunkt zu positionieren (Bilder 4.30 c und d). Zur bildbasierten Regelung wird in diesem Beispiel nur die linke Kamera des Stereo-Kamerasystems verwendet (Abschnitt 4.3.2.2). Zu erkennen ist,

dass der Regler das vorgegebene Regelungsziel (Bildpunkt für einen vorgegebenen Zeitraum innerhalb des inneren Totzonenbereichs positionieren, grünes Kreuz zeigt die eingeregelte Endposition) erreicht (Bild 4.30 c) und damit die Trackingbewegung erfolgreich durchgeführt werden konnte. Daraufhin wird die Transition T_{Su} im Statusnetz aktiviert und das Robotersystem ist bereit zur Ausführung einer weiteren Aufgabe.

4.4 Diskussion

In diesem Kapitel wurde erstmals anhand von ausgewählten Roboterkomponenten gezeigt, wie mit dem in Abschnitt 2.2 vorgestellten, ganzheitlichen Entwurfskonzept Aufgabenwissen für humanoide Roboter generiert werden kann. Um die Funktionalität des vorgeschlagenen Entwurfskonzepts zu validieren, wurden aufgabenspezifisch hierarchisch strukturierte Petri-Netze (Koordinierungsnetze) entworfen und von der auf der Entwicklungsplattform umgesetzten Steuerungsarchitektur des Robotersystems petrinetzbasiert ausgeführt (vgl. Abschnitt 3.3).

Der Nachweis der Funktionalität wurde hier am Beispiel von zwei Roboter-Teilkomponenten mit jeweils zwei grundlegend verschiedenen Aufgabentypen erbracht. Beim fluidisch angetriebenen Mehrfinger-Robotergreifer des Forschungszentrum Karlsruhe wurden eine Greifergeste (Zeigegeste) und ein komplexer Greifvorgang (Präzisionsgriff) erfolgreich umgesetzt. Für die anthropomorphe Roboterhalseinheit wurde das Konzept anhand dreier sequentiell ausgeführter Kopfgesten (Kopfnicken, Kopfschütteln und Kopfneigen) und einer Trackingaufgabe erfolgreich evaluiert. Weiterhin konnte in den Beispielen gezeigt werden, dass aufgrund der Flexibilität des vorgeschlagenen petrinetzbasierten Entwurfskonzepts Vorteile bei der erfolgreichen Ausführung bestimmter Roboteraufgaben entstehen (intelligentes Aufteilen der Aufgabe in kleinere Teilaufgaben (Aktionsfolgen) um Restriktionen der Mechanik zu umgehen. Beispiel: Gestenbewegung des Greifers).

Um mit dem vorgestellten Entwurfskonzept erfolgreich Roboteraufgaben ausführen zu können, ist neben der Definition von aufgabenspezifischen Petri-Netzen (Koordinierungsnetze) auch die Generierung von Führungsgrößentrajektorien und der Entwurf geeigneter Regelungsalgorithmen notwendig. Deshalb wurden in den Abschnitten 4.2.2 und 4.3.2 Algorithmen zur Positions- und/oder Kraftregelung bzw. zur Positions- und bildbasierten Positionsregelung der Freiheitsgrade des Robotergreifers und der Roboterhalseinheit vorgestellt sowie die Vor- und Nachteile dieser Konzepte diskutiert.

Um die entworfenen und parametrierten Roboteraufgaben auf der Entwicklungsplattform durch Simulationsstudien testen zu können, wurden weiterhin die mathematischen Modelle der beiden hier beispielhaft untersuchten Roboter-Teilkomponenten aufgestellt. Dies beinhaltet jeweils die kinematischen und dynamischen Zusammenhänge sowie die Modellierung der Teilsysteme des Antriebs. Hierzu wurden sowohl Methoden der theoretischen Modellbildung als auch Verfahren der experimentellen Modellbildung eingesetzt. Diese Modelle sind weiterhin wichtig für die u. U. erforderliche modellbasierte Regelung, modellbasierte Kontakterkennung und/oder modellbasierte Fehlerdetektion (vgl. Abschnitt 2.3 und Kapitel 5).

Neben dem in diesem Kapitel behandelten Routinebetrieb bei der Ausführung von Aufgaben muss ein autonomes Robotersystem in der Lage sein, Fehler selbstständig zu erkennen und durch geeignete Reaktion zu behandeln (Ausnahmesituationsbehandlung). Deshalb wurde im Abschnitt 2.3 eine aus einem allgemeinen mehrstufigen Überwachungsprozess abgeleitete Grundstruktur für ein neues Überwachungskonzept von humanoiden Robotern präsentiert, das zudem in der Lage ist, die Strategien zur autonomen Fehlerdetektion und Problemlösung einfach und systematisch in die vorgegebene Roboter-Steuerungsarchitektur zu integrieren. Beispiele für die Funktionalität dieses Konzepts erfolgen im Kapitel 5.

5 Neue modellbasierte Überwachung ausgewählter Roboterkomponenten

5.1 Übersicht

Das im Abschnitt 2.3 vorgestellte neue, modulare Konzept zur Überwachung von Roboterkomponenten mit Petri-Netzen gliedert sich in die modellbasierte Fehlerdetektion und die petrinetzbasierte Problemlösung. Die modellbasierte Fehlerdiagnose hat dabei die Aufgabe, die Zustände aller Komponenten der unterlagerten Regelkreise, bestehend aus Basis-Regelung (Stabilitätsüberwachung Abschnitt 2.3.1.3) und Roboterhardware (Hardwareüberwachung, Abschnitt 2.3.1.4), zur Laufzeit zu überwachen und entsprechend der geforderten Aufgabe zu bewerten. Die Funktionalität von Stabilitäts- und Hardwareüberwachung wird in den Abschnitten 5.2.1 und 5.2.2 am Beispiel von Bewegungs- und Greifvorgängen des flexiblen Mehrfinger-Robotergreifers aus Abschnitt 4.2 demonstriert. Zusätzlich wurde die Stabilitätsüberwachung auf der Roboterplattform [72] des Fraunhofer Instituts IITB implementiert und anhand von Bewegungsvorgängen mit einem 7-achsigen AMTEC-Roboterarm experimentell validiert (Abschnitt 5.2.1).

Im Anschluss erfolgt die Umsetzung der vorgeschlagenen petrinetzbasierten Problemlösung. Die Problemlösung nicht planbarer Ereignisse kann nach Abschnitt 2.3.2 entweder durch bewusst steuerbare oder unbewusste, autonome Strategien (Reflexe) erfolgen. Aus diesem Grund erfolgt im Abschnitt 5.3 die simulative Evaluierung von bewusst steuerbaren und autonomen, reflexartigen Problemlösungsstrategien. Diese Strategien werden dabei zur Bewältigung von Ausnahmesituationen verwendet, die bei simulierten Bewegungsvorgängen des Robotergreifers und der anthropomorphen Halseinheit auftreten.

Abschnitt 5.4 zeigt, wie das hier erstmals vorgestellte Gesamtkonzept zur Prioritätsverwaltung von Roboteraufgaben verwendet werden kann. So ist es beispielsweise möglich, den verschiedenen im Aufgabenspeicher hinterlegten Aufgaben unterschiedliche Prioritäten zu zuordnen oder diese komplett aus dem Aufgabenspeicher zu löschen und damit die Ausführungsreihenfolge zu manipulieren.

5.2 Modellbasierte Fehlerdetektion

5.2.1 Erweiterte Stabilitätsüberwachung

Um die Funktionalität der in Abschnitt 2.3.1.3 vorgestellten integrierten Stabilitätsüberwachung nachzuweisen, wird zunächst der Annäherungsvorgang eines einzelnen Fingerglieds des hydraulisch betriebenen Mehrfinger-Robotergreifers aus Abschnitt 4.2 an ein ortsfestes, verformbares Objekt mit einer anschließenden definierten Kontaktphase analysiert. Anhand von drei Szenarien (1-3) mit unterschiedlichen Materialkonstanten k_O (Gleichung (4.10)) und Reglerparametern K_{RF} und T_{NF} (Gleichung (4.34)),

aber sonst identischen Simulationsparametern, soll detailliert die Funktion des entwickelten Überwachungskonzepts verdeutlicht werden. Um die Funktionsfähigkeit des Konzepts auch im Mehraktorfall nachzuweisen, wird als viertes Szenario (4) der Annäherungsvorgang eines Fingers i mit $j_{Gr} = 1..3$ Gelenken an ein Objekt mit einer anschließenden Kontaktphase vorgestellt. Anhand dieses Szenarios sollen die Reaktionen der Stabilitätsüberwachung auf nicht zu realisierende Vorgaben der Bahnplanung dargestellt werden. Tabelle 5.1 zeigt die Parameter der untersuchten Szenarien.

Parameter Szenario	Anzahl Gelenke	k_O	K_{RF}	T_{NF}
1	1	0.5	0.2	0.2
2	1	0.5	**20**	**0.08**
3	1	**5**	**20**	**0.08**
4	**3**	0.5	0.2	0.2

Tabelle 5.1: Parameter der analysierten Szenarien

Das für die Umsetzung des Annäherungsvorgangs erforderliche Koordinierungsnetz besteht für alle hier untersuchten Szenarien aus einem diskreten Platz und ist durch eine Instanz der Elementaraktion 'Fahren' beschreibbar. Für die anschließende Kontaktphase ist eine Instanz der Elementaraktion 'Zugreifen' notwendig (vgl. Tabelle 2.1). Die Transitionen des entsprechenden Koordinierungsnetzes hängen in diesem Beispiel allein von der Kontaktsituation der Fingerglieder des flexiblen Mehrfinger-Robotergreifers ab. Die Umsetzung der rampenförmigen Führungsgrößentrajektorien erfolgt in diesem Beispiel durch den hybriden Kraft-Positions-Regelungsalgorithmus aus Abschnitt 4.2.2. Der Robotergreifer und das bei der Annäherung kontaktierte Objekt werden mit den im Abschnitt 4.2.1 vorgestellten hydraulischen Modellen simuliert.

Die Bilder 5.1 bis 5.3 zeigen jeweils den Führungs- (gestrichelt) und Regelgrößenverlauf (durchgezogen) des Gelenkwinkels $\varphi_{Soll(i,j)}, \varphi_{(i,j)}$ mit den dazugehörenden Führungsgrößenintervallen (gepunktet) $\varphi_{Max(i,j)}$, $\varphi_{Min(i,j)}$ (a), den Ventil- $x_{V(i,j)}$ und Pumpenzustand x_P und die Kontaktsituation $K_{(i,j)}$ (b), den Führungs- und Regelgrößenverlauf des Gelenkmoments $M_{Soll(i,j)}, M_{(i,j)}$ mit den dazugehörenden Führungsgrößenintervallen $M_{Max(i,j)}$, $M_{Min(i,j)}$ (c) sowie dem berechneten Stabilitätsgrad S (d) über der Zeit. Die Stabilitätseinschätzung S bezieht sich dabei in allen hier dargestellten Beispielen (Szenarien 1-4) auf die Mehrgrößenregelung von Position und Moment.

Bei den drei ersten Szenarien ist deutlich der Umschaltvorgang der Führungsgrößenintervalle zum Zeitpunkt $t = 1.4$ s zu erkennen (vgl. Abschnitt 2.3.1.3 und Bild 2.14). Hier findet der Objektkontakt statt. Der Regler schaltet vom positions- in den kraftgeregelten Modus. Während der *freien Bewegung* beträgt das Gelenkwinkel-Führungsgrößenintervall $\pm 3°$ Grad und das Momenten-Führungsgrößenintervall ± 0.25 Nm. Im *Kontaktfall* ist das Gelenkwinkel-Führungsgrößenintervall auf $\pm 17.5°$ Grad und das Momenten-Führungsgrößenintervall auf ± 0.025 Nm eingestellt. Ab dem Zeitpunkt 1.4 s arbeitet der Regler als reiner Momentenregler und regelt das von der Führungsgröße vorgegebene Moment $M_{Soll(i,j)}$ ein.

In Szenario 1 beträgt die Materialkonstante $k_O = 0.5$. Das Objekt ist damit leicht verformbar. Die Regelgröße $M_{(i,j)}$ befindet sich bereits kurz nach Objektkontakt, bedingt durch die in diesem Beispiel verwendete geringe Objektsteifigkeit und 'passend eingestellte' Reglerparameter (Tabelle 5.1), innerhalb ihres vorgegebenen Intervalls $M_{Max(i,j)}$, $M_{Min(i,j)}$. Weiterhin bleibt auch die Regelgröße $\varphi_{(i,j)}$ innerhalb ihres vorgegebenen Intervalls $\varphi_{Max(i,j)}$, $\varphi_{Min(i,j)}$.

Bild 5.1 (d) zeigt den Verlauf des Stabilitätsgrads S. Die Stabilitätsüberwachung detektiert die leicht verzögerte Reaktion des Reglers auf das Verlassen der Führungsgrößenintervalle ab $t = 1.4$ s mit einem Stabilitätsgrad von $S = 0.85$, d. h mit einer Stabilitätseinschätzung zwischen 'STABIL' und 'NICHT ENTSCHEIDBAR'. Ab dem Simulationszeitpunkt $t = 1.5$ s baut der Regler die Regeldifferenz ab, und es ergibt sich ein Stabilitätsgrad von $S = 1$, d. h. die Situation wird wieder als vollständig 'STABIL' eingestuft.

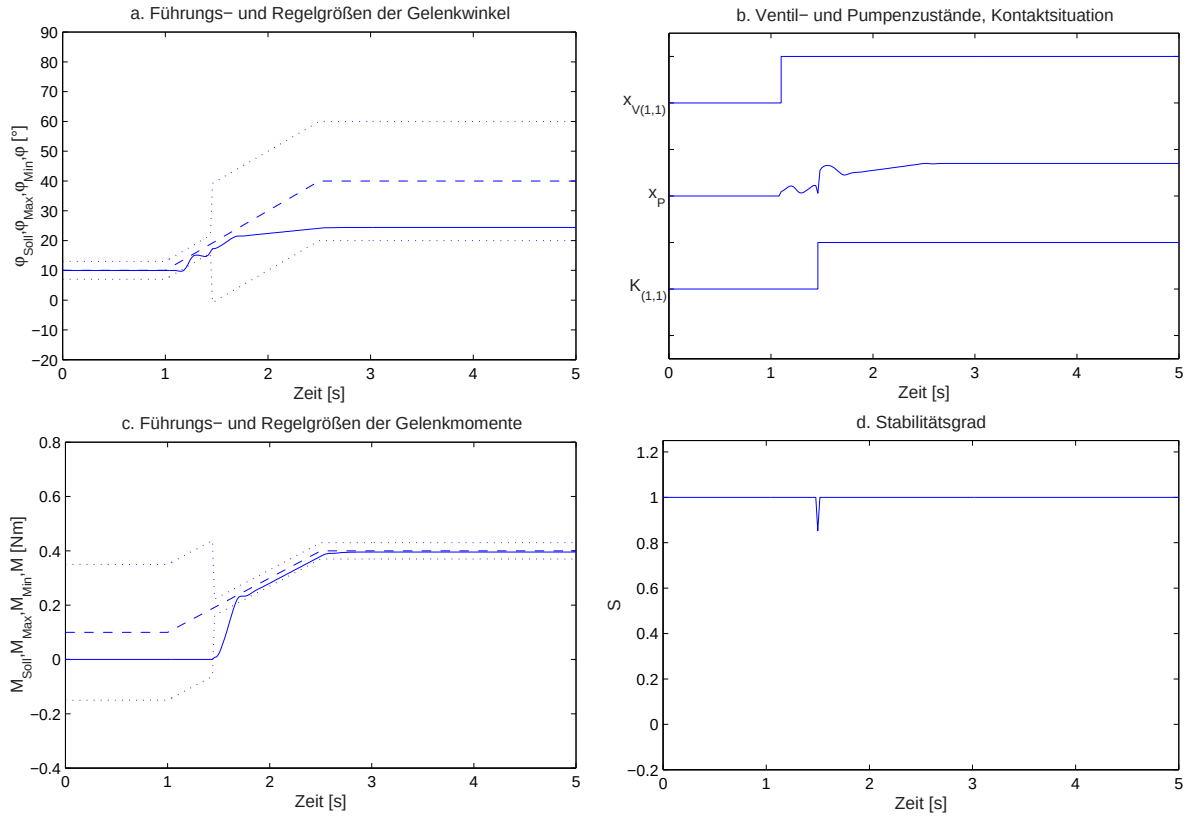


Bild 5.1: Führungs- und Regelgrößenverlauf der Gelenkwinkel (a), Ventil- und Pumpenzustände sowie Kontaktsituation (b), Führungs- und Regelgrößenverlauf der Gelenkmomente (c) sowie Stabilitätsgrad (d) über der Zeit (Szenario 1)

In Szenario 2 werden bei unveränderter Materialkonstante die Reglerparameter K_{RF} und T_{NF} verstellt. Als Folge der deutlich zu großen Reglerverstärkung $K_{RF} = 20$ und der zu kleinen Zeitkonstante $T_{NF} = 0.08$ beginnt das System zum Zeitpunkt $t = 1.7$ s zu schwingen. Die sich durch das Reglerfehlverhalten ergebenden Abweichungen des Systems vom Sollverhalten zu den Zeitpunkten $t = 2.1$ s und $t = 2.4$ s werden von der Stabilitätsüberwachung erkannt und mit 'INSTABIL' beurteilt ($S = 0$). Ab dem Zeitpunkt $t = 2.5$ s ist der Regler in der Lage das System zu stabilisieren und der Stabilitätsgrad beträgt wieder $S = 1$ (Bild 5.2 d). Wichtig ist, dass hier der Fokus nur auf der Detektion des instabilen Systemverhaltens liegt. Strategien zur Problemlösung werden in Abschnitt 5.3 vorgestellt.

Bei Szenario 3 wird bei sonst unveränderten Simulationsparametern die Materialkonstante auf $k_O = 5$ erhöht (Tabelle 5.1). Es ergibt sich bedingt durch das steifere Objekt ein größerer permanenter Positionsfehler als bei den Szenarien 1 und 2. Da der Regelungsalgorithmus im Kontaktfall nur das vorgegebene Moment $M_{Soll(i,j)}$ einregelt, bleibt die Regelgröße $\varphi_{(i,j)}$ jetzt außerhalb des vorgegeben Intervalls (Bild 5.3 a).

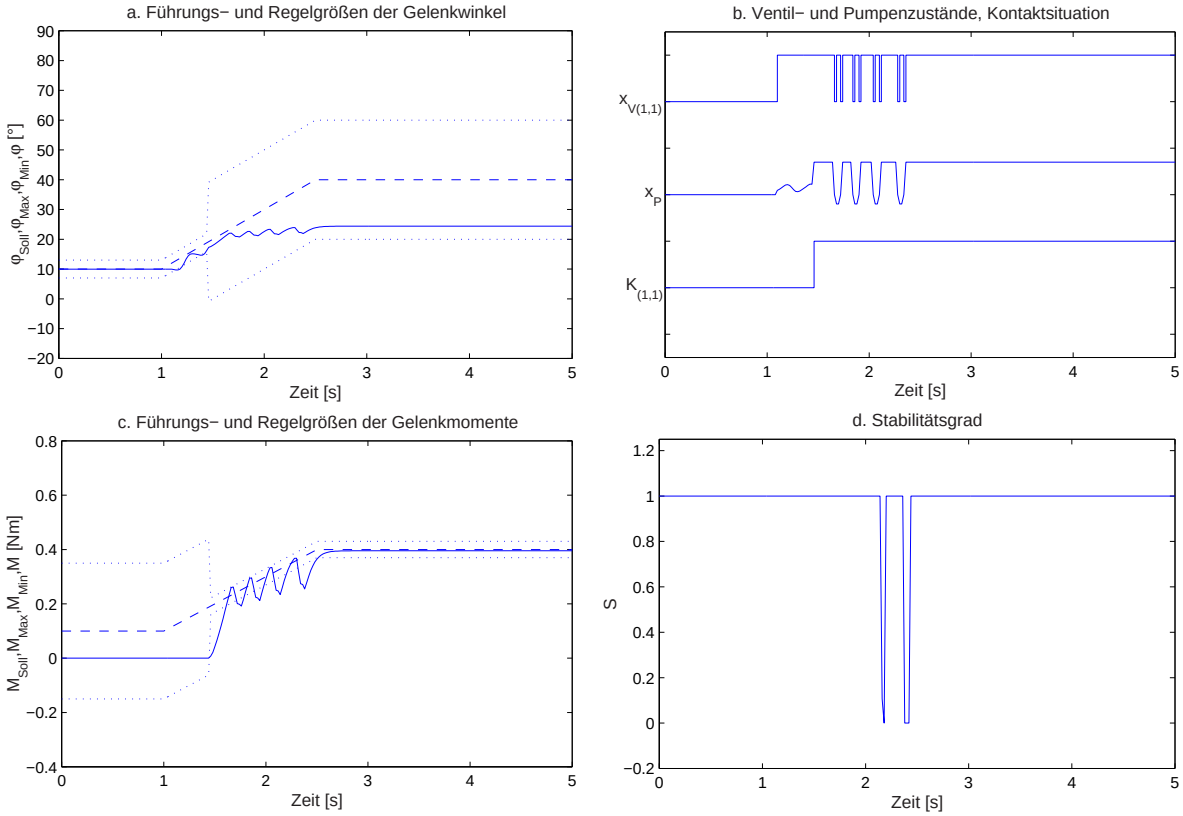


Bild 5.2: Führungs- und Regelgrößenverlauf der Gelenkwinkel (a), Ventil- und Pumpenzustände sowie Kontaktsituation (b), Führungs- und Regelgrößenverlauf der Gelenkmomente (c) sowie Stabilitätsgrad (d) über der Zeit (Szenario 2)

Als Folge der großen Objektsteifigkeit und der im Vergleich zu Szenario 2 noch schlechter an die Regelstrecke angepassten Reglerparameter, beginnt das System zu schwingen. Im Gegensatz zu Szenario 2 ist der Regler diesmal allerdings nicht in der Lage, das System wieder zu stabilisieren. Die Stabilitätsüberwachung detektiert dieses nicht zu tolerierende Systemverhalten mit periodisch wechselnden Stabilitätseinschätzungen zwischen 'STABIL' und 'INSTABIL'.

In Szenario 4 wird die Anzahl der Gelenke, bei sonst gleichen Simulationsparametern wie in Szenario 1, auf 3 erhöht (Tabelle 5.1). Bild 5.4 zeigt die Führungs- und Regelgrößenverläufe sowie die dazugehörenden Führungsgrößenintervalle in Abhängigkeit von der Kontaktsituation, den Pumpenzustand, das Steuersignal der Pumpe und die Ventilzustände sowie den berechneten Stabilitätsgrad über der Zeit.

Der Regler arbeitet bis zum Zeitpunkt $t = 1.8$ s als Positionsregler. An den Regelgrößenverläufen für die Gelenkwinkel $\varphi_{(i,j)}$ zwischen $t = 1.0$ s und $t = 1.8$ s ist deutlich die bereits in Abschnitt 4.2.3 diskutierte Problematik bei der Realisierung von gegenläufigen Fingergelenkbewegungen im hydraulischen Betrieb zu erkennen, da der Regler sich nur für eine Pumpenrichtung und damit für ein entsprechendes pulsweitenmoduliertes Steuersignal pro Abtastintervall entscheiden kann. Aus diesem Grund ist der Regler nur wechselseitig in der Lage, Streck- und Beugebewegungen zu realisieren. Ab dem Zeitpunkt $t = 1.8$ s ergibt sich Objektkontakt. Der Regler schaltet vom positions- in den kraftgeregelten Modus und regelt die von den Führungsgrößen vorgegebenen Momente $M_{Soll(i,j)}$ ein (Bild 5.4 c).

Die Stabilitätsüberwachung detektiert das Verlassen der Führungsgrößenintervalle, bedingt durch die gegenläufige Führungsgrößenvorgabe ab $t = 1.5$ s, mit Stabilitätsgraden zwischen $S = 0.6$ und $S = 0.5$,

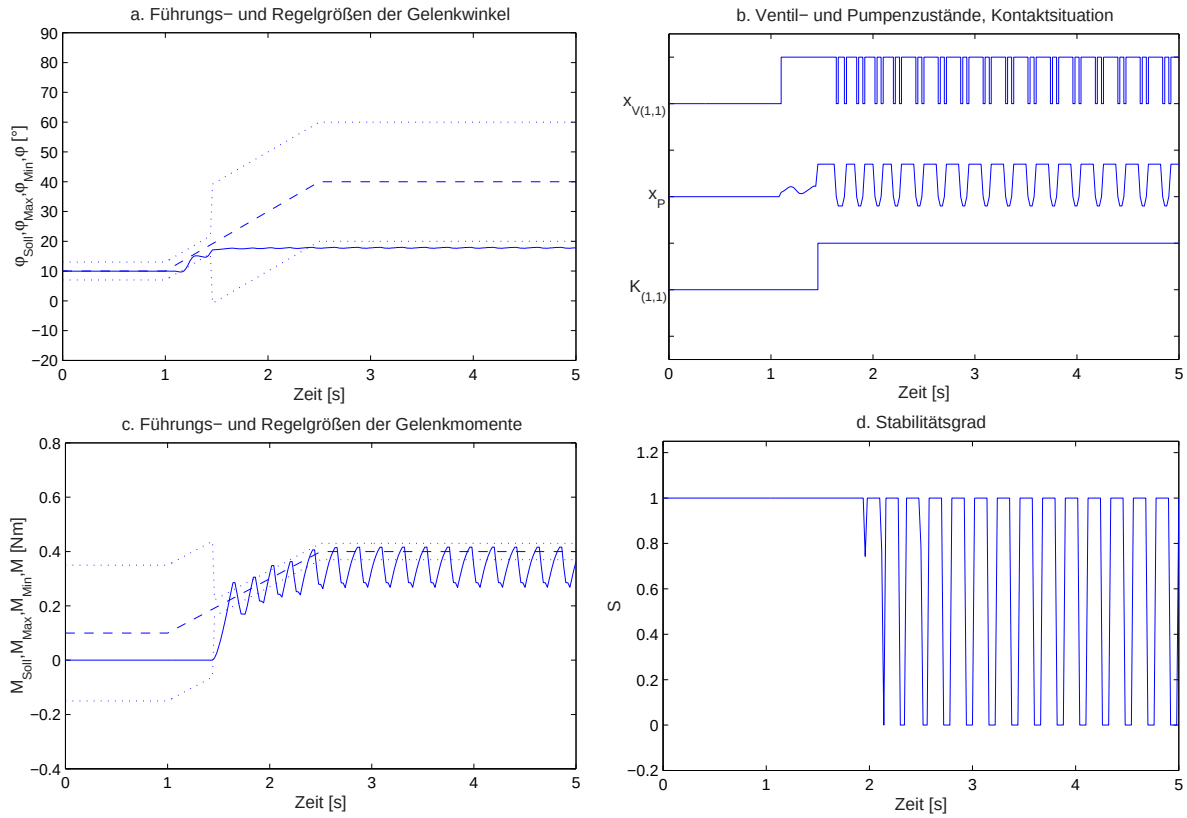


Bild 5.3: Führungs- und Regelgrößenverlauf der Gelenkwinkel (a), Ventil- und Pumpenzustände sowie Kontaktsituation (b), Führungs- und Regelgrößenverlauf der Gelenkmomente (c) sowie Stabilitätsgrad (d) über der Zeit (Szenario 3)

d. h. mit einer Stabilitätseinschätzung zwischen ’STABIL’ und ’NICHT ENTSCHEIDBAR’. Resultierend aus dem Objektkontakt, ergeben sich zwischen den Zeitpunkten $t = 2.0$ s und $t = 2.5$ s sowie $t = 4.2$ s und $t = 4.7$ s vom Regler nicht auszuregelnde Abweichungen von den Gelenkwinkel-Sollvorgaben $\varphi_{Soll(i,j)}$, da der Regelungsalgorithmus im Kontaktfall nur das vorgegebene Moment $M_{Soll(i,j)}$ einregelt. Die Stabilitätsüberwachung detektiert das nicht zu tolerierende Systemverhalten zu diesen Zeiten mit Stabilitätsgraden von $S = 0$, d. h. mit der Stabilitätseinschätzung ’INSTABIL’ (Bild 5.4 d) und erkennt somit die nicht zu realisierenden Vorgaben der Bahnplanung.

Um das Überwachungskonzept experimentell zu validieren, wurden die Komponenten der Stabilitätsüberwachung (Abschnitt 2.3.1.3) mit dem Portierungskonzept aus Abschnitt 3.4 auf die Roboterplattform [72, 199] des Fraunhofer Instituts IITB (Karlsruhe) transferiert. Die Plattform besteht aus einem Roboterkopf und zwei Armen (AMTEC-Module), wobei nur ein Arm für das hier betrachtete Szenario benötigt wird. Der Kopf verfügt über zwei und jeder Arm über sieben Bewegungsfreiheitsgrade (DOF). Als Endeffektor dient entweder ein Lichtschnittsensor oder der pneumatisch angetriebene Mehrfinger-Robotergreifer aus Abschnitt 4.2. Als Sensoren stehen dem System zur Verfügung:

- ein Kraft-Momenten-Sensor in jedem Handgelenk,
- ein Lichtschnittsensor am Greifer,
- eine Kamera im Ballen des Robotergreifers,
- eine Stereokamera im Roboterkopf sowie
- ein Audioarray im Roboterkopf [72].

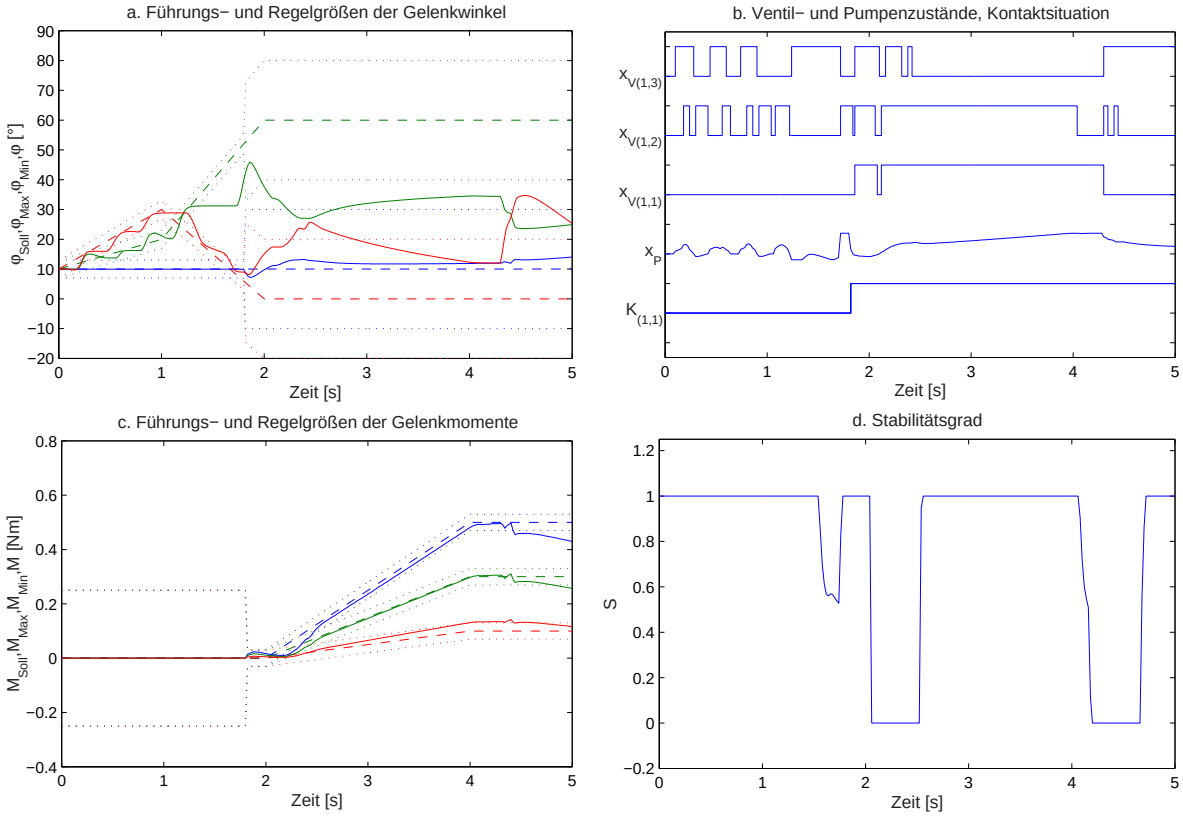


Bild 5.4: Führungs- und Regelgrößenverlauf der Gelenkwinkel (a), Ventil- und Pumpenzustände sowie Kontaktsituation (b), Führungs- und Regelgrößenverlauf der Gelenkmomente (c) sowie Stabilitätsgrad (d) über der Zeit (Szenario 4)

Bild 5.5 zeigt den Aufbau der Versuchs- und Entwicklungsplattform am Fraunhofer Institut IITB.

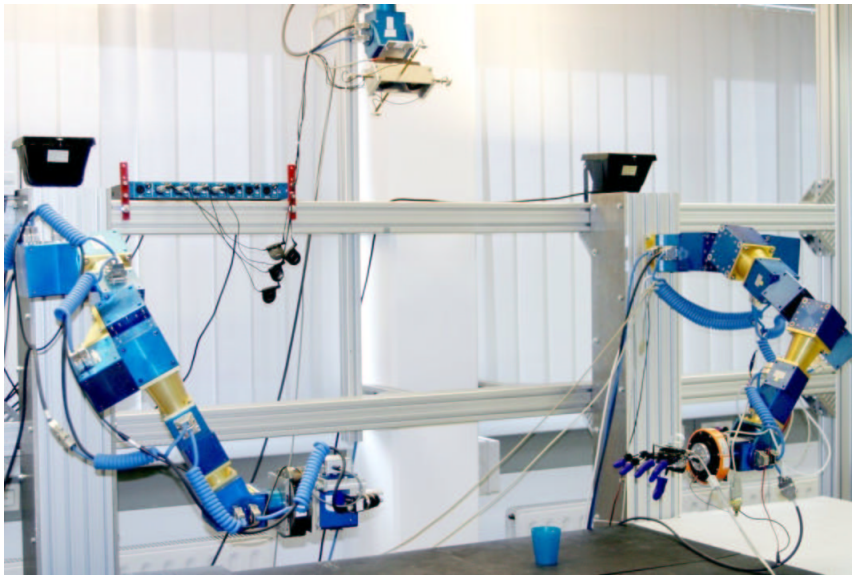


Bild 5.5: Aufbau der Versuchs- und Entwicklungsplattform am Fraunhofer Institut IITB [72]

Die Steuerung des Robotersystems erfolgt mit MCA (Abschnitt 3.4). Als Testszenario werden Bewegungsvorgänge mit einem Arm der Roboterplattform durchgeführt. Auftretende Abweichungen vom Sollverhalten werden dabei mit Hilfe der Stabilitätsüberwachung detektiert. Bild 5.6 zeigt die Führungs- (gestrichelt) und Regelgrößenverläufe (durchgezogen) der Gelenkwinkel (a) sowie den Verlauf des Stabilitätsgrads (b) über der Zeit für das experimentell validierte Szenario.

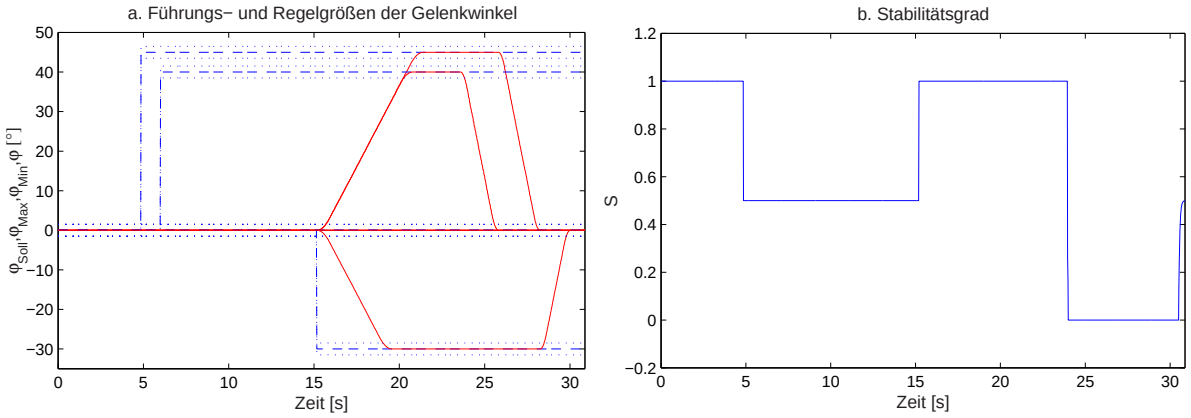


Bild 5.6: Führungs- (gestrichelt) und Regelgrößenverläufe (durchgezogen) der Gelenkwinkel (a) sowie den Verlauf des Stabilitätsgrads (b) über der Zeit für das experimentell validierte Szenario am Fraunhofer IITB

Die Vorgabe der Führungsgrößen erfolgt über ein Bedienerinterface (MCA-GUI). Gut zu erkennen ist die sprunghafte Änderung der Führungsgröße für den dritten ($\varphi_{Soll(Max,3)} = 45°$) und den fünften ($\varphi_{Soll(Max,5)} = 40°$) Freiheitsgrad des Roboterarms zu den Zeitpunkten $t = 4.8$ s und $t = 6$ s (Bild 5.6 a). Da die koordinierte Ausführung der Bewegungsaufgabe in diesem Beispiel erst nach der vollständigen Definition aller Führungsgrößen (hier sechster Freiheitsgrad ($\varphi_{Soll(Max,6)} = -30°$) zum Zeitpunkt $t = 15.1$ s) gestartet wird, ergeben sich zwischen $t = 4.8$ s und $t = 15.1$ s Abweichungen von den geplanten Führungsgrößen. Die Stabilitätsüberwachung bewertet dieses vom Benutzer verursachte jedoch nicht zu tolerierende Systemverhalten mit einem Stabilitätsgrad von $S = 0.5$. Zum Simulationszeitpunkt $t = 15.1$ s erfolgt der koordinierte Start der Bewegung (Benutzerkommando). Die Regelgrößen werden von den internen Gelenkreglern des AMTEC-Systems auf die vorgegebenen Führungsgrößen eingeregelt (Bild 5.6 a). Da das System jetzt in der Lage ist, für alle Freiheitsgrade eine Annäherung an den Sollbereich zu gewährleisten, erfolgt eine positive Bewertung der Gesamtsituation durch die Stabilitätsüberwachung (Stabilitätsgrad $S = 1$ ab dem Zeitpunkt $t = 15.2$ s, Bild 5.6 b). Zum Zeitpunkt $t = 23.6$ s erfolgt ein weiteres Benutzerkommando, das den Roboterarm veranlasst, alle Freiheitsgrade sequentiell in die Initialposition zurückzufahren (Bild 5.6 a). Dies erfolgt über eine von der Firma AMTEC mitgelieferte Funktion (’Home’), ohne dabei die entsprechenden Führungsgrößen über das Bedienerinterface zu verändern ($\varphi_{Soll(Max,j)} = const.$ für alle $j = 1..7$ Freiheitsgrade des Roboterarms, Bild 5.6 a). Da sich die Regelgrößen nun vom vorgegebenen Sollbereich entfernen, wird das Systemverhalten ab dem Zeitpunkt $t = 23.9$ s von der Stabilitätsüberwachung mit ’INSTABIL’ beurteilt (Stabilitätsgrad $S = 0$, Bild 5.6 b). Erst nachdem alle Freiheitsgrade ihre Initialposition ($\varphi_{Soll(Init,j)} = 0°$ für $j = 1..7$) erreicht haben und keine weitere Entfernung vom Sollbereich stattfindet (Zeitpunkt $t = 30.0$ s, Bild 5.6 a), steigt der Stabilitätsgrad wieder auf $S = 0.5$ (Bild 5.6 b). Um Schäden an der teuren Roboterhardware auszuschließen, wurden in diesem Beispiel bewusst keine Instabilitäten durch falsch angepasste Reglerparameter provoziert (vgl. Szenarien 1-4).

Fazit: Die Stabilitätsüberwachung ist in der Lage, eine zusammengefasste Bewertung der Situation der

Basisregler vorzunehmen. Solange sich die Basisregler im Sollbereich (innerhalb der Führungsgrößenintervalle) befinden oder für eine Annäherung an den Sollbereich sorgen, erfolgt eine positive, andernfalls eine negative Beurteilung der Gesamtsituation. Mit den auf diese Weise generierten Bewertungen ist das Überwachungssystem dann in der Lage, modellbasiert Fehler zu detektieren. So kann als Kriterium, ob ein nicht zu tolerierendes Systemverhalten vorliegt, beispielsweise ein periodisches Schwingen des Stabilitätsgrads (Anzahl an Maxima $S = 1$ und Minima $S = 0$ des Stabilitätsgrads in einem bestimmten Zeitintervall, vgl. Szenario 3) oder ein Stabilitätsgrad < 0.5 für eine vorgegebene Zeit t_{max} (vgl. Szenario 4)) verwendet werden. Die modellbasiert detektierten Fehler werden dann in die regelbasierte Auswertung aus Abschnitt 2.3.2.3 weitergeleitet und aktivieren je nach Schweregrad die Transitionen Stopped T_{St} bzw. Cancelled T_{Ca} im jeweiligen Verwaltungsnetz. Daraufhin kann dann die geeignete Problemlösungsstrategie durch das Robotersystem oder den Benutzer bzw. den Teleoperator ausgewählt werden (Abschnitt 5.3).

5.2.2 Hardwareüberwachung

Die Funktionalität des in Abschnitt 2.3.1.4 vorgeschlagenen Konzepts wird in diesem Abschnitt anhand eines Annäherungsvorgangs des hydraulisch angetriebenen flexiblen Robotergreifers aus Abschnitt 4.2 mit einer anschließenden Kontaktphase an das zu greifende Objekt demonstriert. Bei dem hier umgesetzten Beispiel handelt es sich um einen Ausschnitt (’koordiniertes Annähern’, ’Zugreifen’ und ’Halten’) des in Abschnitt 4.2.3 dargestellten Präzisionsgriffs. Bild 4.14 (unten Mitte) zeigt die Umsetzung des entsprechenden Koordinierungsnetzes. Tabelle 4.4 erläutert alle zu diesem Netz gehörenden Stellen und Transitionen. Die Umsetzung der rampenförmigen Führungsgrößentrajektorien erfolgt wiederum durch den hybriden Kraft-Positions-Regelungsalgorithmus aus Abschnitt 4.2.2. Der hydraulisch angetriebene Robotergreifer und das zu greifende Objekt werden mit den im Abschnitt 4.2.1 vorgestellten Modellen simuliert.

In diesem Anwendungsfall werden zwei verschiedene Fehlerarten betrachtet:

- Fehlerart 1: Defekt im hydraulischen Antriebssystem
  Ein defekte Pumpe zeichnet sich dadurch aus, dass unabhängig vom Steuersignal des Reglers, der Fördervolumenstrom der Pumpe Null ist. Bei einem Ventildefekt verbleibt das Ventil, ab dem Auftreten des Defekts, in seiner derzeitigen Position. Damit ergeben sich zwei zu unterscheidende Fehlerfälle. Der Defekt ereignet sich im geöffneten Zustand (Fall a.), dann kann das Ventil vom Regelungsalgorithmus nicht mehr geschlossen werden. Erfolgt der Defekt im geschlossenen Zustand, dann kann das Ventil vom Regelungsalgorithmus nicht mehr geöffnet werden (Fall b.). In beiden Fällen ist eine Regelung des Systems nicht mehr möglich.
- Fehlerart 2: Leckage im Fluidaktor bzw. in den Fluidleitungen
  Tritt ein Leckageverlust ein, so ist entweder der flexible Fluidaktor oder eine der Leitungen des hydraulischen Systems undicht. Dies führt in beiden Fällen zu einem Druckverlust im hydraulischen System.

Zur Detektion dieser Fehlerarten wird die in Abschnitt 2.3.1.4 vorgeschlagene Hardwareüberwachung in Form eines regelbasierten Auswertungsmoduls verwendet. Die beiden wesentlichen Vorteile des hier verwendeten Konzepts sind die geringe Komplexität und die leichte Implementierbarkeit. So benötigt die regelbasierte Auswertung der Hardwareüberwachung für dieses Beispiel nur eine Regel:

R_1 :	WENN	Steuerspannung Pumpe Max	ODER	Min	UND	Ventil i offen
	UND	kein Kontakt	UND	keine Gelenkbewegung	DANN	Hardwarefehler

Durch die Definition von weiteren Regeln kann dabei die Funktionalität der Hardwareüberwachung

gesteigert werden. Denkbar sind beispielsweise Regeln, die den erkannten Hardwarefehlern die entsprechende Fehlerart zuordnen. Durch die Modularität des Gesamtkonzepts ist das hier verwendete quantitativ modellgestützte Verfahren zudem jederzeit durch komplexere analytische [80, 94, 283, 286] und/oder heuristische [29, 95, 96, 105, 164] Verfahren zu erweitern oder zu ersetzen.

Die Bilder 5.7-5.8 zeigen als Ergebnis jeweils die Verläufe der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) für die Gelenkwinkel (a) und die Gelenkmomente (b), Ventil- und Pumpen- (c) sowie Fehlerzustände (d), den Fördervolumenstrom der Pumpe (e), die Durchflüsse durch die Ventile (f), die Leckage-Volumenströme (g) und die Fluidvolumina in den Aktoren (h) über der Zeit für die beiden untersuchten Fehlerarten.

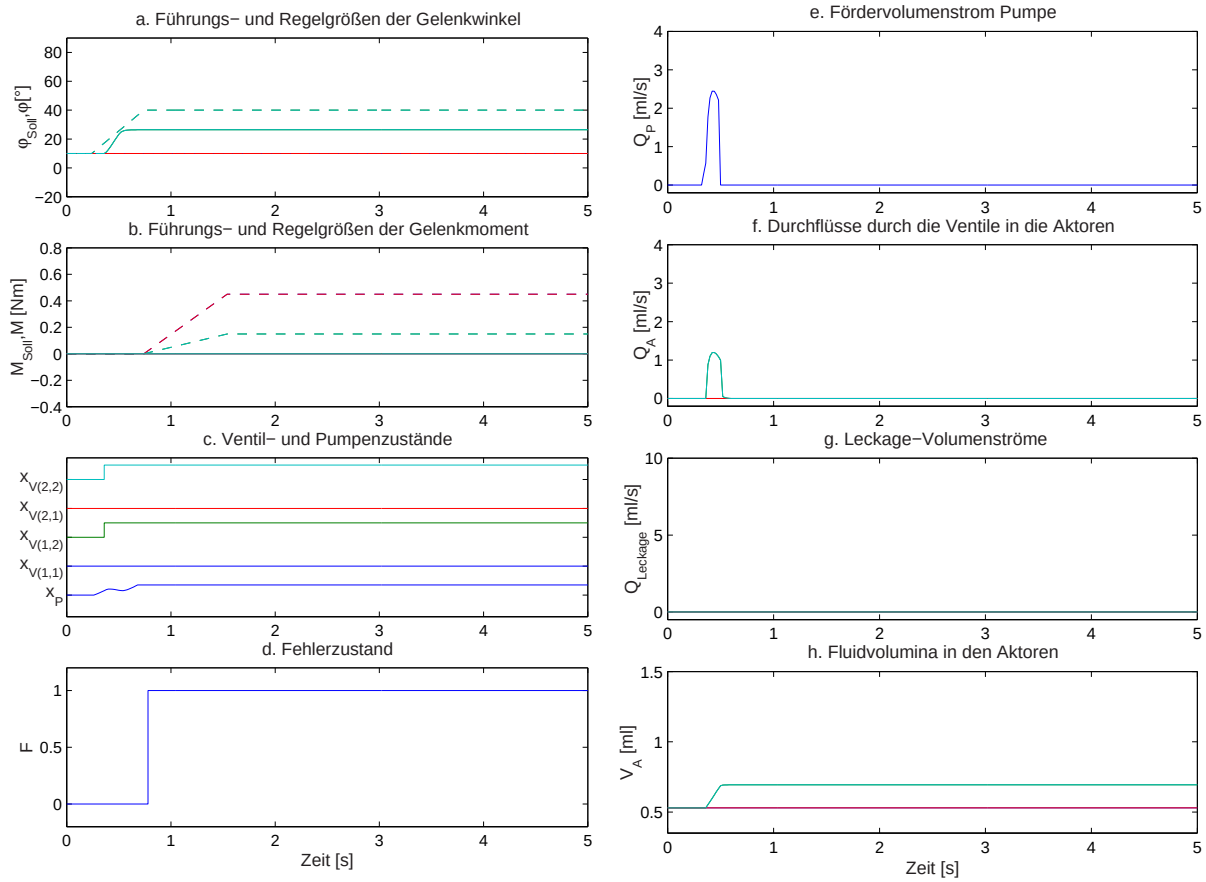


Bild 5.7: Ergebnisse der Hardwareüberwachung für die Fehlerart 1 (Pumpendefekt)

Bei der Fehlerart 1 tritt ab dem Simulationszeitpunkt $t = 0.5$ s ein simulierter Pumpendefekt auf (Fördervolumenstrom der Pumpe wird Null, vgl. Bild 5.7 e). Der implementierte Regelungsalgorithmus versucht die sich ergebende Regeldifferenz auszuregeln. Dafür werden bei den hier vorliegenden positiven Regeldifferenzen die entsprechenden Ventile geöffnet und die Pumpe wird vorwärts angesteuert (c). Da sich aufgrund der defekten Pumpe kein Fördervolumenstrom ergibt, sind auch die resultierenden Ventildurchflüsse gleich Null (f). Damit bleiben die Fluidvolumina in den Aktoren (h) sowie die Regelgrößen für die Gelenkwinkel (a) im Weiteren konstant. Das regelbasierte Auswertungsmodul erkennt den auftretenden Pumpendefekt in diesem Beispiel mit einer Zeitverzögerung von 0.3 s zum Simulationszeitpunkt $t = 0.8$ s (d).

Bei der Fehlerart 2 tritt ab dem Simulationszeitpunkt $t = 0.5$ s ein simulierter Leckagedefekt im zweiten

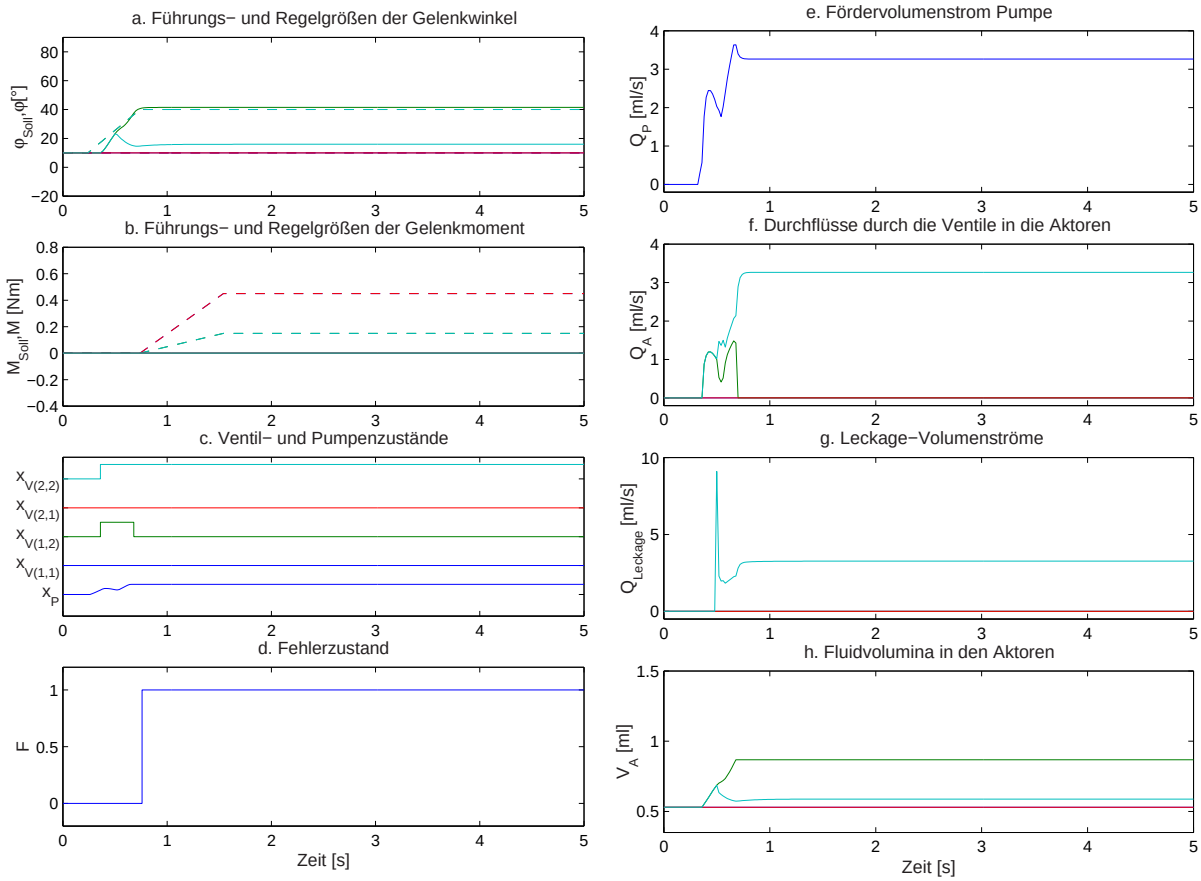


Bild 5.8: Ergebnisse der Hardwareüberwachung für die Fehlerart 2 (Leckage)

Aktor des Zeigefingers auf. Die simulative Umsetzung dieses Fehlers erfolgt durch einen zum Aktordruck proportionalen Leckage-Volumenstrom aus dem Aktor heraus (g). Dieser Leckage-Volumenstrom führt zu einem kleiner werdenden Fluidvolumen in dem entsprechenden Aktor (h) und einer daraus resultierenden Streckbewegung dieses Gelenkes (a). Damit ist der Regler, trotz Ansteuerung der Pumpe mit maximaler Steuerspannung (c), nicht in der Lage, die resultierende Regeldifferenz zu minimieren. Zum Simulationszeitpunkt $t = 1.0$ s hat sich ein Druckgleichgewicht zwischen ausströmendem (Leckage, g) und einströmendem Fluid (Pumpe, e) eingestellt und das entsprechende Gelenk erreicht seine Gleichgewichtslage (a) unabhängig von den vom Regelungsalgorithmus vorgegebenen Steuerspannungen für die Pumpe und Ventile (c). Das regelbasierte Auswertungsmodul erkennt den auftretenden Pumpendefekt in diesem Beispiel mit einer Zeitverzögerung von 0.2 s zum Simulationszeitpunkt $t = 0.7$ s (d).

Fazit: Neben den durch die Stabilitätsüberwachung modellbasiert erkannten Problemen der unterlagerten Basis-Regelkreise, können mit der hier präsentierten, einfach zu implementierenden, regelbasierten Hardwareüberwachung auftretende Hardwaredefekte frühzeitig online erkannt werden, um so weitere Schäden an der oftmals teuren Roboterhardware zu verhindern. Die Hardwareüberwachung vervollständigt somit die vorgeschlagene Grundstruktur des in Abschnitt 2.3 vorgestellten Gesamtkonzepts. Probleme ergeben sich bei der hier verwendeten einfachen Regelbasis beispielsweise bei Defekten, die erst während einer Greifbewegung auftreten. Da dort Objektkontakt vorliegt, kann aus einer aktiver Pumpe und einem offenem Ventil nicht eindeutig auf eine Bewegung des bzw. der entsprechenden Gelenke geschlossen werden. Aus diesem Grund ist die hier implementierte Hardwareüberwachung nicht in der Lage, solche Fehler zu erkennen und korrekt zuzuordnen. Derlei Probleme können beispielsweise durch

die Definition von weiteren Regeln oder dem Einsatz von zusätzlichen analytischen [80, 94, 283, 286] und/oder heuristischen [29, 95, 96, 105, 164] Verfahren vermieden werden. Die durch die Hardwareüberwachung detektierten Fehler werden dann in die regelbasierte Auswertung auf der mittleren Ebene der Roboter-Steuerungsarchitektur weitergeleitet (Abschnitt 2.3.2.3), um dort die routinemäßige Durchführung der Aufgabe zu unterbrechen und somit den Start einer geeigneten Problemlösungsstrategie zu ermöglichen (Aktivierung der Transitionen Cancelled T_{Ca} im jeweiligen Verwaltungsnetz der Aufgabe, Beispiele folgen in Abschnitt 5.3).

5.3 Petrinetzbasierte Problemlösung

In diesem Abschnitt erfolgt die erstmalige Umsetzung von bewussten und unbewussten Problemlösungsstrategien in Form von Elementaraktionen mit dem neuen ganzheitlichen Überwachungskonzept. Insgesamt werden sieben verschiedene Szenarien vorgestellt und die erzielten Ergebnisse diskutiert. Bei den untersuchten Szenarien handelt es sich um reine Bewegungsvorgänge wie z. B. Zeigegesten des Robotergreifers oder Kopfschütteln der Halseinheit, aber auch komplexe Handlungsabläufe wie Greifbewegungen oder bildbasierte Personen- und Objektverfolgungsvorgänge. Zur Fehlerdetektion werden, neben den modellbasierten Komponenten zur Stabilitäts- und Hardwareüberwachung, nicht planbare Benutzereingriffe, Zeitüberschreitungen (Time Outs) in Plätzen sowie unerwartete Kollisionen verwendet. Hier dargestellte Strategien zur Problemlösung sind das Ignorieren (’Ignore-Mode’), das langsam Fahren (’Safety-Mode’), das zur Initial-Pose zurückkehren (’Initial-Mode’), das reflexbasierte Ausweichen (’Reflex-Mode’) sowie das irreversible Abbrechen (’Cancelled-Mode’). Tabelle 5.2 vergleicht die untersuchten Szenarien anhand von eintretendem Ereignis, der verwendeten Problemlösungsstrategie, der ursprünglich ausgeführten Aktionsfolge AF sowie der dazu verwendeten Roboterkomponente.

Szenario	eintretendes Ereignis	Strategie	Aktionsfolge AF	Roboterkomponente
1	Benutzerunterbrechung	Ignore-Mode	Zeigegeste	Greifer
2	Benutzerunterbrechung	Safety-Mode	Zeigegeste	Greifer
3	Benutzerunterbrechung	Initial-Mode	Kopfgeste	Halseinheit
4	Kollision	Reflex-Mode	Kopfgesten	Halseinheit
5	Time Out	Cancelled-Mode	Aufmerksamkeits-steuerung	Halseinheit
6	Hardwaredefekt	Cancelled-Mode	Aufmerksamkeits-steuerung	Halseinheit
7	Stabilitätsproblem	Safety-Mode	Greifvorgang	Greifer

Tabelle 5.2: Vergleich der analysierten Szenarien

Die Bilder 5.9-5.15 zeigen als Ergebnis jeweils die Verläufe der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) für die Gelenkwinkel (a) sowie die Steuerspannungen der Motoren (Szenarien 3-6) bzw. die Steuerspannungen Ventile und der Pumpe (Szenarien 1, 2 und 7) (b) über der Zeit. Die Tabellen 5.3-5.6 zeigen jeweils den Zeitpunkt des in Abhängigkeit vom dargestellten Szenario eintretenden Ereignisses sowie die zugehörigen Zeitpunkte der von der Steuerungsarchitektur bzw. dem Benutzer aktivierten Transitionen (T_{Ac}, T_{St}, T_{Ca}) des Verwaltungsnetzes für alle sieben untersuchten Szenarien.

Bei dem ersten beiden Szenarien handelt es sich um die Ausführung von Zeigegestenbewegungen mit dem Robotergreifer. Die Umsetzung dieser Aufgaben erfolgt durch eine Instanz der Aktionsfolge ’Fah-

ren'. Als Führungsgrößentrajektorien werden Rampen und zur Berechnung der Regeldifferenz wird der Admittanz-Regelungsalgorithmus aus Abschnitt 4.2.2 eingesetzt. Der Robotergreifer wird in beiden Szenarien mit den in Abschnitt 4.2.1 vorgestellten hydraulischen Modellen simuliert.

Szenario	Zeitpunkt eintretendes Ereignis	Zeitpunkte aktivierter Transitionen		
		T_{St}	T_{Ac}	T_{Ca}
1	204.0 s	204.1 s	206.1 s	-
2	207.7 s	207.8 s	209.0 s	-

Tabelle 5.3: Zeitpunkte eintretender Ereignisses und aktivierter Transitionen (Szenarien 1, 2)

Zum Simulationszeitpunkt $t = 204.0$ s erfolgt bei Szenario 1 ein nicht im Voraus planbarer Eingriff in die routinemäßige Ausführung der Aktionsfolge durch den Benutzer (Tabelle 5.3). Mögliche Ursachen hierfür sind beispielsweise das frühzeitige Erkennen einer Gefahrensituation durch den Menschen oder ein spontaner Meinungswechsel des Benutzers, der zu einer Planänderung für das Robotersystem führt. Die verspätete Reaktion des Reglers auf die Führungsgrößenvorgaben (Bild 5.9 a) ist in diesem Beispiel darauf zurückzuführen, dass zu Beginn der Aktionsfolge ein zu geringes Druckniveau in der Ventilbank herrscht. Erst wenn die Pumpe einen höheren Druck aufgebaut hat, als die beiden in diesem Beispiel angesteuerten Aktoren in ihren Ruhelagen aufweisen, werden die entsprechenden Ventile durch den Regelungsalgorithmus (Gleichung (4.36)) geöffnet (Bild 5.9 b, Simulationszeitpunkt $t = 203.9$ s). Dadurch wird gewährleistet, dass vor dem Öffnen der Ventile die richtigen Druckverhältnisse in der Ventilbank und den Aktoren vorliegen. Nach dem Benutzereingriff wird die Aktionsfolge zum Simulationszeitpunkt $t = 204.1$ s von der Roboter-Steuerungsarchitektur autonom unterbrochen (Tabelle 5.3). Zum Simulationszeitpunkt $t = 206.1$ s erfolgt die Fortsetzung der Aktionsfolge durch den Benutzer (Tabelle 5.3). Als Problemlösungsstrategie wird in diesem Beispiel 'Problem ignorieren' verwendet ('Ignore-Mode', Tabelle 5.2). Damit wird die Aktionsfolge mit den routinemäßigen Parametern fortgesetzt und zum Simulationszeitpunkt $t = 207.4$ s erfolgreich beendet (Bild 5.9 a).

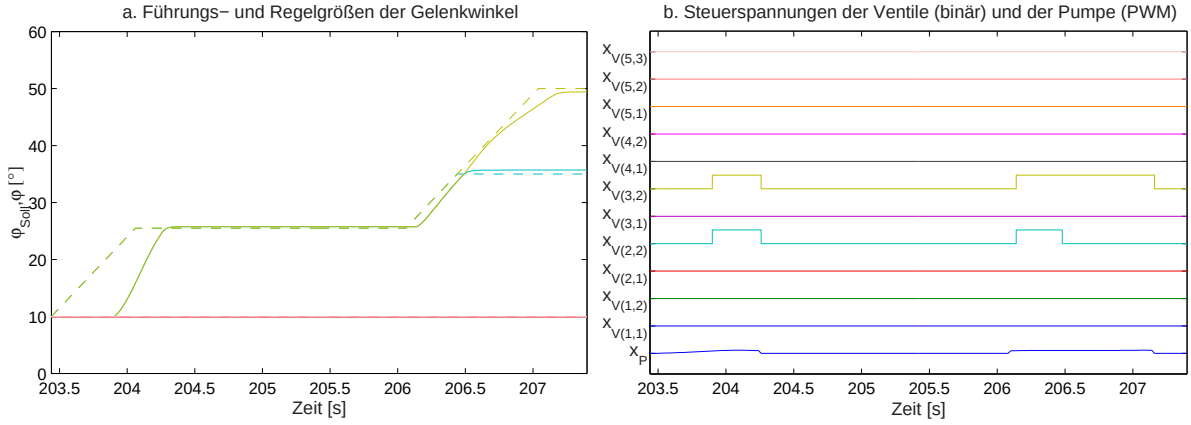


Bild 5.9: Simulationsergebnisse für Szenario 1

Bei Szenario 2 erfolgt zum Simulationszeitpunkt $t = 207.7$ s wiederum ein nicht im Voraus planbarer Eingriff in die routinemäßige Ausführung der Aktionsfolge durch den Benutzer (Tabelle 5.3). Nach dem Benutzereingriff wird die Aktionsfolge zum Simulationszeitpunkt $t = 207.8$ s von der Überwachung der Roboter-Steuerungsarchitektur autonom unterbrochen. Zum Simulationszeitpunkt $t = 209.0$ s erfolgt die Fortsetzung der Aktionsfolge durch den Benutzer. Als Problemlösungsstrategie wird in Szenario 2

allerdings ’fahre langsam weiter’ verwendet (’Safety-Mode’, Tabelle 5.2). Dabei wird die Aktionsfolge mit adaptierten Parametern für die Steigung der rampenförmigen Führungsgrößentrajektorien und angepassten Reglerparametern fortgesetzt und zum Simulationszeitpunkt $t = 212.5$ s erfolgreich beendet.

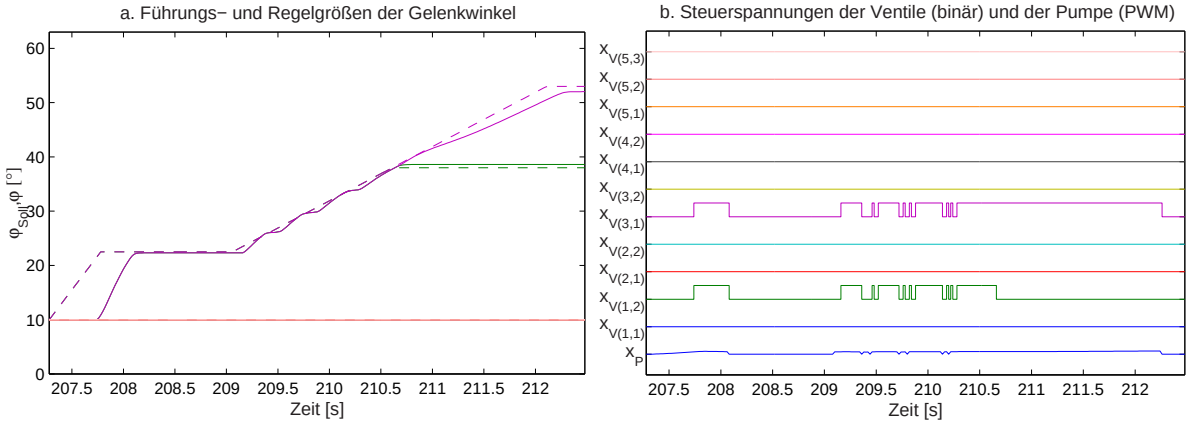


Bild 5.10: Simulationsergebnisse für Szenario 2

Die Werte der adaptierten Parameter und damit die resultierende Problemlösungsstrategie müssen dabei vom Entwickler bereits bei der Implementierung der Strategie festgelegt werden. Zur Realisierung der geforderten langsamen Bewegungsausführung hat es als praktikabel erwiesen, die Steigungen der rampenförmigen Führungsgrößentrajektorien um den Faktor zehn und die Parameter des Admittanz-Regelungsalgorithmus um den Faktor fünf im Vergleich zu Szenario 1 zu verkleinern (Abschnitt 4.2.2).

Bei den Szenarien 3 und 4 handelt es sich um die Ausführung von Kopfgesten (’Nicken’) mit der anthropomorphen Halseinheit des Robotersystems. Die Umsetzung dieser Aufgaben erfolgt wiederum durch eine Instanz der Aktionsfolge ’Fahren’. Als Führungsgrößentrajektorien werden Rampen und zur Berechnung der Regeldifferenz wird der Positions-Regelungsalgorithmus aus Abschnitt 4.3.2 eingesetzt. Die Roboterhalseinheit wird in beiden Szenarien mit den im Abschnitt 4.3.1 vorgestellten Modellen simuliert.

Szenario	Zeitpunkt eintretendes Ereignis	Zeitpunkte aktivierter Transitionen		
		T_{St}	T_{Ac}	T_{Ca}
3	206.8 s	206.9 s	210.7 s	-
4	170.7 s	-	-	-

Tabelle 5.4: Zeitpunkte eintretender Ereignisses und aktivierter Transitionen (Szenarien 3, 4)

Zum Simulationszeitpunkt $t = 206.8$ s erfolgt bei Szenario 3 wiederum ein nicht im Voraus planbarer Eingriff in die routinemäßige Ausführung der Aktionsfolge durch den Benutzer (Tabelle 5.4). Daraufhin wird die Aktionsfolge zum Simulationszeitpunkt $t = 206.9$ s autonom von der Überwachung der Roboter-Steuerungsarchitektur unterbrochen. Die Führungsgrößentrajektorien bleiben daraufhin im Weiteren konstant (Bild 5.11 a). Zum Simulationszeitpunkt $t = 210.7$ s erfolgt die Fortsetzung der Aktionsfolge durch den Benutzer (Tabelle 5.3). Als Problemlösungsstrategie wird in Szenario 3 zur ’Initial-Pose zurückkehren’ verwendet (’Initial-Mode’, Tabelle 5.2). Dabei werden die rampenförmigen Führungsgrößentrajektorien so adaptiert, dass die entsprechende Roboterkomponente in ihre ursprüng-

liche Initial-Pose zurückbewegt wird. Die Reglerparameter werden dabei nicht verändert. Zum Simulationszeitpunkt $t = 211.1$ s erreicht die Halseinheit in diesem Beispiel die initiale Position (Bild 5.9 a, hier Nulllage), woraufhin die Aktionsfolge zum Simulationszeitpunkt $t = 211.3$ s erfolgreich beendet wird.

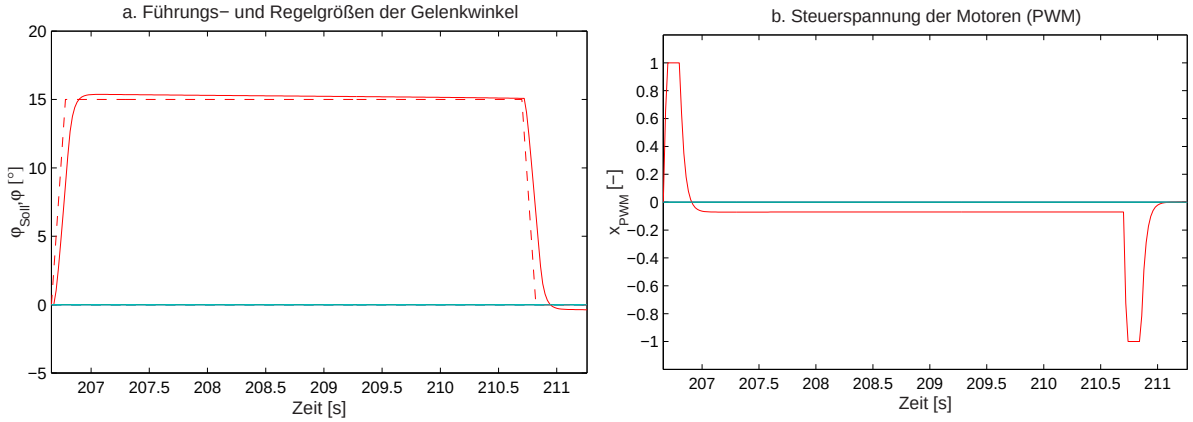


Bild 5.11: Simulationsergebnisse für Szenario 3

Bei Szenario 4 wird der Einsatz einer im Vergleich zu den ersten drei Szenarien grundlegend anderen Problemlösungsstrategie vorgestellt. So handelt es sich bei der durchgeführten Aktionsfolge zwar prinzipiell wieder um eine Nickbewegung der anthropomorphen Halseinheit, allerdings erfolgt zum Simulationszeitpunkt $t = 170.7$ s eine unerwartete Kollision mit einem externen Objekt (Tabelle 5.4). Die Kontaktdetektion erfolgt wiederum modellbasiert oder durch entsprechende externe Sensorik. Beim Auftreten dieses Ereignisses initiiert die Überwachung der Roboter-Steuerungsarchitektur ohne vorangehende Unterbrechung der Aufgabe eine reflexbasierte Ausweichbewegung (’Reflex-Mode’, Tabelle 5.2). Da es sich in dem hier gezeigten Beispiel um eine aktive Kollision handelt, wird eine Ausweichbewegung entgegen der ursprünglichen Bewegungsrichtung ausgeführt (Bild 5.12 a).

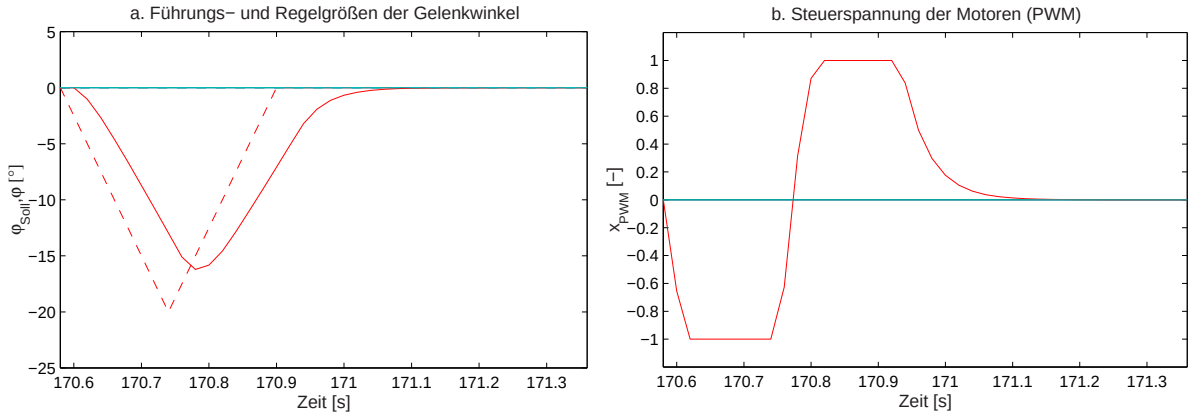


Bild 5.12: Simulationsergebnisse für Szenario 4

Die hier dargestellte Reflexbewegung ist somit prinzipiell eine von der Planungskomponente des Robotersystems nicht bewusst zu steuernde, intuitive und schnelle Ausweichbewegung. Dies bedeutet einen fundamentalen Unterschied zu den bisher beschriebenen Problemlösungsstrategien, bei denen eine eingehende Situationsbewertung durch die Aufgabenplanung oder den Benutzer der letztendlichen Problemlösung vorangeht.

Bei den Szenarien 5 und 6 handelt es sich um die Ausführung einer 'Aufmerksamkeitssteuerung' mit der anthropomorphen Halseinheit des Robotersystems. Die Umsetzung dieser Aufgaben erfolgt durch Instanzen der Aktionsfolgen 'Suchen' und 'Verfolgen' (Abschnitt 4.3.3). Als Führungsgrößentrajektorien werden Rampen und zur Berechnung der Regeldifferenz wird der bildbasierte Regelungsalgorithmus aus Abschnitt 4.3.2 eingesetzt. Die Roboterhalseinheit wird in beiden Szenarien wiederum mit den im Abschnitt 4.3.1 vorgestellten Modellen simuliert.

Szenario	Zeitpunkt eintretendes Ereignis	Zeitpunkte aktivierter Transitionen		
		T_{St}	T_{Ac}	T_{Ca}
5	98.8 s	-	-	99.0 s
6	69.0 s	69.1 s	-	70.0 s

Tabelle 5.5: Zeitpunkte eintretender Ereignisses und aktivierter Transitionen (Szenarien 5, 6)

Nach dem Start von Szenario 5 befindet sich dass zu verfolgende Objekt zunächst außerhalb des Sichtfeldes des Stereo-Kamerasystems. Das Robotersystem startet daraufhin einen Suchvorgang. Dazu werden alternierende, rampenförmige Führungsgrößentrajektorien für den dritten Freiheitsgrad (Drehen) der Roboterhalseinheit generiert und vom Regler rein positionsgeregelt umgesetzt. Zum Simulationszeitpunkt $t = 98.7$ s hat die Roboterhalseinheit dabei die konstruktiv vorgegebene maximale negative Auslenkung (Linksdrehung) für den dritten Freiheitsgrad $\varphi_{Soll,Min(3)} = -90°$ erreicht (Bild 5.13 a). Danach wird die Suchrichtung umgedreht und die Halseinheit dreht sich nach rechts. Ab dem Simulationszeitpunkt $t = 98.8$ s tritt ein simulierter Defekt bei der Antriebseinheit des dritten Freiheitsgrads auf. Daraufhin bleibt die Regelgröße im Weiteren trotz maximaler Ansteuerung des Motors durch den Regler (Bild 5.13 b) konstant (Bild 5.13 a). Zum Simulationszeitpunkt $t = 99.0$ s, d. h. mit einer Zeitverzögerung von $t = 0.2$ s erkennt das regelbasierten Auswertungsmodul der Überwachung den auftretenden Motordefekt und bricht daraufhin die Ausführung der Aktionsfolge autonom ab (Tabelle 5.5). Die hierzu implementierte Regel lautet:

R_1 :	WENN	Steuerspannung Motor Max	ODER	Min
	UND	keine Gelenkbewegung für $t > t_{Min}$	DANN	Hardwarefehler

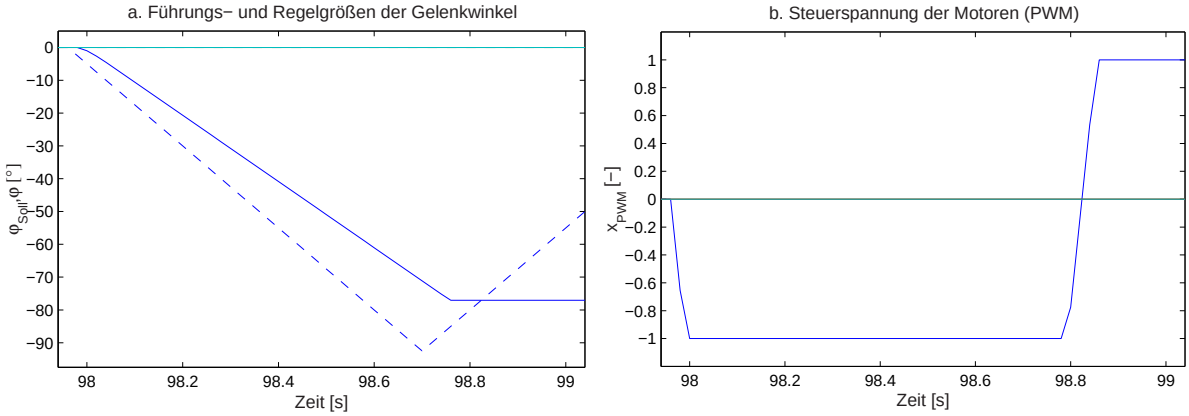


Bild 5.13: Simulationsergebnisse für Szenario 5

Bei Szenario 6 befindet sich das zu verfolgende Objekt zunächst wiederum außerhalb des Sichtfeldes des Stereo-Kamerasystems. Das Robotersystem startet demzufolge einen Suchvorgang mit alternierenden, rampenförmigen Führungsgrößentrajektorien für den dritten Freiheitsgrad (Drehen). Nachdem die

Halseinheit zu den Simulationszeitpunkten $t = 67.7$ s und $t = 69.0$ s die beiden konstruktiv vorgegebenen maximalen negativen und positiven Auslenkungen erreicht hat (Bild 5.14 a) und sich immer noch kein Objekt im Sichtfeld des Stereo-Kamerasystems befindet, ergibt sich eine Zeitüberschreitung der entsprechenden Marke im zugehörenden Koordinierungsnetz.

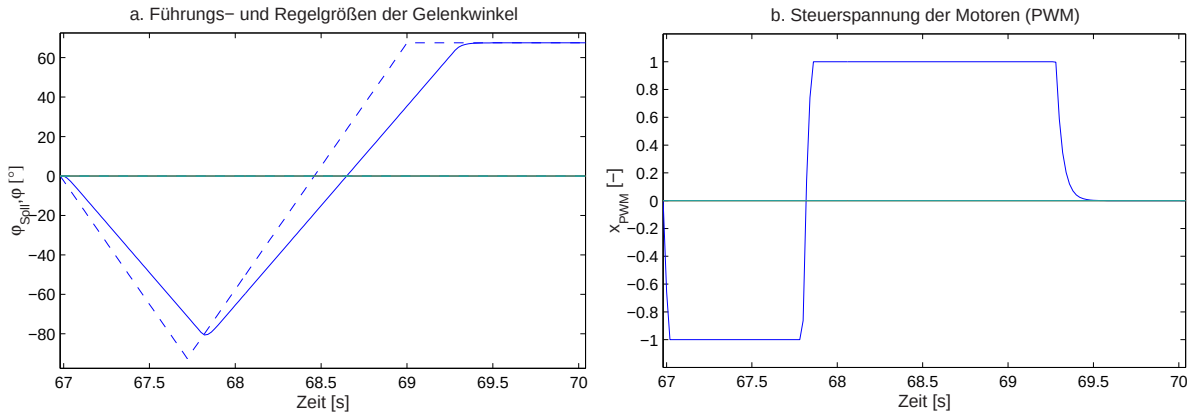


Bild 5.14: Simulationsergebnisse für Szenario 6

Daraufhin aktiviert das regelbasierte Auswertungsmodul der Überwachung die Stopped-Transition T_{St} im Verwaltungsnetz und unterbricht die Ausführung der Aktionsfolge (Tabelle 5.5). Da sich in diesem Fall kein der vorgegebenen Spezifikation entsprechendes Objekt finden lässt, bricht das Robotersystem die Ausführung der Aktionsfolge zum Simulationszeitpunkt $t = 70.0$ s irreversibel ab (Tabelle 5.5).

Bei dem letzten hier vorgestellten Szenario handelt es sich um die Ausführung einer Greifbewegung mit dem flexiblen Robotergreifer aus Abschnitt 4.2.1. Für die erfolgreiche Ausführung dieser Aktionsfolge sind nach Abschnitt 4.2.3 die Elementaraktionen ’Fahren’, ’Zugreifen’ und ’Halten’ und ’Warten’ [174] zu einem Koordinierungsnetz zusammenzufassen. Als Führungsgrößentrajektorien werden Rampen und zur Berechnung der Regeldifferenz wird der Admittanz-Regelungsalgorithmus aus Abschnitt 4.2.2 eingesetzt. Der Robotergreifer und das zu greifende Objekt werden wiederum mit den im Abschnitt 4.2.1 vorgestellten Modellen simuliert.

Szenario	Zeitpunkt eintretendes Ereignis	Zeitpunkte aktivierter Transitionen		
		T_{St}	T_{Ac}	T_{Ca}
7	$81.5 - 82.0$ s	82.0 s	84.0 s	-

Tabelle 5.6: Zeitpunkte eintretender Ereignisses und aktivierter Transitionen (Szenario 7)

In Szenario 7 werden die Objektparameter des dritten Szenarios aus Abschnitt 5.2.1 zur Durchführung der Greifbewegung verwendet. Als Folge der schlecht an die Regelstrecke angepassten Reglerparameter beginnt das System ab dem Simulationszeitpunkt $t = 81.5$ s zu schwingen (Bild 5.15 a). Die modellbasierte Stabilitätsüberwachung detektiert dieses nicht zu tolerierende Systemverhalten mit periodisch wechselnden Stabilitätseinschätzungen zwischen ’STABIL’ und ’INSTABIL’. Aufgrund dieser Stabilitätsbeurteilung unterbricht das regelbasierte Auswertungsmodul der Überwachung die geplante Durchführung der Aktionsfolge zum Simulationszeitpunkt $t = 82.0$ s autonom (Tabelle 5.6). Als Kriterium dient die Anzahl (hier größer vier) an absoluten Maxima und Minima des Stabilitätsgrads im Zeitintervall zwischen $t = 81.5$ s und $t = 82.0$ s.

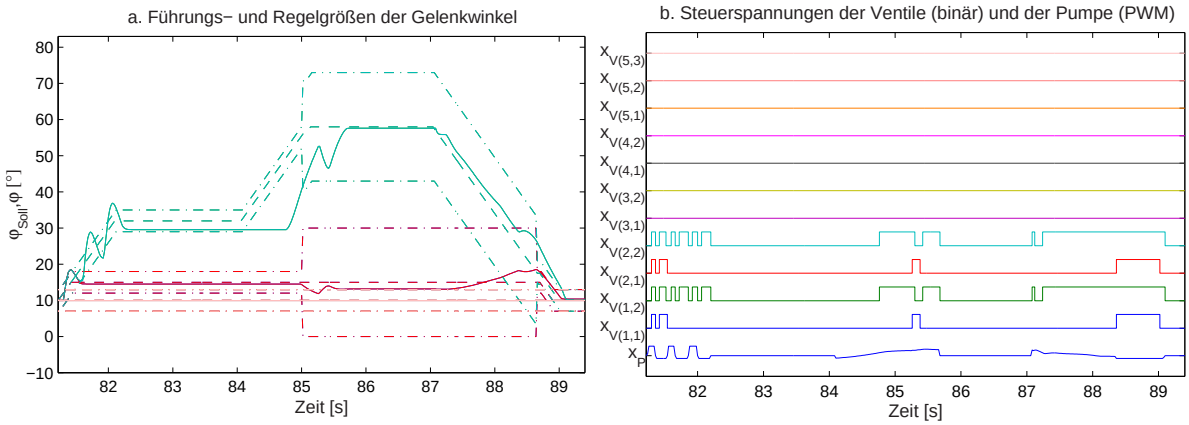


Bild 5.15: Simulationsergebnisse für Szenario 7

Als Problemlösungsstrategie wird in diesem Beispiel wie in Szenario 2 'fahre langsam' weiter verwendet ('Safety-Mode', Tabelle 5.2). Dabei wird die Aktionsfolge zum Simulationszeitpunkt $t = 84.0$ s (Tabelle 5.6) mit adaptierten Reglerparametern fortgesetzt (Bild 5.15 a). Um die geforderte langsamere und damit stabilisierend wirkende Bewegungsausführung zu realisieren, hat es sich als praktikabel erwiesen, die Parameter des Admittanz-Regelungsalgorithmus um den Faktor zehn zu verkleinern. Mit den nun besser an die Strecke angepassten Reglerparametern ist das System in der Lage, die Greifbewegung ohne weitere Systemschwingungen fortzusetzen (Bild 5.15 a) und zum Simulationszeitpunkt $t = 89.4$ s erfolgreich zu beenden.

Fazit: In diesem Abschnitt wurden erstmals die Komponenten der petrinetzbasierten Problemlösung mit den bereits vorgestellten Komponenten der modellbasierten Fehlerdetektion verknüpft. Zur petrinetzbasierten Problemlösung wurden bewusste und unbewusste Problemlösungsstrategien in Form von Elementaraktionen eingesetzt. Die Ergebnisse wurden anhand von sieben verschiedenen Simulationsszenarien diskutiert. Bei den untersuchten Szenarien handelt es sich dabei sowohl um reine Bewegungsvorgänge wie z. B. Zeigegesten des Robotergreifers oder Kopfschütteln der Halseinheit, als auch um komplexe Handlungsabläufe wie Greifbewegungen oder bildbasierte Personen- und Objektverfolgungsvorgänge.

5.4 Prioritätsverwaltung

Zum Abschluss des Kapitels werden Beispielszenarien präsentiert, die zeigen, dass die Hardwareressourcen des Roboters mit in die Planung und Ausführung von Aufgaben integriert werden müssen. Treten bei der Planung Fehler (z. B. Ressourcenkonflikte) auf, muss die Steuerungsarchitektur in der Lage sein, entsprechend zu reagieren. Dies erfolgt durch die autonome Prioritätsverwaltung (Abschnitt 2.2). Neben der Planung, Durchführung und Überwachung von Roboteraufgaben ist mit dem im Kapitel 2 vorgestellten Konzept auch die Prioritätsverwaltung von Aufgaben zur Laufzeit möglich. Beispiele sind die Zuordnung der im Aufgabenspeicher hinterlegten Aktionsfolgen zu unterschiedlichen Prioritätenlisten, die Blockierung von Aktionsfolgen bei Ressourcenkonflikten oder die Eliminierung einzelner Aktionsfolgen aus dem Aufgabenspeicher.

Die Funktionsweise des neuen Konzepts bei der Verwaltung von Hardwareressourcen wird in diesem Abschnitt anhand von drei Szenarien dargestellt. Dazu werden mehrere Aktionsfolgen zu einer Handlungssequenz zusammengefasst und in den Aufgabenspeicher auf der mittleren Ebene der Steuerungsarchitektur weitergeleitet. Wird die derzeit aktiv ausgeführte Aktionsfolge, z. B. durch Benutzereingriff

unterbrochen, dann initiiert die Aufgabenkoordinierung nur dann die Ausführung der nächsten im Aufgabenspeicher wartenden Aktionsfolge, falls die dazu notwendige Hardwareressourcen nicht durch eine andere Aktionsfolge aktiv oder passiv blockiert werden. Im dritten Szenario wird abschließend gezeigt, dass mit dem neuen Konzept bereits bestehende Handlungssequenzen auf den unteren Ebenen der Steuerungsarchitektur autonom manipuliert werden können. Dazu werden aus bereits im Aufgabenspeicher abgelegten Handlungssequenzen zur Laufzeit einzelne Aktionsfolgen eliminiert (Aktivierung von T_{Ca} im Verwaltungsnetz) und somit die gesamte Handlungssequenz zur Laufzeit verändert.

Die Bilder 5.16-5.18 zeigen als Ergebnis jeweils die Verläufe der Führungs- (gestrichelt) und Regelgrößen (durchgezogen) der Gelenkwinkel für den Greifer (a) und die Halseinheit (b) sowie die Platzbelegung der Plätze A4-A9 im Statusnetz (c) über der Zeit für alle in diesem Abschnitt dargestellten Szenarien.

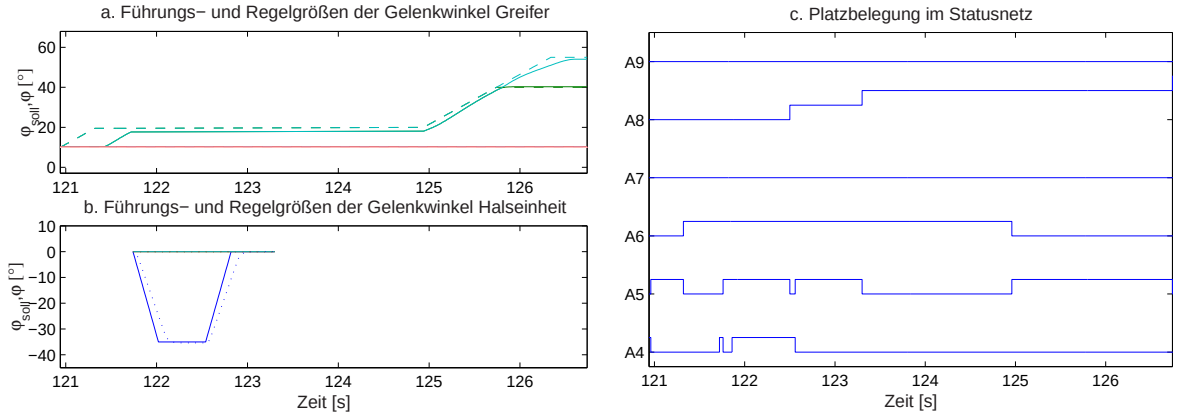


Bild 5.16: Simulationsergebnisse für Szenario 1

In Szenario 1 wird zum Simulationszeitpunkt $t = 120.1$ s die erste Aktionsfolge (’Zeigegeste’ des Robotergreifers, Bild 5.16 a) von der Aufgabenkoordinierung der Steuerungsarchitektur gestartet (Platz A5 mit einer Marke belegt, Bild 5.16 c) und zum Simulationszeitpunkt $t = 121.3$ s durch einen Benutzereingriff unterbrochen (Platz A5 frei und Platz A6 mit einer Marke belegt, Bild 5.16 c). Daraufhin bleiben die Führungs- und Regelgrößen dieser Aktionsfolge im Weiteren konstant (Bild 5.16 a). Zum Simulationszeitpunkt $t = 121.7$ s wird eine weitere Aktionsfolge (’Kopfdrehung’ der Roboterhalseinheit) von der Aufgabenplanung der Steuerungsarchitektur in den Aufgabenspeicher weitergeleitet (Platz A4 mit einer Marke belegt, Bild 5.16 c). Da zu diesem Zeitpunkt der Platz A5 nicht belegt ist, und auch kein Ressourcenkonflikt durch die unterbrochene Aktionsfolge ’Zeigegeste’ besteht (verschiedene Roboterkomponenten), wird die Aktionsfolge ’Kopfdrehung’ von der Prioritätsverwaltung gestartet (Platz A4 frei und Platz A5 mit einer Marke belegt, Bild 5.16 c) und ausgeführt (Bild 5.16 b). Neben der hier vorgestellten Realisierung mit einer aktiv ausgeführten Aktionsfolge ermöglicht das Konzept zudem die parallele Ausführung mehrerer Aktionsfolgen (allerdings immer unter Berücksichtigung der entsprechenden Ressourcen). Dazu ist die Kapazität des Platzes A5 entsprechend den Anforderungen zu erhöhen. Die Ausführung der Aktionsfolge ’Zeigegeste’ ist dabei weiterhin unterbrochen (Bild 5.16 a). Nach der erfolgreichen Beendigung der Aktionsfolge ’Kopfdrehung’ zum Simulationszeitpunkt $t = 122.5$ s (Platz A8 mit einer Marke belegt, Bild 5.16 c) wird die dritte bereits im Aufgabenspeicher wartende Aktionsfolge (wiederum ’Kopfdrehung’ allerdings diesmal in die andere Richtung) ausgeführt (Bild 5.16 b) und zum Simulationszeitpunkt $t = 123.3$ s erfolgreich beendet (Platz A8 mit zwei Marken belegt, Bild 5.16 c). Ab $t = 125.0$ s wird die unterbrochene Aktionsfolge ’Zeigegeste’ durch den Benutzer im ’Ignore-Mode’, d. h. mit unveränderten Parametern fortgesetzt (Platz A6 frei und Platz A5 mit einer Marke belegt, Bild 5.16 c) und zum Simulationszeitpunkt $t = 126.7$ s erfolgreich beendet (Platz A8 mit drei Marken belegt, Bild 5.16 c).

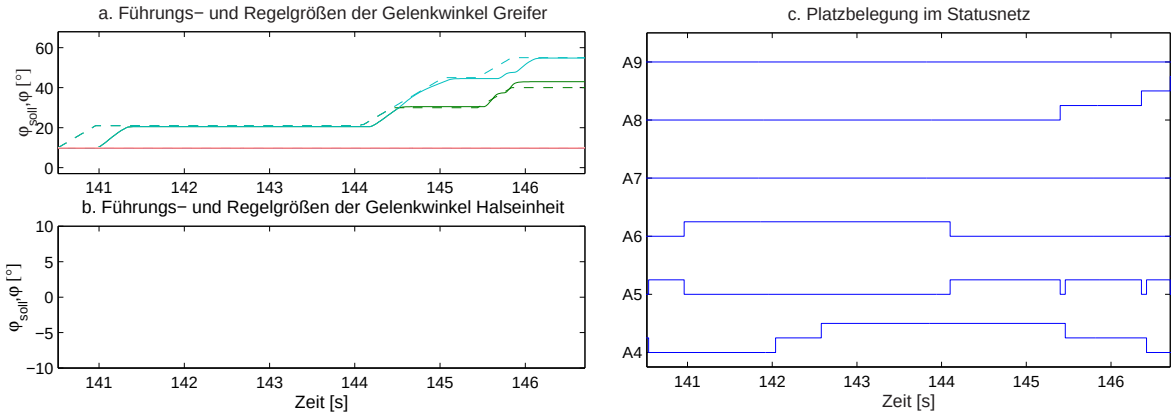


Bild 5.17: Simulationsergebnisse für Szenario 2

Bei Szenario 2 erfolgt zunächst wiederum die Durchführung einer ’Zeigegeste’ des Robotergreifers, die zum Simulationszeitpunkt $t = 141.0$ s durch den Bediener unterbrochen wird (Bilder 5.17 a, c). Zu den Simulationszeitpunkten $t = 142.0$ s bzw. $t = 142.6$ s werden zwei weitere Aktionsfolgen (im Gegensatz zu Szenario 1 wiederum ’Zeigegesten’ des Robotergreifers) von der Aufgabenplanung der Steuerungsarchitektur in den Aufgabenspeicher weitergeleitet (Platz A4 zunächst mit einer Marke, dann mit zwei Marken belegt, Bild 5.17 c). Obwohl zu diesen Zeitpunkten der Platz A5 nicht belegt ist (die erste Aktionsfolge ist unterbrochen, d. h. lediglich der Platz A6 ist belegt), werden weder die zweite noch die dritte Aktionsfolge gestartet, da in diesem Fall ein Ressourcenkonflikt durch die passiv blockierte Ressource des Robotergreifers im Stopped-Platz des Verwaltungsnetzes besteht (dieselben Roboterkomponenten werden zur Ausführung der verschiedenen Aktionsfolgen benötigt). Erst nachdem zum Simulationszeitpunkt $t = 144.1$ s die erste Aktionsfolge durch Benutzereingriff fortgesetzt (Platz A6 frei und Platz A5 mit einer Marke belegt, Bild 5.17 c) und zum Simulationszeitpunkt $t = 145.4$ s erfolgreich beendet wurde (Platz A8 mit einer Marke belegt, Bild 5.16 c), ist die entsprechende Ressource (hier der Greifer) frei und die zweite und dritte im Aufgabenspeicher wartenden Aktionsfolgen können sequentiell ausgeführt werden (Bilder 5.17 a, c).

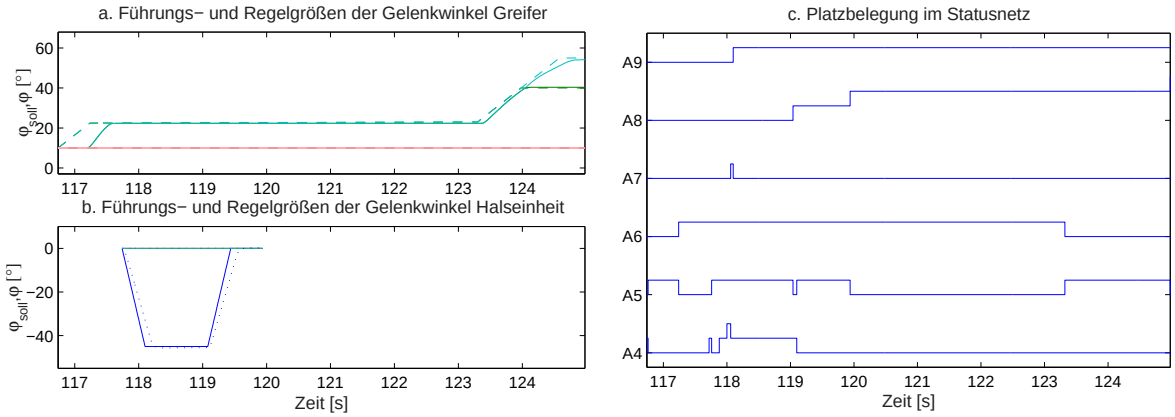


Bild 5.18: Simulationsergebnisse für Szenario 3

In Szenario 3 wird nach der Unterbrechung der ersten Aktionsfolge (’Zeigegeste’ des Robotergreifers) wie in Szenario 1 die zweite im Aufgabenspeicher wartende Aktionsfolge aktiviert (Platz A5 mit einer Marken belegt, Bild 5.18 c), da es sich um eine ’Kopfdrehung’ der Halseinheit des Robotersystems

handelt, und damit kein Ressourcenkonflikt vorliegt (vgl. Szenario 1). Danach werden zwei weitere Aktionsfolgen in den Aufgabenspeicher geschrieben (Platz A4 mit zwei Marken belegt, Bild 5.18 c), von der die erste zum Simulationszeitpunkt $t = 118.0$ s durch Aktivierung der Transition T_{Ca} im Verwaltungsnetz aus dem Aufgabenspeicher eliminiert wird (Platz A4 und Platz A9 jeweils mit einer Marke belegt, Bild 5.17 c). Statt der ursprünglich geplanten dritten Aktionsfolge wird nach dem erfolgreichen Beenden der zweiten Aktionsfolge zum Zeitpunkt $t = 119.1$ s direkt die ursprünglich erst an vierter Stelle vorgesehene Aktionsfolge (eine entgegengesetzte 'Kopfdrehung') ausgeführt und erfolgreich beendet. Danach erfolgt die Fortsetzung und Beendigung der ersten Aktionsfolge ('Zeigegeste' des Robotergreifers) analog zu Szenario 1.

Fazit: Das in dieser Arbeit vorgeschlagene Gesamtkonzept stellt, zusätzlich zu den bisher vorgestellten Funktionalitäten zur Planung, Durchführung und Überwachung von Roboteraufgaben, eine funktionsfähige Realisierung einer Prioritätsverwaltung dar. Mit dem hier gezeigten Entwicklungsstand erfolgreich validierte Beispiele sind, die Zuordnung der im Aufgabenspeicher hinterlegten Aktionsfolgen zu unterschiedlichen Prioritätenlisten, die Blockierung von Aktionsfolgen bei Ressourcenkonflikten sowie die Eliminierung einzelner Aktionsfolgen aus dem Aufgabenspeicher zur Laufzeit.

5.5 Diskussion

In Abschnitt 5.2.1 wurde anhand von fünf repräsentativen Beispielszenarien sowohl simulativ als auch experimentell nachgewiesen, dass die Stabilitätsüberwachung in der Lage ist, eine zusammengefasste Bewertung der Situation der Basisregler vorzunehmen und damit zur modellbasierten Fehlerdetektion einzusetzen ist. Abschnitt 5.2.2 zeigt, wie auftretende Hardwaredefekte frühzeitig online auf einfache Weise erkannt werden können, um so weitere Schäden an der oftmals teuren Roboterhardware zu verhindern. Dies beinhaltet dabei sowohl die Aktorik als auch die Sensorik des überwachten Systems.

In Abschnitt 5.3 wurden erstmals die Komponenten der petrinetzbasierten Problemlösung erfolgreich mit den bereits vorgestellten Komponenten der modellbasierten Fehlerdetektion verknüpft. Zur petrinetzbasierten Problemlösung wurden bewusste und unbewusste Problemlösungsstrategien in Form von Elementaraktionen eingesetzt und die Ergebnisse anhand von Simulationsstudien evaluiert.

Zusätzlich zu den bisher vorgestellten Funktionalitäten zur Planung, Durchführung und Überwachung von Roboteraufgaben wurde in Abschnitt 5.4 weiterhin eine erste funktionsfähige Realisierung einer vereinfachten Prioritätsverwaltung vorgestellt und die Ergebnisse wiederum anhand von Simulationsstudien erfolgreich validiert.

6 Zusammenfassung

Eine der wichtigsten Anforderungen an ein modernes humanoides Robotersystem ist die Fähigkeit, Alltagsaufgaben kooperativ mit dem Menschen bewältigen zu können [76]. Diese Forderung geht deutlich über den heutigen Stand der Robotertechnik hinaus [175]. So erfolgt bei Industrierobotern aus sicherheitstechnischen Aspekten derzeit noch eine strikte Trennung zwischen den Arbeitsräumen von Roboter und Mensch [5, 73]. Der direkte Kontakt mit dem Menschen bedingt bei humanoiden Robotern allerdings ein leistungsfähiges Überwachungskonzept, das in der Lage ist, Gefahrensituationen schnell und flexibel zu erkennen, um gegebenenfalls geeignete Gegenmaßnahmen einzuleiten. Das in dieser Arbeit erstmals vorgestellte, modulare Konzept stellt einen solchen integrierten Lösungsvorschlag zur flexiblen Sicherheitsüberwachung von humanoiden Robotersystemen dar.

Die inhaltlichen Schwerpunkte dieser Arbeit umfassen

- die Entwicklung eines Verfahrens zur systematisierten einheitlichen Beschreibung von Handlungen für humanoide Roboter mit hierarchischen Petri-Netzen zur Generierung und Speicherung von Aufgabenwissen in der Datenbank des Robotersystems (Kapitel 2),
- die Bereitstellung und Integration flexibler und modularer modellbasierter Überwachungskomponenten auf allen Ebenen der Roboter-Steuerungsarchitektur. Dies beinhaltet ein Konzept zur einfachen und systematischen Behandlung von Ausnahmesituationen bei humanoiden Robotern durch eine spezielle Dekomposition von Überwachungsmodulen und deren Kopplung mit geplanten Routineabläufen (Kapitel 2),
- die rechentechnische Implementierung und Validierung des in Kapitel 2 vorgestellten Konzepts (Kapitel 3),
- die Evaluierung des petrinetzbasierten Konzepts zur Generierung von Aufgabenwissen am Beispiel von ausgewählten Roboterkomponenten. Dies beinhaltet die theoretische und experimentelle Modellbildung und den Entwurf modifizierter, modellbasierter Regelungskonzepte zur Beherrschung nichtlinearer Systemdynamiken (Kapitel 4) sowie
- die erstmalige Umsetzung des neuen flexiblen, petrinetzbasierten Überwachungskonzepts für humanoide Roboter anhand von Simulationsstudien und Experimenten (Kapitel 5).

Die wichtigsten Ergebnisse der Arbeit sind:

1. Spezifikation einer diskret-kontinuierlichen Systemstruktur zur Kopplung von kontinuierlichen Führungsgrößentrajektorien und Stellgrößen der Regelung auf der Ausführungsebene und ereignisdiskreten Handlungsabläufen auf der Planungsebene der Roboter-Steuerungsarchitektur.

2. Entwicklung einer Systematik zur Generierung von modellbasiertem Aufgabenwissen in Form von Elementaraktionen, deren Vorteile beim einheitlichen Entwurfsschema und den verfügbaren Schnittstellen liegen.

3. Erweiterung des Konzepts zur systematischen, einheitlichen Beschreibung von komplexen Roboterhandlungen mit hierarchisch strukturierten Petri-Netzen.

4. Bereitstellung eines modularen, petrinetzbasierten und online-fähigen Überwachungskonzepts. Dies beinhaltet die Verknüpfung der modellbasierten Überwachung der Aufgabenausführung auf der unteren Ebene mit einem neuen petrinetzbasierten Konzept zur einfachen und systematischen Behandlung von erkannten Ausnahmesituationen auf der mittleren Ebene der hierarchischen Roboter-Steuerungsarchitektur.

5. Integration des Gesamtkonzepts in die hybride Steuerungsarchitektur eines humanoiden Roboters. Nachweis der Funktionalität des Gesamtkonzepts anhand von Simulationen.

6. Erweiterung und Modifikation der modellbasierten Fuzzy-Stabilitätsüberwachung zur Verarbeitung gekoppelter Mehrgrößensysteme sowie Einführung von Führungsgrößenintervallen zur aufgabenspezifischen Priorisierung der Stabilitätsbeurteilung.

7. Erste Umsetzung eines intuitiv zu entwerfenden und leicht zu implementierenden Konzepts zur modellbasierten Überwachung der Hardware des humanoiden Robotersystems. Dies beinhaltet sowohl die Aktorik als auch die Sensorik des betrachteten Systems.

8. Entwurf eines neuen einheitlichen Konzepts zur Ausnahmesituationsbehandlung mit Petri-Netzen durch Gliederung in je ein Modul zur *Verwaltung* und zur *Auswertung* sowie die Zuweisung von zusätzlichen Funktionen für eine platzspezifische Ausnahmebehandlung.

9. Spezifikation und Umsetzung von bewussten und unbewussten Problemlösungsstrategien in Form der Elementaraktionen.

10. Entwurf und Validierung von Algorithmen zum Entwurf, zur Simulation und zur Analyse von Petri-Netzen auf der Entwicklungsplattform (MATLAB). Ableiten eines neuen Portierungskonzepts zur Portierung der entwickelten Algorithmen von der Entwicklungsplattform (MATLAB) auf die Zielplattform (C/C++ Framework Modular Controller Architecture (MCA)).

11. Evaluierung des Konzepts zur Generierung von Aufgabenwissen am Beispiel von ausgewählten Roboterkomponenten. Dies beinhaltet die theoretische und experimentelle Modellbildung und den modellbasierten Reglerentwurf für einen fluidisch betriebenen Mehrfinger-Robotergreifer und die anthropomorphe Halseinheit eines humanoiden Roboters.

12. Nachweis der Funktionalität des neuen Konzepts zur Generierung von Aufgabenwissen anhand von Simulationen und Experimenten. Nachweis der Funktionalität des flexiblen, petrinetzbasierten Überwachungskonzepts anhand von Simulationen und Experimenten.

Da derzeit erst einzelne Teilkomponenten des hier vorgestellten, integrierten Gesamtkonzepts experimentell erprobt sind, muss der Fokus zukünftiger Arbeiten auf der weiteren ausführlichen Umsetzung und Validierung des vorgeschlagenen Gesamtkonzepts auf realen Robotersystemen liegen. Dies beinhaltet u. A. die umfassende Erweiterung des derzeit existierenden petrinetzbasierten Aufgabenwissens sowie die Integration geeigneter Lernverfahren.

Potenzial für weiterführende methodische Entwicklungen bieten analytische [80, 94, 283, 286] und/oder heuristische [29, 95, 96, 105, 164] Verfahren zur modellbasierten Hardwareüberwachung. Weiterhin bieten entscheidungstheoretische [35, 37] und datenbasierte [195, 196] Ansätze eine interessante Möglichkeit, die in dieser Arbeit vom Entwickler entworfenen Überwachungsregeln autonom zu generieren.

A Petri-Netze

Ein Petri-Netz ist allgemein eine Methode, um Prozesse mit wertediskreten Zuständen aller Art abstrakt darzustellen. Definiert wird ein Petri-Netz durch ein 6-Tupel ($\mathcal{A}$, $\mathcal{T}$, $\mathcal{F}$, $\mathcal{K}$, $\mathcal{W}$, $\mathcal{M}_0$), dessen Bestandteile Plätze ($\mathcal{A}$) mit Kapazitäten ($\mathcal{K}$) und Markenbelegungen ($\mathcal{M}$) (gegeben anfängliche Markenbelegung $\mathcal{M}_0$), Transitionen ($\mathcal{T}$) und Kanten ($\mathcal{F}$) mit Kantengewichten ($\mathcal{W}$) sind [221]:

$$\mathcal{A} = \{A_1, A_2, \ldots, A_{|\mathcal{A}|}\} \tag{A.1}$$
$$\mathcal{T} = \{T_1, T_2, \ldots, T_{|\mathcal{T}|}\} \tag{A.2}$$
$$\mathcal{F} \subseteq (\mathcal{A} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{A}) \tag{A.3}$$
$$\mathcal{K} : \mathcal{A} \rightarrow \mathbb{N}\backslash\{0\} \tag{A.4}$$
$$\mathcal{W} : \mathcal{F} \rightarrow \mathbb{N}\backslash\{0\} \tag{A.5}$$
$$\mathcal{M}_0 : \mathcal{A} \rightarrow \mathbb{N} \tag{A.6}$$

Die Repräsentation eines Petri-Netzes erfolgt mit Hilfe grafischer Symbole. Dabei werden die Plätze als Kreise, die Transitionen als Rechtecke und die Kanten als Pfeile dargestellt (Bild A.1).

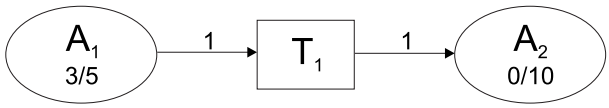


Bild A.1: Beispiel Petri-Netz mit zwei Plätzen und einer Transition

Bild A.1 repräsentiert folgendes Petri-Netz:

$$\mathcal{A} = \{A_1, A_2\} \tag{A.7}$$
$$\mathcal{T} = \{T_1\} \tag{A.8}$$
$$\mathcal{F} = \{(A_1, T_1), (T_1, A_2)\} \tag{A.9}$$
$$\mathcal{K} = \{(A_1, 5), (A_2, 10)\} \tag{A.10}$$
$$\mathcal{W} = \{((A_1, T_1), 1), ((T_1, A_2), 1)\} \tag{A.11}$$
$$\mathcal{M}_0 = \{(A_1, 2), (A_2, 0)\} \tag{A.12}$$

Um den Zusammenhang zwischen zu beschreibendem Prozess und entsprechender Petri-Netz-Repräsentation herzustellen, werden bei steuerungstechnisch interpretierten Petri-Netzen auftretende Ereignisse als Transitionen, Bedingungen als Plätze und die Verknüpfung von Ereignissen und Bedingungen als Kanten modelliert. Weiterhin werden Regeln benötigt, die den Prozessfortschritt im Netz beschreiben. Dieser Vorgang wird durch das 'Schalten' nachgebildet. Das Schalten verändert die Markenbelegung im Netz. Nur eine Transition kann zeitgleich schalten. Dabei werden so viele Marken von jedem

Platz des Vorbereichs wie das Kantengewicht angibt an die Plätze des Nachbereichs übergeben [221]. Der Vorbereich bezeichnet dabei alle Plätze A, von denen eine Kante zu einer Transition T führt:

$$\bullet T = \{A \in \mathcal{A} \mid (T, A) \in \mathcal{F}\}. \tag{A.13}$$

Analog bezeichnet der Nachbereich alle Plätze A, zu denen eine Kante von einer Transition T aus führt:

$$T\bullet = \{A \in \mathcal{A} \mid (A, T) \in \mathcal{F}\}. \tag{A.14}$$

Ein Schaltvorgang ist definiert durch:

$$M'(A) = \begin{cases} M(A) - W(A,T) & \text{für} \quad A \in \bullet T \backslash T\bullet \\ M(A) + W(A,T) & \text{für} \quad A \in T\bullet \setminus \bullet T \\ M(A) - W(A,T) + W(A,T) & \text{für} \quad A \in \bullet T \cap T\bullet \\ M(A) & \text{sonst} \end{cases} \tag{A.15}$$

Ein für diese Arbeit wichtiges Merkmal von Petri-Netzen ist die Möglichkeit zur Modularisierung. Dabei werden transitionsberandete Teilnetze durch Transitionen und platzberandete Teilnetze durch Plätze ersetzt (Vergröberung) bzw. Plätze werden durch platzberandete Teilnetze und Transitionen werden durch transitionsberandete Teilnetze ersetzt (Verfeinerung).

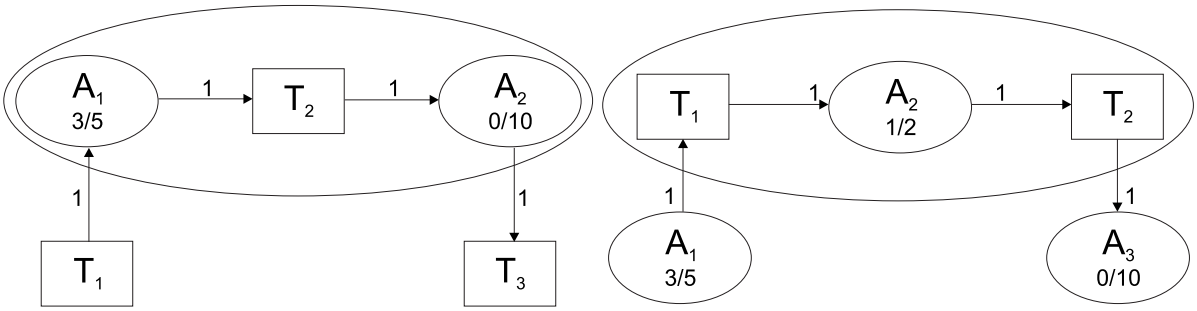


Bild A.2: Beispiele zur Modularisierung von Petri-Netzen. Platzberandetes Teilnetz (links) und transitionsberandetes Teilnetz (rechts)

Zur Abbildung von komplexen, parallel und/oder sequenziell zu verarbeitenden Prozessen verfügen Petri-Netze zudem über Mechanismen zur Verzweigung (eine Transition bedient mehrere Nachfolgeplätze), Zusammenführung (die Folgetransition kann erst schalten, wenn alle Zweige fertig sind) und Synchronisation (mehrere Zweige werden durch eine gemeinsame Transition synchronisiert, Bild A.3).

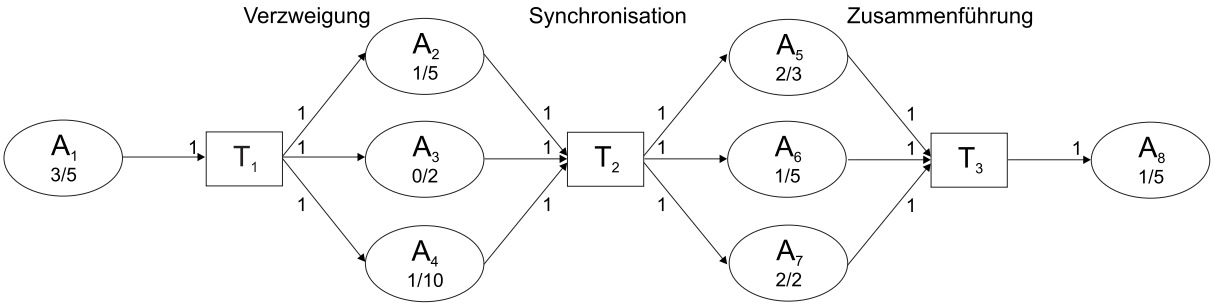


Bild A.3: Verzweigung, Synchronisation und Zusammenführung von Petri-Netzen

B Denavit-Hartenberg-Notation

Mit der Denavit-Hartenberg-Notation lassen sich offene kinematische Ketten in kartesischen Bezugskoordinaten beschreiben. Die kinematische Kette wird dabei als eine Menge benachbarter starrer Glieder angesehen, die durch Gelenke verbunden sind [254]. Ein Objekt im Raum mit körperfestem Koordinatensystem (Robotik: Gelenkkoordinatensystem) ist durch seine Position und Ausrichtung bezüglich eines raumfesten Koordinatensystems (Robotik: Weltkoordinatensystem) vollständig beschrieben [78](vgl. Bild B.1).

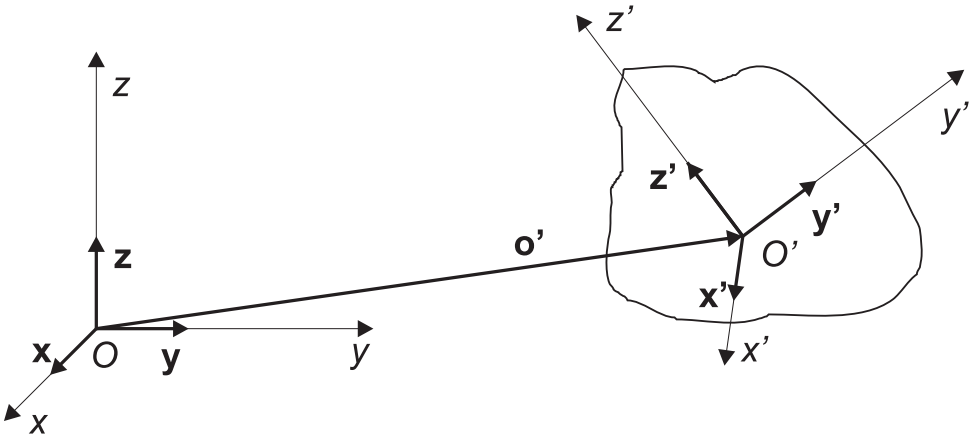


Bild B.1: Position und Ausrichtung eines Objektes im Raum

Die Position wird durch die Translation des körperfesten Koordinatensystems um den Ortsvektor **o'** in Bezug auf das raumfeste Koordinatensystem ausgedrückt.

$$\mathbf{o'} = \begin{pmatrix} o_x \\ o_y \\ o_z \end{pmatrix} \tag{B.1}$$

Die Ausrichtung ist durch die Winkelbeziehungen zwischen den Koordinatensystemen formuliert. Dargestellt wird dies in der Rotationsmatrix **R**. Für eine elementare Drehung des Objektes mit dem Winkel α um die x-Achse ergibt sich folgende Rotationsmatrix:

$$\mathbf{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{B.2}$$

Analoge Formeln gelten für die Drehungen um die y- und z-Achse:

$$\mathbf{R}_y(\nu) = \begin{pmatrix} \cos(\nu) & 0 & \sin(\nu) \\ 0 & 1 & 0 \\ -\sin(\nu) & 0 & \cos(\nu) \end{pmatrix} \tag{B.3}$$

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.4}$$

Verkettete Rotationen werden durch Multiplikation der elementaren Rotationen erhalten. Da die Matrixmultiplikation nicht kommutativ ist, gibt es zwei Interpretationen von verketteten Rotationen [254]. Dabei wird unterschieden, ob um Achsen des raumfesten Koordinatensystems (x, y, z) oder um die körperfesten Achsen (x', y', z') gedreht wird. Im ersten Fall werden die Rotationsmatrizen von links zur resultierenden Matrix multipliziert. Bei Drehungen um die eigenen Achsen werden die elementaren Rotationen nachmultipliziert, d.h. von rechts zur resultierenden Rotationsmatrix multipliziert. Beispielsweise wird bei nacheinander ausgeführten Drehungen des O'-KS um die x-, y- und z-Achse Gleichung (B.5) erhalten, aber bei aufeinander folgenden Drehungen um x'-, y'- und z'-Achse Gleichung (B.6).

$$\mathbf{R} = \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(\nu) \cdot \mathbf{R}_x(\alpha) \tag{B.5}$$

$$\mathbf{R} = \mathbf{R}_{x'}(\alpha) \cdot \mathbf{R}_{y'}(\nu) \cdot \mathbf{R}_{z'}(\theta) \tag{B.6}$$

Die geometrische Beziehung zwischen den Koordinatensystemen ist mathematisch formuliert eine zusammengesetzte Transformation, bestehend aus Translation und Rotation. Damit Translation und Rotation gemeinsam berechnet werden können, werden Translationsvektor **t** und Rotationsmatrix **R** durch die Einführung homogener Koordinaten zu einer homogenen (4x4)-Transformationsmatrix zusammengefasst, wobei die Skalierung üblicherweise Eins ist [78]:

$$\mathbf{X} = \begin{pmatrix} \text{Rotation (3x3)} & \text{Translation (3x1)} \\ \\ \text{[0 0 0] (1x3)} & \text{Skalierung (1x1)} \end{pmatrix}$$

Das Zusammenfassen von Rotation und Translation zu einer Transformationsmatrix wird als Frame bezeichnet.

Zur Beschreibung der Lage eines Objekts im raumfesten Koordinatensystem sind im Allgemeinen drei rotatorische und drei translatorische Parameter notwendig. Werden Transformationsmatrizen zur Darstellung kinematischer Ketten verwendet, bei denen jedem Gelenk ein Koordinatensystem zugeordnet ist, kann die Anzahl der Parameter durch geeignetes Platzieren der Gelenkkoordinatensysteme verringert werden. Diesem Zweck dient das Verfahren von Denavit-Hartenberg [78, 254] (vgl. Bild B.2).

Die Transformation vom Koordinatensystem des $(j-1)$-ten Gelenkes auf das Koordinatensystem des j-ten Gelenkes erfolgt nach folgenden Regeln (DH-Konvention):

- die Koordinatensysteme liegen in den Bewegungsachsen,
- die $\mathrm{z_{j-1}}$-Achse liegt entlang der Bewegungsachse des i-ten Gliedes,
- die $\mathrm{x_j}$-Achse steht senkrecht zur $\mathrm{z_{j-1}}$-Achse und zeigt von ihr weg,
- die $\mathrm{y_j}$-Achse ergänzt die $\mathrm{x_j}$- und $\mathrm{z_j}$-Achse zu einem rechts orientierten kartesischen Koordinatensystem.

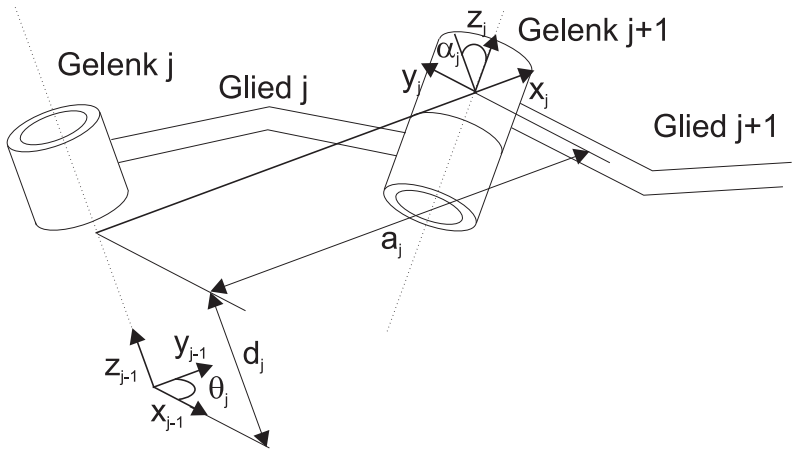


Bild B.2: DENAVIT-HARTENBERG-Parameter

Die Transformation vom Gelenkkoordinatensystem $(j-1)$ auf das Gelenkkoordinatensystem (j) erfolgt durch vier Elementarbewegungen:

1. Rotation θ_j um die $\mathrm{z_{j-1}}$-Achse, damit die $\mathrm{x_{j-1}}$-Achse parallel zu der $\mathrm{x_j}$-Achse liegt:

$$\mathbf{R}_{z_{j-1}}(\theta_j) = \begin{pmatrix} \cos(\theta_j) & -\sin(\theta_j) & 0 & 0 \\ \sin(\theta_j) & \cos(\theta_j) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.7}$$

2. Translation d_j entlang der $\mathrm{z_{j-1}}$-Achse zu dem Punkt, wo sich $\mathrm{z_{j-1}}$ und $\mathrm{x_j}$ schneiden:

$$\mathbf{T}_{z_{j-1}}(d_j) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.8}$$

3. Translation a_j entlang der $\mathrm{x_j}$-Achse um die Ursprünge in Deckung zu bringen:

$$\mathbf{T}_{x_j}(a_j) = \begin{pmatrix} 1 & 0 & 0 & a_j \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.9}$$

4. Rotation α_j um die $\mathrm{x_j}$-Achse um die $\mathrm{z_j}$- und $\mathrm{y_j}$-Achse in Übereinstimmung zu bringen:

$$\mathbf{R}_{x_j}(\alpha_j) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_j) & -\sin(\alpha_j) & 0 \\ 0 & \sin(\alpha_j) & \cos(\alpha_j) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.10}$$

In Matrizenschreibweise lautet die Denavit-Hartenberg-Transformation von Gelenk (j-1) auf (j):

$$\mathbf{X}_{j-1,j} = \mathbf{R}_{z_{j-1}}(\theta_j) \cdot \mathbf{T}_{z_{j-1}}(d_j) \cdot \mathbf{T}_{x_j}(a_j) \cdot \mathbf{R}_{x_j}(\alpha_j)$$

$$\mathbf{X}_{j-1,j} = \begin{pmatrix} \cos(\theta_j) & -\sin(\theta_j)\cdot\cos(\alpha_j) & \sin(\theta_j)\cdot\sin(\alpha_j) & a_j\cdot\cos(\theta_j) \\ \sin(\theta_j) & \cos(\theta_j)\cdot\cos(\alpha_j) & -\cos(\theta_j)\cdot\sin(\alpha_j) & a_j\cdot\sin(\theta_j) \\ 0 & \sin(\alpha_j) & \cos(\alpha_j) & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.11}$$

Durch Multiplikation aller DH-Matrizen wird wieder eine (4x4)-Matrix erhalten, die die Position und Orientierung des Endeffektors (Index E) in Weltkoordinaten (Index W) enthält.

$$\mathbf{X}_{W,E} = \mathbf{X}_{W,1}\cdot\mathbf{X}_{1,2}\cdot ... \cdot\mathbf{X}_{j-1,j}\cdot\mathbf{X}_{j,E}$$

Die so genannten DH-Parameter werden in einer Tabelle, welche üblicherweise als DH-Notation bezeichnet wird, festgehalten (vgl. Tabelle B.1):

Glied	θ_j	d_j	a_j	α_j
1	θ_1	d_1	a_1	α_1
2	θ_2	d_2	a_2	α_2
:	..	..	..	..

Tabelle B.1: DH-Notation

C Modellparameter

C.1 Flexibler Mehrfinger-Robotergreifer

Parameter [Einheit]	Bedeutung	Wert
$l_{(1,1A)}$ [m]	Länge Fingerglied 1A Kleiner Finger	0.024
$l_{(1,1B)}$ [m]	Länge Fingerglied 1B Kleiner Finger	0.010
$l_{(1,2)}$ [m]	Länge Fingerglied 2 Kleiner Finger	0.018
$l_{(1,Tip)}$ [m]	Länge Fingerglied 3 Kleiner Finger	0.017
$l_{(2,1A)}$ [m]	Länge Fingerglied 1A Ringfinger	0.034
$l_{(2,1B)}$ [m]	Länge Fingerglied 1B Ringfinger	0.010
$l_{(2,2)}$ [m]	Länge Fingerglied 2 Ringfinger	0.020
$l_{(2,Tip)}$ [m]	Länge Fingerglied 3 Ringfinger	0.020
$l_{(3,1)}$ [m]	Länge Fingerglied 1 Mittelfinger	0.049
$l_{(3,2)}$ [m]	Länge Fingerglied 2 Mittelfinger	0.022
$l_{(3,Tip)}$ [m]	Länge Fingerglied 3 Mittelfinger	0.020
$l_{(4,1A)}$ [m]	Länge Fingerglied 1A Zeigefinger	0.034
$l_{(4,1B)}$ [m]	Länge Fingerglied 1B Zeigefinger	0.010
$l_{(4,2)}$ [m]	Länge Fingerglied 2 Zeigefinger	0.020
$l_{(4,Tip)}$ [m]	Länge Fingerglied 3 Zeigefinger	0.020
$l_{(5,1A)}$ [m]	Länge Fingerglied 1A Daumen	0.008
$l_{(5,1B)}$ [m]	Länge Fingerglied 1B Daumen	0.020
$l_{(5,2)}$ [m]	Länge Fingerglied 2 Daumen	0.015
$l_{(5,Tip)}$ [m]	Länge Fingerglied 3 Daumen	0.015

Tabelle C.1: Kinematische Modellparameter des Greifers

Parameter [Einheit]	Bedeutung	Wert
$c\,[m^3/rad]$	Federkonstante des Aktors	$6.5 \cdot 10^{-7}$
d [m]	geometrische Konstante des Aktors	$12 \cdot 10^{-3}$
e [m]	geometrische Konstante des Aktors	$3 \cdot 10^{-3}$
K_V [m]	geometrische Ventilkonstante	$3.245 \cdot 10^{-3}$

Tabelle C.2: Modellparameter der Antriebskomponenten des Greifers

C.2 Anthropomorphe Halseinheit

Parameter [Einheit]	Bedeutung	Wert
a_3 [m]	Translationsweg entlang x_3-Achse der Halseinheit	0.12
a_4 [m]	Translationsweg entlang x_4-Achse der Halseinheit	0.09

Tabelle C.3: Kinematische Modellparameter der Halseinheit

Parameter [Einheit]	Bedeutung	Wert
J_{GE} [gcm^2]	Massenträgheitsmoment bezogen auf den Wave-Generator	$3.0 \cdot 10^{-7}$
J_M [gcm^2]	Rotorträgheitsmoment	$5.8 \cdot 10^{-7}$
k_M [mNm/A]	Drehmomentkonstante	26.1
L [μH]	Anschlussinduktivität	265
m_K [kg]	Kopfmasse	1.5
R [Ω]	Anschlusswiderstand	7.1

Tabelle C.4: Modellparameter der Antriebskomponenten der Halseinheit

D Programmierte Funktionen

D.1 Entwicklungsplattform - MATLAB

Dateiname	Funktion	Zeilen
BedienerInterface.m	Schnittstelle zur Robotersteuerung	484
Beenden_Callback.m	Callback zum Abbruch der Aufgabenausführung	48
dv_netz_laden.m	Laden eines ausgewählten Koordinierungsnetzes	145
dv_netz_stoppen.m	Unterbrechen des aktiven Koordinierungsnetzes	158
dv_netz_loeschen.m	Löschen des derzeit aktiven Koordinierungsnetzes	163
Enable_On.m	Callback zur Aktivierung eines Buttons	115
Fortsetzen_Callback.m	Callback zum Fortsetzen der Aufgabenausführung	121
HWD_Callback.m	Callback zur modellbasierten Hardwareüberwachung	7
Kontakt_Callback.m	Callback zur Simulation von Objektkontakt	7
PetriEXE.m	Ausführung von Petri-Netzen	217
PetriPARA.m	Einlesen von Petri-Netz-Parametern	75
PetriVISU.m	Animation der eingelesenen Petri-Netze	404
popup_menu_callback.m	Callback zur Aktivierung eines Popupmenus	363
PowerON_Callback.m	Callback zur Aktivierung des Robotersystems	7
Radiobutton_Callback.m	Callback zur Aktivierung eines Radiobuttons	5
RessourcenManager.m	Funktion zur Hardware-Ressourcenverwaltung	209
StandBy_Callback.m	Callback zur Aktivierung des StandBy-Zustands	7
start_steuerungsarchitektur.m	Startfile zur Initialisierung	370
success.m	Roboteraufgabe erfolgreich beendet	111
Trigger.m	Trigger-Funktion	116
Unterbrechen_Callback.m	Callback zur Unterbrechung der Aufgabenausführung	7

Tabelle D.1: MATLAB-m-Files zum Entwurf, zur Analyse und zur Simulation von Petri-Netzen

Dateiname	Funktion	Zeilen
cam.m	Lochkamera-Modell	141
dynamik_halseinheit.m	Dynamikmodell der Halseinheit	186
emotor.m	Modell eines E-Motors der Halseinheit	148
forkin.m	Vorwärtskinematik der Halseinheit	101
gravitationskompensation.m	Gravitationskompensation der Halseinheit	111
hydraulik.m	hydraulisches Modell des Greifers	276
invkin.m	vereinfachte inverse Kinematik der Halseinheit	102
kontaktkraft.m	Berechnung der Kontaktkraft bei Objektkontakt	111
mechanik.m	Dynamikmodell des Robotergreifers	186
objekt.m	Objektmodell des zu greifenden Objekts	135
pi_regler_regler.m	Regelungsalgorithmus des Greifers	348
regler_kondisk.m	Positions-Regelungsalgorithmus der Halseinheit	202
robotergreifer_kinematik.m	Vorwärtskinematik des Greifers	92
stoermoment.m	Aus der Kontaktkraft resultierendes Störmoment	44
TrajektoriengenerierungBP.m	Trajektorie Bewegungsprogramm Halseinheit	169
TrajektoriengenerierungBP_Gr.m	Trajektorie Bewegungsprogramm Greifer	168
TrajektoriengenerierungGreifen.m	Trajektorie Greifbewegung Greifer	513
TrajektoriengenerierungOV.m	Trajektorie Aufmerksamkeitssteuerung Halseinheit	299
TrajektoriengenerierungPR.m	Trajektorie Positionsregelung Halseinheit	221
TrajektoriengenerierungPR_Gr.m	Trajektorie Positionsregelung Greifer	177
visu_greifer.m	Visualisierung der Simulationsergebnisse Greifer	206
visu_greifer_2d.m	2D-Animation der Simulationsergebnisse Greifer	206
visu_hals.m	Visualisierung der Simulationsergebnisse Halseinheit	299
visu_hals_3d.m	3D-Animation der Simulationsergebnisse Halseinheit	164

Tabelle D.2: MATLAB-m-Files zur Trajektoriengenerierung, Regelung und Modellbildung

Dateiname	Funktion	Zeilen
bewertung.m	Berechnung der Bewertungsfunktion	16
FGI_Berechnung.m	Berechnung der Führungsgrößenintervalle	210
huebwa_hals.m	Hardwareüberwachung Halseinheit	126
huebwa_greifer.m	Hardwareüberwachung Greifer	141
ueberwachung.m	Berechnung Interna Stabilitätsüberwachung	181
VerwaltungsmodulGreifer.m	Aufgabenverwaltung Greifer	282
VerwaltungsmodulHals.m	Aufgabenverwaltung Halseinheit	282
visu_problembew.m	Visualisierung von Fehlerdetektion und Problemlösung für Halseinheit und Greifer	299

Tabelle D.3: MATLAB-m-Files zur Fehlerdetektion und Problemlösung

D.2 Zielplattform - MCA

Dateiname	Funktion	Zeilen
ControllerModule.C	MCA-Module pi_Regler.C	277
ControllerModule.h	Def. Eingangs- und Ausgangsgrößen pi_Regler.C	198
fuzzstab.C	Berechnung des Stabilitätsgrads	104
fuzzstab.h	Parameterdeklaration für fuzzstab.C	21
hydraulik.C	hydraulisches Modell des Greifers	224
hydraulik.h	Parameterdeklaration für hydraulik.C	26
HydrlPlantModule.C	MCA-Module hydraulik.C	204
HydrlPlantModule.h	Def. Eingangs- und Ausgangsgrößen hydraulik.C	160
intervall.C	Berechnung der Führungsgrößenintervalle	41
intervall.h	Parameterdeklaration für intervall.C	25
mechanik.C	Dynamikmodell des Robotergreifers	124
mechanik.h	Parameterdeklaration für mechanik.C	26
MechPlantModule.C	MCA-Module mechanik.C	156
MechPlantModule.h	Def. Eingangs- und Ausgangsgrößen mechanik.C	136
pi_Regler.C	Regelungsalgorithmus des Greifers	255
pi_Regler.h	Parameterdeklaration für pi_Regler.C	35
ReglerGroup.C	MCA Gruppe Regler	111
ReglerPart.C	MCA Part Regler	62
sfun_fuzzstab.C	Simulink Wrapper S-Funktion fuzzstab.C	119
sfun_hydraulik.C	Simulink Wrapper S-Funktion hydraulik.C	135
sfun_intervall.C	Simulink Wrapper S-Funktion intervall.C	137
sfun_mechanik.C	Simulink Wrapper S-Funktion mechanik.C	136
sfun_pi_Regler.C	Simulink Wrapper S-Funktion pi_Regler.C	151
sfun_ueberwachung.C	Simulink Wrapper S-Funktion ueberwachung.C	133
SupervisionModule.C	MCA-Module ueberwachung.C	175
SupervisionModule.h	Definition der Eingangs- und Ausgangsgrößen der Funktion ueberwachung.C	136
ueberwachung.C	Berechnung der Bewertungsfunktion	163
ueberwachung.h	Parameterdeklaration für ueberwachung.C	30
VHPGroup.C	MCA Gruppe Greifer	132
VHPGroup.h	Definition der Eingangs- und Ausgangsgrößen der Funktion VHPGroup.C	138
VHPPart.C	MCA Part Greifer	51

Tabelle D.4: C-Files zur Ausführung auf der Entwicklungs- und Zielplattform

Literaturverzeichnis

[1] VDI-Richtlinie 3542: Sicherheitstechnische Begriffe für Automatisierungssysteme.

[2] DIN 25448: Ausfalleffektanalyse. 1980.

[3] DIN 25424: Fehlerbaumanalyse, Methode und Bildzeichen. 1981.

[4] DIN 40041: Zuverlässigkeit - Begriffe. 1990.

[5] ISO 10218: Industrieroboter - Sicherheit. 1993.

[6] Robotersteuerung KRC1 und Roboter KR6/1. Techn. Ber., KUKA Roboter GmbH, Augsburg, 1996.

[7] Sirotec RCM2 und RCM3 Handbücher. Techn. Ber., Siemens AG, 1998.

[8] DIN EN 292: Safety of machinery - Basic concepts, general principles for design: Basic terminology, methodology, technical principles. 2000.

[9] DIN EN 61508: Funktionale Sicherheit elektrischer/elektronischer/programmierbar elektronischer sicherheitsbezogener Systeme: Begriffe und Abkürzungen. 2002.

[10] ABEL, D.: *Petri-Netze für Ingenieure - Modellbildung und Analyse diskret gesteuerter Systeme*. Berlin: Springer Verlag, 1990.

[11] ADAMS, B.; BREAZEAL, C.; BROOKS, R.; SCASSELLATI, B.: Humanoid Robots: A New Kind of Tool. *IEEE Journal on Intelligent Systems* 15(4) (2000), S. 25–31.

[12] ADAMY, J.; BECHTEL, P.: Sicherheit mobiler Roboter. *at - Automatisierungstechnik* 51 (2003), S. 435–444.

[13] ALBERS, A.; BRUDNIOK, S.: Systematische Entwicklung des mechatronischen Gesamtsystems der Arme eines Humanoiden Roboters. In: *Proc., 3. Paderborner Workshop Intelligente Mechatronische Systeme*, 2005.

[14] ALBERS, A.; BRUDNIOK, S.; BURGER, W.: Design and Development Process of a Humanoid Robot upper Body through Experimentation. In: *Proc., IEEE-RAS/RSJ International Conference on Humanoid Robots*, 2004.

[15] ALBERS, A.; BRUDNIOK, S.; OTTNAD, J.; SAUTER, C.; SEDCHAICHARN, K.: ARMAR-III: Design of the Upper Body. In: *Proc., International Workshop on Human-Centered Robotic Systems (HCRS), Munich*, S. 63–68, 2006.

[16] ALBU-SCHÄFFER, A.: *Regelung von Robotern mit elastischen Gelenken am Beispiel der DLR-Leichtbauarme*. Dissertation, TU München, 2002.

[17] ALBU-SCHÄFFER, A.; HIRZINGER, G.: Cartesian Impedance Control Techniques for Torque Controlled Light-Weight Robots. *IEEE Transactions on Robotics and Automation* (2002), S. 657–663.

[18] AMIR, E.; MAYNARD-REID II, P.: LISA: A Robot Driven by Logical Subsumption. In: *Proc., Symposium on Logical Formalizations of Commonsense Reasoning (CommonSense)*, 2001.

[19] ANGELBAUER, K.: *Modellbildung und Reglerentwurf für die Halseinheit eines humanoiden Roboters*. Diplomarbeit, Universität Karlsruhe (TH), 2005.

[20] ARACIL, J.; CARCIA-CEZERO, A.; OLLERO, A.: Stability Indices for the Global Analysis of Expert Control Systems. *IEEE Transactions on Systems, Man, and Cybernetics* 19 (1989), S. 998–1007.

[21] ARKIN, R. C.: Path Planning for a Vision-based Autonomous Robot. In: *Proc., SPIE Conference on Mobile Robots, Cambridge*, S. 240–249, 1986.

[22] ASFOUR, T.: *Sensomotorische Bewegungskoordination zur Handlungsausführung eines humanoiden Roboters*. Dissertation, Universität Karlsruhe (TH), 2003.

[23] ASFOUR, T.; BERNS, K.; DILLMANN, R.: The Humanoid Robot ARMAR: Design and Control. In: *Proc., IEEE-RAS International Conference on Humanoid Robots*, 2000.

[24] ASFOUR, T.; LY, D.; REGENSTEIN, K.; DILLMANN, R.: Coordinated Task Execution for Humanoid Robots. In: *Proc., 9th International Symposium on Experimental Robotics 2004 (ISER 04)*, 2004.

[25] ASFOUR, T.; REGENSTEIN, K.; AZAD, P.; SCHRÖDER, J.; DILLMANN, R.: ARMAR-III: A Humanoid Platform for Perception-Action Integration. In: *Proc., International Workshop on Human-Centered Robotic Systems (HCRS), Munich*, S. 51–56, 2006.

[26] ATKESON, C. G.; HALLE, J.; POLLICK, F.; RILEY, M.; KOTOSAKA, S.; SCHAAL, S.; SHIBATA, T.AND TEVATIA, G.; UDE, A.; VIJAYAKAUMAR, S.; KAWATO, M.: Using Humanoid Robots to Study Human Behaviour. *IEEE International Journal on Intelligent Systems* 15(4) (2000), S. 46–55.

[27] BAAS, M.; FINKENZELLER, D.; PREUSS, S.; THÜRING, S.; SCHMITT, A.: VISUM: Ein Virtual-Reality-System für das Testen und Trainieren von humanoiden Robotern. In: *Proc., Human Centered Robotic Systems, Karlsruhe*, S. 35–42, 2002.

[28] BANDEMER, H.; HARTMANN, S.: A Fuzzy Approach to Stability of Fuzzy Controllers. *Fuzzy Sets and Systems* 96 (1998), S. 161–172.

[29] BARSCHDORFF, D.; BECKER, D.: Neuronale Netze als Signals- und Musterklassifikatoren. *Technisches Messen* 57 (1990), S. 437–444.

[30] BECHER, R.; STEINHAUS, P.; DILLMANN, R.: Interactive Object Modelling for a Humanoid Service Robot. In: *Proc., Humanoids*, 2003.

[31] BECHER, R.; STEINHAUS, P.; ZÖLLNER, R.; DILLMANN, R.: Design and Implementation of an Interactive Object Modelling System. In: *Proc., Conference Robotik/ISR*, 2006.

[32] BECHLER, D.; KROSCHEL, K.: Confidence Scoring of Time Difference of Arrival Estimation for Speaker Localization with Microphone Arrays. In: *Proc., 13. Konferenz Elektronische Sprachsignalverarbeitung (ESSV)*, 2002.

[33] BECHLER, D.; SCHLOSSER, M. S.; KROSCHEL, K.: System for Robust 3D Speaker Tracking using Microphone Array Measurements. In: *Proc., International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[34] BECK, S.: *Entwicklung, Aufbau und Erprobung einer Testumgebung für die Evaluierung flexibler Fluidaktoren*. Diplomarbeit, Universität Karlsruhe, 2001.

[35] BECK, S.: *Ein Beitrag zum automatischen Entwurf von Fuzzy-Entscheidungssystemen bei unvollständiger Information*. Dissertation, Universität Karlsruhe, Universitätsverlag Karlsruhe, 2005.

[36] BECK, S.; LEHMANN, A.; LOTZ, T.; MARTIN, J.; KEPPLER, R.; MIKUT, R.: Modellgestützte adaptive Regelungskonzepte für eine fluidisch betriebene Roboterhand. In: *Proc., GMA-Kongress*, S. 65–72, VDI-Verlag, 2003.

[37] BECK, S.; MIKUT, R.; JÄKEL, J.: A Cost-Sensitive Learning Algorithm for Fuzzy Rule-Based Classifiers. *Mathware and Soft Computing* 11(2-3) (2004), S. 175–195.

[38] BECK, S.; MIKUT, R.; LEHMANN, A.; BRETTHAUER, G.: Model-Based Control and Object Contact Detection for a Fluidic Actuated Robotic Hand. In: *Proc., 42nd IEEE Conference on Decision and Control*, S. 6369–6374, 2003.

[39] BECK, S.; REISCHL, R.; BRETTHAUER, G.: Steuerungs- und Regelungskonzepte für eine myoelektrische Handprothese. In: *Proc., 37. Regelungstechnisches Kolloquium in Boppard*, S. 43, 2003.

[40] BEETZ, M.; MCDERMOTT, D.: Improving Robot Plans during their Execution. In: *Proc., AI Planning Systems Conference*, S. 3–12, 1994.

[41] BEICHELT, F.: *Zuverlässigkeit strukturierter Systeme*. VEB Verlag Technik, Berlin, 1988.

[42] BEICHELT, F.: *Zuverlässigkeits- und Instandhaltungstheorie*. B.G. Teubner Verlag, Stuttgart, 1993.

[43] BENDER, J.; BAAS, M.; SCHMITT, A.: Ein neues Verfahren für die mechanische Simulation in VR-Systemen und in der Robotik. *Proc., 17. Symposium Simulationstechnik (ASIM)* (2003).

[44] BERNARDIN, K.; GEHRIG, T.; STIEFELHAGEN, R.: Multi- and Single View Multiperson Tracking for Smart Room Environments. In: *Proc., CLEAR Evaluation Workshop*, 2006.

[45] BETH, T.; BOESNACH, I.; HAIMERL, M.; MOLDENHAUER, J.; BÖS, K.; WANK, V.: Analyse, Modellierung und Erkennung menschlicher Bewegungen. In: *Proc., Human Centered Robotic Systems, Karlsruhe*, S. 1–8, 2002.

[46] BETH, T.; BOESNACH, I.; HAIMERL, M.; MOLDENHAUER, J.; BÖS, K.; WANK, V.: Characteristics in Human Motion - From Acquisition to Analysis. In: *Proc., Humanoids 2003*, 2003.

[47] BILLARD, A.: Learning Motor Skills by Imitation: A Biologically Inspired Robotic Model. *Cybernetics and Systems* (2000), S. 155–193.

[48] BILLARD, A.; MATARIC, M.: Learning Human Arm Movements by Imitation: Evaluation of a Biologically-Inspired Connectionist Architecture. *Robotics and Autonomous Systems 941* (2001), S. 1–16.

[49] BISCHOFF, R.: HERMES - A Humanoid Mobile Manipulator for Service Tasks. In: *Proc., International Conference on Field and Service Robotics*, S. 508–515, 1997.

[50] BISCHOFF, R.: Design Concept and Realization of the Humanoid Service Robot HERMES. *Field and Service Robotics* (1998), S. 485–492.

[51] BISCHOFF, R.; SOMRAK, P.; GRAEFE, V.: Improving Dependability of Humanoids. In: *Proc., IEEE-RAS International Conference on Humanoid Robots*, 2001.

[52] BÖHM, R.; KREBS, V.: Ein Ansatz zur Stabilitätsanalyse und Synthese von Fuzzy-Regelungen. *at - Automatisierungstechnik* 41 (1993), S. 288–293.

[53] BOLL, M.: *Einsatz von Fuzzy-Control zur Regelung verfahrenstechnischer Prozesse*. Dissertation, Universität Paderborn, FIT-Verlag, 1997.

[54] BOLL, M.; HÖTTECKE, M.; DÖRRSCHEIDT, F.: Analyse von Fuzzy-Reglern in der Zustandsebene. *at - Automatisierungstechnik* 41 (1993), S. 145–151.

[55] BONASSO, R.; FIRBY, R.; GAT, E.; KORTENKAMP, D.; MILLER, D. P.; SLACK, M. G.: Experiences with an Architecture for Intelligent, Reactive Agents. *Journal of Experimental & Theoretical Artificial Intelligence* 9(2) (1997), S. 237–256.

[56] BRANICKY, M. S.: *Studies in Hybrid Systems: Modeling, Analysis, and Control*. Dissertation, Laboratory for Information and Decision Systems, MIT, 1995.

[57] BRETTHAUER, G.; OPITZ, H.-P.: Stability of Fuzzy Systems - A Survey. In: *Proc., 2nd European Congress on Intelligent Techniques and Soft Computing EUFIT'94*, S. 283–290, Aachen, 1994.

[58] BROOKS, R.: A Robust Layered Control System for a Mobile Robot. *MIT AI Lab Memo 864* (1985).

[59] BROOKS, R.: Achieving Artificial Intelligence through Building Robots. Techn. Ber., Massachussetts Institute of Technology, Artificial Intelligence Laboratory, 1986.

[60] BROOKS, R.; BREAZEAL, C.; MARJANOVIC, M.; SCASSELLATI, B.; WILLIAMSON, M.: The Cog Project: Building a Humanoid Robot. In: *Proc., 1st International Conference on Humanoid Robots and Human friendly Robotics*, 1998.

[61] BÜHLER, H.: Stabilitätsuntersuchung von Fuzzy-Regelungen. In: *Proc., GMA Aussprachetag Fuzzy Control (VDI-Bericht 1113)*, S. 309–318, 1994.

[62] BURGHART, C.; MIKUT, R.; STIEFELHAGEN, R.; ASFOUR, T.; HOLZAPFEL, H.; STEINHAUS, P.; DILLMANN, R.: A Cognitive Architecture for a Humanoid Robot: A First Approach. In: *Proc., IEEE-RAS Conference on Humanoid Robots, Tsukuba, Japan*, S. 357–362, 2005.

[63] BUTTERFASS, J.; GREBENSTEIN, M.; LIU, H.; HIRZINGER, G.: DLR Hand II: Next Generation of a Dextrous Robot Hand. In: *Proc., 19th IEEE International Conference on Robotics and Automation*, 2001.

[64] CANIOU, D.: *Entwicklung und Erprobung eines hybriden Regelungs- und Steuerungskonzeptes zum sicheren Greifen mit einer flexiblen Roboterhand*. Diplomarbeit, Universität Karlsruhe (TH), 2004.

[65] CAO, S. G.; REES, N. W.; FENG, G.: Lyapunov-Like Stability Theorems for Continuous-Time Fuzzy Control Systems. *International Journal of Control* 69 (1998), S. 49–64.

[66] CHEN, J.; CHEN, L.: Study on Stability of Fuzzy Closed-Loop Control Systems. *Fuzzy Sets and Systems* 57 (1993), S. 159–168.

[67] CHEN, Y. Y.: Stability Analysis of Fuzzy Control - a Lyapunov Approach. In: *Proc., IEEE Annual Conference Systems, Man, and Cybernetics*, S. 1027–1031, 1987.

[68] CHENG, G.; Y., K.: Complex Continous Meaningful Humanoid Interaction: A Multi Sensory-Cue Based Approach. In: *Proc., 2000 IEEE International Conference on Robotics & Automation*, S. 2235–2242, 2000.

[69] CONNELL, J.: SSS: A Hybrid Architecture Applied to Robot Navigation. In: *Proc., IEEE Conference on Robotics and Automation (ICRA)*, S. 2719–2724, 1992.

[70] CRAIG, J.: *Introduction to Robotics*. Addison-Wesley, 1986.

[71] DAL CIN, M.: *Fehlertolerante Systeme: Modelle der Zuverlässigkeit*. B.G. Teubner Verlag, Stuttgart, 1979.

[72] D'ANGELO, L.: *Ereignis-diskrete Regelung für humanoide Roboter basierend auf dem Aktionsprimitivenkonzept*. Diplomarbeit, Universität Karlsruhe, 2006.

[73] DHILLON, B.: *Robot Reliability and Safety*. Springer, 1991.

[74] DHILLON, B.; FASHADI, A.: Safety and Reliability Assessment Techniques in Robotics. *Robotica* 15 (1997), S. 701–708.

[75] DILLMANN, R.: *Lernende Roboter - Aspekte maschinellen Lernens*. Springer, 1988.

[76] DILLMANN, R.: Towards Humanoid Robots in Human-Centered Environments. In: *Proc., IEEE-RAS International Conference on Humanoid Robots*, 2003.

[77] DILLMANN, R.: Vorlesungsskript: Verhaltensbasierte Steuerungsarchitekturen. Universität Karlsruhe, Sommersemester 2005.

[78] DILLMANN, R.; HUCK, M.: *Informationsverarbeitung in der Robotik*. Springer, 1991.

[79] DILLMANN, R.; ROGALLA, O.; EHRENMANN, M.; ZÖLLNER, R.; BORDEGONI, M.: Learning Robot Behaviour and Skills Based on Human Demonstration and Advice: The Machine Learning Paradigm. In: *Proc., 9th International Symposium of Robotics Research*, S. 229–238, 1999.

[80] DING, X.: *Frequenzbereichsverfahren zur beobachtergestützten Fehlerentdeckung*. Dissertation, Universität Duisburg, 1992.

[81] DING, X.; FRANK, P. M.: Frequency Domain Approach and Threshold Selector for Robust Model-Based Fault Detection and Isolation. In: *Proc., IFAC/IMACS Symposium SAFERPROCESS Baden-Baden*, S. 307–312, 1991.

[82] EHRENMANN, M.; ROGALLA, O.; ZÖLLNER, R.; DILLMANN, R.: Analyse der Instrumentarien zur Belehrung und Kommandierung von Robotern. In: *Proc., Human Centered Robotic Systems, Karlsruhe*, S. 25–35, 2002.

[83] EHRENMANN, M.; ZÖLLNER, R.; ROGALLA, O.; VACEK, S.; DILLMANN, R.: Observation in Programming by Demonstration: Training and Execution Environment. In: *Humanoids 2003*, 2003.

[84] FESTO AG & CO.: Fluidic Muscle MAS Produktdatenblatt. Datenblatt, `www.festo.com`, 2003.

[85] FIKES, R. E.; NILSSON, N. J.: STRIPS: A New Approach to Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2 (1971), S. 189–208.

[86] FINKEMEYER, B.: *Robotersteuerungsarchitektur auf der Basis von Aktionsprimitiven*. Dissertation, TU Braunschweig, 2004.

[87] FINKEMEYER, B.; KRÖGER, T.; WAHL, F.: Aktionsprimitive: Ein universelles Roboter-Programmierparadigma. *at - Automatisierungstechnik* 53(4-5) (2005), S. 189–196.

[88] FIRBY, R. J.: Building Symbolic Primitives with Continuous Control Routines. In: *Proc., 1st International Conference on Artificial Intelligence Planning Systems*, S. 62–69, 1992.

[89] FLANDIN, G.: XMLTree: An XML Toolbox for Matlab. 2002.

[90] FÖLLINGER, O.: *Regelungstechnik*. Heidelberg: Hüthing, 1992.

[91] FÖLLINGER, O.: *Nichtlineare Regelungen*. München: Oldenbourg, 1993.

[92] FRANK, D.; REICHARD, G.: *Grenzen der Verkehrssicherheit und gesellschaftliche Akzeptanz*. Campus Verlag, Frankfurt/Main, 1997.

[93] FRANK, M.: Diagnoseverfahren in der Automatisierungstechnik. *at - Automatisierungstechnik* 42(2) (1994), S. 47–64.

[94] FRANK, P. M.: Advances in Observer-Based Fault Diagnosis. In: *Proc., Conference TOOLDIAG'93*, 1993.

[95] FRANK, P. M.; KIUPEL, N.: Fuzzy Supervision. *at - Automatisierungstechnik* 41(8) (1993), S. 293–299.

[96] FREYERMUTH, B.: *Wissensbasierte Fehlerdiagnose am Beispiel eines Industrieroboters*. Dissertation, TH Darmstadt, VDI-Verlag, Düsseldorf, 1993.

[97] FÜGEN, C.; HOLZAPFEL, H.: Tight Coupling of Speech Recognition and Dialog Management – Dialog-Context Dependent Grammar Weighting for Speech Recognition. In: *Proc., 1st International Conference on Logistics Strategy for Ports*, 2004.

[98] FUJITA, M.; KUROKI, Y.; ISHIDA, T.; DOI, T.: A Small Humanoid Robot SDR-4X for Entertainment Applications. *Advanced Intelligent Mechatronics* 2 (2003), S. 938–943.

[99] FUKAYA, N.; TOYAMA, N.; ASFOUR, T.; DILLMANN, R.: Design of the TUAT / Karlsruhe Humanoid Hand. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[100] GAT, E.: Integrating Planning and Reacting in a Heterogenous Asynchronous Architecture for Controlling Real-World Mobile Robots. In: *Proc., National Conference on Artificial Intelligence (AAAI)*, S. 809–815, 1992.

[101] GAT, E.: Three-Layer Architectures. In: *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems* (KORTENKAMP, D., Hg.), S. 195–210, AAAI Press, 1998.

[102] GERTLER, J.; CHANG, H.: An Instability Indicator for Expert Control. *IEEE Transactions on Control Systems* 8 (1986), S. 14–17.

[103] GIENGER, M.; LÖFFLER, K.; PFEIFFER, F.: Towards the Design of Biped Jogging Robot. In: *Proc., IEEE International Conference on Robotics and Automation*, S. 4140–4145, 2001.

[104] GIESELMANN, P.; DENECKE, M.: Towards Multimodal Interaction with an Intelligent Room. In: *Proc., Eurospeech 2003*, 2003.

[105] GIESEN, K.; MILIGHETTI, G.; FREY, C. W.; KUNTZE, H. B.: Strukturvariable multisensorielle Überwachung und Regelung von Mensch-Roboter-Systemen. In: *Proc., 38. Regelungstechnisches Kolloquium Boppard*, 2004.

[106] GOLUBOVIC, D.: *Modellbildung und Entwurf einer Druckregelung für pneumatisch betriebene flexible Fluidaktoren*. Diplomarbeit, Universität Karlsruhe (TH), 2003.

[107] GORGES, N.; BIERBAUM, A.; WÖRN, H.; DILLMANN, R.: Towards a Comprehensive Grasping System for ARMAR-III. In: *Proc., International Workshop on Human-Centered Robotic Systems (HCRS), Munich*, S. 57–62, 2006.

[108] GROSS, D.; HAUGER, W.; SCHNELL, W.: *Technische Mechanik*. Springer Lehrbuch, 1992.

[109] GROTE, K.-H.; FELDHUSEN, J.: *Dubbel Taschenbuch für den Maschinenbau*. Springer, 2001.

[110] GRUSCHINSKI, H.: *Simulation mechatronischer Systeme unter Nutzung der Systemtheorie und Matlab*. Dissertation, Otto-von-Guericke Universität Magdeburg, 2004.

[111] HÄGELE, M.; SCHAAF, W.; HELMS, E.: Robot Assistants at Manual Workplaces: Effective Cooperation and Safety Aspects. In: *Proc., 33rd International Symposium on Robotics*, 2002.

[112] HANEBECK, U. D.: Progressive Bayesian Estimation for Non-linear Discrete-Time Systems: The Measurement Step. In: *Proc., 2003 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2003.

[113] HARA, F.; KOBAYASHI, H.: A Face Robot Able to Recognize and Produce Facial Expression. In: *Proc., IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*, S. 1600–1607, 1996.

[114] HASHIMOTO, S.: Humanoid Robots in Waseda University - Hadaly-2 and WABIAN. In: *Proc., 1st International Workshop on Humanoid Robots and Human friendly Robotics*, S. 1–2, 1998.

[115] HIRAI, K.: Current and Future Perspective of Honda Humanoid Robot. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems*, S. 500–508, 1997.

[116] HIRAI, K.; HIROSE, M.; HAIKAWA, Y.; TAKENAKA, T.: The Development of Honda Humanoid Robot. In: *Proc., IEEE International Conference on Robotics and Automation*, S. 1321–1326, 1998.

[117] HIROSE, M.; HAIKAWA, Y.; TAKENAKA, T.; HIRAI, K.: Development of Humanoid Robot ASIMO. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[118] HODRUS, T.; MÜNZ, E.: Hybride Phänomene in zeitdiskreter Darstellung. *at - Automatisierungstechnik* 12 (2003), S. 574–582.

[119] HOLZAPFEL, H.; FUEGEN, C.; DENECKE, M.; WAIBEL, A.: Integrating Emotional Cues into a Framework for Dialog Management. In: *Proc., IEEE International Conference on Multimodal Interfaces*, 2002.

[120] HOLZAPFEL, H.; SCHAAF, T.; EKENEL, H. K.; SCHAA, C.; WAIBEL, A.: A Robot Learns to Know People - First Contacts of Lecture Notes in Artificial Intelligence. In: *Proc., KI*, 2006.

[121] HONDA: Informationsblatt - ASIMO. *Quelle: http://www.honda-robots.com* (2005), S. 1–3.

[122] HORSWILL, I.: Polly: A Vision-Based Artificial Agent. In: *Proc., National Conference on Artificial Intelligence (AAAI), AAAI/MIT Press*, S. 824–829, 1993.

[123] HWANG, G.-C.; LIU, S. C.: A Stability Approach to Fuzzy Control Design for Non-Linear Systems. *Fuzzy Sets and Systems* 48 (1992), S. 179–187.

[124] HWANG, Y.: Cross Motion Planning - A Survey. *ACM Computing Surveys 24* 3 (1992).

[125] IJSPEERT, A.: Learning Rhythmic Movements by Demonstration using Non-linear Oscillators. In: *Proc., 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL Lausanne, Switzerland*, S. 958–963, 2002.

[126] IJSPEERT, A.; NAKANISHI, J.; SCHAAL, S.: Learning Rhythmic Movements by Demonstration using Nonlinear Oscillators. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2002)*, S. 958–963, 2002.

[127] INOUE, H.; TACHI, S.; NAKAMURA, Y.; HIRAI, K.; OHYU, N.; HIRAI, S.; TANIE, K.; YOKOI, K.; HIRUKAWA, H.: Overview of Humanoid Robotics Project of METI. In: *Proc., 32nd International Symposium on Robotics*, S. 1478–1482, 2001.

[128] ISERMANN, R.: Fehlerdiagnose mit Prozessmodellen. *Technisches Messen* 51, 10 (1984), S. 345–355.

[129] ISERMANN, R.: *Digitale Regelsysteme*. Berlin: Springer, 1987.

[130] ISERMANN, R.: *Identifikation Dynamischer Systeme*. Berlin: Springer, 1992.

[131] ISERMANN, R.: *Überwachung und Fehlerdiagnose*. VDI-Verlag, 1994.

[132] ISHIGURO, H.; KANDA, T.; KIMOTO, K.; ISHIDA, T.: A Robot Architecture Based on Situated Modules. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems*, S. 1617–1623, 1999.

[133] JENNINGS, N. R.: Specification and Implementation of a Belief Desire Jointintention Architecture for Collaborative Agents. *Journal of Intelligent and Cooperative Information Systems* (1993), S. 289–318.

[134] JOHANSEN, T. A.: Fuzzy Model Based Control: Stability, Robustness, and Performance Indices. *IEEE Transactions on Fuzzy Systems* 2 (1993), S. 221–234.

[135] JØRGENSEN, R. B.: *Development and Test of Methods for Fault Detection and Isolation*. Dissertation, Aalborg University, 1995.

[136] JUNG-HOON, K.; JUN-HO, O.: Walking Control of the Humanoid Platform KHR-1 Based on Torque Feedback Control. In: *Proc., IEEE International Conference on Robotics and Automations*, 2004.

[137] KAGAMI, S.; NISHIWAKA, K.; KUFFNER, J.; SUGIHARA, T.; INABA, M.; INOUE, H.: Design and Implementation of Humanoid H6 and its Application to Remote Operation. In: *Proc., 7th International Symposium on Experimental Robotics*, 2000.

[138] KANEHIRO, F.; KANEKO, K.; FUJIWARA, K.; HARADA, K.; KAJITA, S.; YOKOI, K.; HIRUKAWA, H.; AKACHI, K.; ISOZUMI, T.: The First Humanoid Robot That Has the Same Size as a Human and That Can Lie Down and Get Up. In: *Proc., IEEE International Conference on Robotics and Automation*, S. 1633–1639, 2003.

[139] KANEKO, K.; KANEHIRO, F.; KAJITA, S.; HIRUKAWA, H.; KAWASAKI, T.; HIRATA, M.; AKACHI, K.; ISOZUMI, T.: Humanoid Robot HRP-2. In: *Proc., IEEE International Conference on Robotics and Automation*, S. 1083–1090, 2004.

[140] KATO, I.: Development of WABOT-1. *Biomechanism 2* (1973), S. 173–214.

[141] KATO, I.; MORI, Y.; MASUDA, T.: Pneumatically Powered Artificial Legs Walking Automatically Under Various Circumstances. In: *Proc., 4th. International Symposium in External Control of Human Extremities*, S. 458–470, 1972.

[142] KAWAMURA, K.; PETERS, R. A.; WILKES, D. M.; A., A. W.; ROGERS, T. E.: ISAC: Foundations in Human-Humanoid Interaction. *IEEE Intelligent Systems* (2000).

[143] KAWAMURA, K.; ROGERS, T.; AO, X.: Development of a Cognitive Model of Humans in a Multi-Agent Framework for Human-Robot Interaction. In: *Proc., 1st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, S. 1379–1386, 2002.

[144] KELLY, R.; SANTIBANEZ, V.: Global Regulation of Elastic Joint Robots Based on Energy Shaping. *IEEE Transactions on Automatic Control* 43(10) (1998), S. 1451–1456.

[145] KENDON, A.: *Gesture: Visible Action as Utterance*. Cambridge Univ Pr, West Nyack, New York, U.S.A. , 2004.

[146] KEPPLER, R.: *Zur Modellierung und Simulation von Mehrkörpersystemen unter Berücksichtigung von Greifkontakt bei Robotern*. Dissertation, Universität Karlsruhe, in Druck, 2006.

[147] KERN, W.; RAMMIG, F. J.: C-LAB Jahresbericht 2001 (2001).

[148] KIENDL, H.: Harmonic Balance for Fuzzy Control Systems. In: *Proc., EUFIT'93*, S. 137–141, Aachen, 1993.

[149] KLEINHAGENBROCK, M.: *Interaktive Verhaltenssteuerung für Robot Companions*. Dissertation, Universität Bielefeld, 2004.

[150] KLUTE, G. K.; CZERNIECKI, J. M.; HANNAFORD, B.: McKibben Artificial Muscles: Pneumatic Actuators with Biomechanical Intelligence. In: *Proc., 1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 1999), 19.–23. September, Atlanta, Georgia, USA*, IEEE Press, 1999.

[151] KONNO, A.; NAGASHIMA, K.; FURUKAWA, R.; NISHIWAKI, K.; NODA, T.; INABA, M.; INOUE, H.: Development of a Humanoid Robot Saika. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems*, S. 805–810, 1997.

[152] KONNO, A.; NISHIWAKI, K.; FURUKAWA, R.; TADA, M.; NAGASHIMA, K.; INABA, M.; INOUE, H.: Dexterous Manipulations of Humanoid Robot Saika. In: *Proc., 5th International Symposium on Experimental Robots*, S. 46–57, 1997.

[153] KONNO, A.; TADA, M.; NAGASHIMA, K.; INABA, M.; INOUE, H.: Development of a 3-Fingered Hand and Grasping Unknown Objects by Groping. In: *Proc., IEEE International Symposium on Assembly and Task Planning*, S. 72–77, 1997.

[154] KRAFT, D.: *Kraftregelung eines Leichtbauarmes am Beispiel des humanoiden Roboters ARMAR.* Studienarbeit, Universität Karlsruhe, 2005.

[155] KRATZ, R.; KLUG, S.; STELZER, M.; VON STRYK, O.: Biologically Inspired Reflex Based Stabilization Control of a Humanoid Robot with Artificial Muscles. In: *Proc., IEEE International Conference on Robotics and Biomimetics (ROBIO)*, S. 1089–1094, 2006.

[156] KRATZ, R.; STELZER, M.; FRIEDMANN, M.; VON STRYK, O.: Control Approach for a Novel High Power-to-Weight Ration Actuator Scalable in Force and Length for Robots. In: *Proc., IEEE/ASME International Conference in Advanced Intelligent Mechatronics*, 2007.

[157] KREBS, V.; SCHNIEDER, E.: Special Issue: Hybrid Systems I: Modeling and Control. *at - Automatisierungstechnik* 9 (2000), S. 413–468.

[158] KREBS, V.; SCHNIEDER, E.: Special Issue: Hybrid Systems II: Analysis, Modeling, and Verification. *at - Automatisierungstechnik* 2 (2001), S. 49–104.

[159] KUFFNER, J.; KAGAMI, S.; INABA, M.; INOUE, H.: Dynamically-stable Motion Planning for Humanoid Robots. In: *Proc., IEEE International Conference on Humanoid Robotics*, 2000.

[160] KUFFNER, J.; KAGAMI, S.; INABA, M.; INOUE, H.: Graphical Simulation and High-Level Control of Humanoid Robots. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems*, Bd. 3, S. 1943–1948, 2000.

[161] KUHLMANN, A.: *Einführung in die Sicherheitswissenschaft.* Vieweg, Wiesbaden, 1981.

[162] KUNIYOSH, Y.; NAGABUKO, A.: Humanoid as a Research Vehicle Into Flexible Complex Interaction. In: *Proc., International Conference on Humanoid Robots*, 1997.

[163] KUNTZE, H.-B.; FREY, C.; GIESEN, K.; MILIGHETTI, G.: On a Smart Structure Variable Supervisory Control Concept for Humanoid Robots. In: *Humanoids 2003 International Conference on Humanoid Robots, Karlsruhe and Munich (Conference Documentation and CD-ROM)*, 2003.

[164] KUNTZE, H. B.; FREY, C. W.; GIESEN, K.: Fault Tolerant Supervisory Control of Human Interactive Robots. In: *Proc., IAR Annual Meeting, ACD Workshop*, S. 55–60, 2003.

[165] KUROKI, Y.; FUJITA, M.; ISHIDA, T.; NAGASAKA, K.; YAMAGUCHI, J.: A Small Biped Entertainment Robot Exploring Attractive Applications. In: *Proc., IEEE International Conference on Robotics and Automation*, S. 471–476, 2003.

[166] KUROKI, Y.; ISHIDA, T.; YAMAGUCHI, Y.; FUJITA, M.; DOI, T.: A Small Biped Entertainment Robot. In: *Proc., IEEE-RAS International Conference on Humanoid Robots*, S. 181–186, 2001.

[167] LABORIE, P.; GHALLAB, M.: Planning with Sharable Resource Constraints. In: *Proc., International Joint Conference on Artificial Intelligence*, S. 1643–1649, 1995.

[168] LANGE, F.: *Adaptiv vorausplanende Steuerung für schnelle sensorbasierte Roboterbewegungen.* Dissertation, Universität Karlsruhe, 2003.

[169] LATOMBE, J.: *Robot motion planning*. Kluwer Academic Publishers, 1991.

[170] LEHMANN, A.; MARTIN, J.; MIKUT, R.; BRETTHAUER, G.: Konzepte zur Regelung und Stabilitätsüberwachung beim Greifen mit flexiblen Robotergreifern. In: *Proc., 13. Workshop Fuzzy Systeme*, S. 79–98, Forschungszentrum Karlsruhe (FZKA 6900), 2003.

[171] LEHMANN, A.; MIKUT, R.; ASFOUR, T.: Petri-Netze zur Aufgabenüberwachung in humanoiden Robotern. In: *Proc., 15. Workshop Computational Intelligence*, S. 157–171, Universitätsverlag Karlsruhe, 2005.

[172] LEHMANN, A.; MIKUT, R.; ASFOUR, T.: Petri Nets for Task Supervision in Humanoid Robots. In: *Proc., 37th International Symposium on Robotics (ISR 2006), München*, S. 71–73, 2006.

[173] LEHMANN, A.; MIKUT, R.; MARTIN, J.; BRETTHAUER, G.: Online-Stabilitätsüberwachung strukturvariabler Roboterregelungen. In: *Proc., Robotik 2004*, S. 55–62, Düsseldorf: VDI-Verlag, 2004.

[174] LEHMANN, A.; MIKUT, R.; OSSWALD, D.: Low-Level Finger Coordination for Compliant Anthropomorphic Robot Grippers. In: *Proc., 44th IEEE Conference on Decision and Control, European Control Conference (CDC-ECC '05)*, S. 8319–8324, 2005.

[175] LIEBERT, B.: Industrierobotik, Quo Vadis? *VDI-Berichte Nr. 1841, Deutsche Gesellschaft für Robotik (DGR)* (2004), S. 1–3.

[176] LIM, H.; TANIE, K.: Human Safety Mechanism of Human-Friendly Robots: Passive Viscoelastic Trunk and Passively Movable Base. *International Journal of Robotic Research* 19(4) (2000), S. 307–335.

[177] LOEFFLER, K.; GIENGER, M.; PFEIFFER, F.: Sensor and Control Design of a Dynamically Stable Biped Robot. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems*, S. 484–490, 2003.

[178] LOOSE, T.: *Konzept für eine modellgestützte Diagnostik mittels Data Mining am Beispiel der Bewegungsanalyse*. Dissertation, Universität Karlsruhe, Universitätsverlag Karlsruhe, 2004.

[179] LOTZ, T.: *Modellbildung und Reglerentwurf für die Fingergelenke einer künstlichen Hand mit hydraulischen Fluidaktoren*. Diplomarbeit, Universität Karlsruhe (TH), 2003.

[180] LOVCHIK, C. S.; DIFTLER, M. A.: The Robonaut Hand: A Dexterous Robot Hand for Space. In: *Proc., IEEE ICRA*, S. 907–913, Detroit, 1999.

[181] MAES, P.: The Dynamics of Action Selection. In: *Proc., IJCAI*, S. 991–997, 1989.

[182] MARIOTTINI, G.; ALUNNO, E.; PRATTICHIZZO, D.: The Epipolar Geometry Toolbox (EGT) for MATLAB (2004).

[183] MARTIN, J.: *Ein Beitrag zur Integration von Sensoren in eine anthropomorphe künstliche Hand mit flexiblen Fluidaktoren*. Dissertation, Universität Karlsruhe, Universitätsverlag Karlsruhe, 2004.

[184] MARTIN, J.; BECK, S.; LEHMANN, A.; MIKUT, R.; PYLATIUK, C.; SCHULZ, S.; BRETTHAUER, G.: Sensors, Identification and Low Level Control of a Flexible Anthropomorphic Robot Hand. *International Journal of Humanoid Robotics* 1(3) (2004), S. 517–532.

[185] MASON, M.: Compliance and Force Control for Computer Controlled Manipulators. *IEEE Transaction on Systems, Man-Machine and Cybernetics* (1986), S. 418–432.

[186] MATARIC, M. J.: Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior. *Journal of Experimental and Theoretical Artificial Intelligence, Special Issue on Software Architectures for Physical Agents, Bd. 9, Nr. 2-3* (1997), S. 323–336.

[187] MATARIC, M. J.: Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents. *Journal of Cognitive Systems Research* 2(1) (2001), S. 81–93.

[188] MATHWORKS, INC.: Matlab: The Language of Technical Computing. 1998.

[189] MATTHES, J.; MIKUT, R.; VOGES, U.: Trackingregelung für Endoskopführungssysteme in der minimal invasiven Chirurgie. *atp - Automatisierungstechnische Praxis* 42(11) (2000), S. 60–66.

[190] MCALLESTER, D. R. D.: Systematic Non-linear Planning. In: *Proc., International Conference on Artificial Intelligence*, S. 634–639, 1991.

[191] MCDERMOTT, H. J.; MCKAY, C. M.; VANDALI, A. E.: A New Portable Sound Processor for the University of Melbourne / Nucleus Limited Multichannel Cochlear Implant. *Journal of the Acoustic Society of America* 91 (1992), S. 3367–3371.

[192] MIKUT, R.: *Modellgestützte on-line Stabilitätsüberwachung komplexer Systeme auf der Basis unscharfer Ljapunov-Funktionen*. Dissertation, Universität Karlsruhe, VDI-Verlag, Düsseldorf, 1999.

[193] MIKUT, R.; BRETTHAUER, G.: The Evaluation of Disturbed Control Systems by Means of Fuzzy Supervision. In: *Proc., 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99) (CD-ROM)*, Aachen, 1999.

[194] MIKUT, R.; BRETTHAUER, G.; BINDEL, T.: On-Line Stability Supervision with a Hierarchical Fuzzy Concept and Fuzzy-Lyapunov Functions. In: *Proc., 3rd European Congress on Intelligent Techniques and Soft Computing (EUFIT'95)*, S. 775–780, Aachen, 1995.

[195] MIKUT, R.; BURMEISTER, O.; REISCHL, M.; LOOSE, T.: Die MATLAB-Toolbox Gait-CAD. In: *Proc., 16. Workshop Computational Intelligence*, S. 114–124, Universitätsverlag Karlsruhe, 2006.

[196] MIKUT, R.; JÄKEL, J.; GRÖLL, L.: Interpretability Issues in Data-Based Learning of Fuzzy Systems. *Fuzzy Sets and Systems* 150(2) (2005), S. 179–197.

[197] MIKUT, R.; LEHMANN, A.; BRETTHAUER, G.: Fuzzy Stability Supervision of Robot Grippers. In: *Proc., IEEE International Conference on Fuzzy Systems, Budapest*, S. 1473–1478, 2004.

[198] MILGHETTI, G.; KUNTZE, H.-B.: On the Discrete-Continuous Control of Basic Skills for Humanoid Robots. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and System (IROS), Bejing, China*, 2006.

[199] MILIGHETTI, G.; KUNTZE, H.-B.: Multi-Sensor Robot Control for Two-Arm Humanoid Skills. In: *Proc. 37th International Symposium on Robotics (ISR 2006), München*, S. 65–67, 2006.

[200] MILUTINOVIC, D.; LIMA, P.: Petri Net Models of Robotic Tasks. In: *Proc., IEEE International Conference on Robotics and Automation (ICRA)*, S. 4059 – 4064, 2002.

[201] MIRZA, K.; HANES, M.; ORIN, D.: Dynamic Simulation of Enveloping Power Grasps. *IEEE Journal of Robotics and Automation* (1993), S. 430–435.

[202] MOUNIER, S.: *Entwicklung einer realitätsnahen Kraftrückkopplung bei fluidisch betriebenen Handprothesen*. Dissertation, Universität Karlsruhe, Forschungszentrum Karlsruhe (FZKA 7004), 2004.

[203] MSC SOFTWARE, C.: Running and Configuring MSC.ADAMS 2005 r2. 2005.

[204] NENNINGER, G. M.: *Modellbildung und Analyse hybrider dynamischer Systeme als Grundlage für den Entwurf hybrider Steuerungen*. Dissertation, Universität Karlsruhe, VDI Verlag, Düsseldorf, 2001.

[205] NEVIS, J.; WITHNEY, D.: Assembly Research. *The Industrial Robot* 7(1) (1980), S. 27–43.

[206] NILSSON, H. J.: Shakey the Robot. *Technical Note 323, AI Center, SRI International, Menlo Park, CA* (1984).

[207] NISHIWAKI, K.; SUGIHARA, S.; KAGAMI, S.; KANEHIRO, M.; INABA, M.; INOUE, H.: Design and Development of Research Platform for Perception-Action Integration in Humanoid Robot: H6. In: *Proc., IEEE/RSJ International Conference on Intelligent Robots and Systems*, S. 1559–1564, 2000.

[208] N.N.: Projektierung mit Harmonic Drive Getrieben. Techn. Ber., Harmonic Drive AG.

[209] N.N.: *Deutsche Risikostudie Kernkraftwerke Phase B*. Gesellschaft für Reaktorsicherheit, Verlag TÜV Rheinland, Köln, 1990.

[210] NORTHRUP, S.; SARKAR, K.; KAWAMURA, K.: Biologically-Inspired Control Architecture for a Humanoid Robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2001).

[211] ÖGÜT, Ö.: *Entwicklung einer Testumgebung zur echtzeitfähigen Regelung und Überwachung von Greifvorgängen eines flexiblen Robotergreifers*. Studienarbeit, Universität Karlsruhe, 2004.

[212] ORTH, P.; ABEL, D.: Rapid Control Prototyping petrinetzbasierter Steuerungen mit dem Tool NETLAB. *at - Automatisierungstechnik* 54 (2006).

[213] OSSWALD, D.: MCA-Petri: Eine Petri-Netz-Implementierung Für MCA2. Interner Bericht, Universität Karlsruhe, 2005.

[214] OSSWALD, D.; MARTIN, J.; BURGHART, C.; MIKUT, R.; WÖRN, H.; BRETTHAUER, G.: Integrating a Robot Hand into the Control System of a Humanoid Robot. *Robotics and Autonomous Systems* 48(4) (2004), S. 213–221.

[215] OTTO, C.; SCHÄFFER, A.; KUGI, A.; HIRZINGER, G.: Decoupling Based Cartesian Impedance Control of Flexible Joint Robot. In: *Proc., IEEE International Conference on Robotic and Automation*, 2003.

[216] OZAKI, H.; MOHRI, A.; TAKATA, M.: On the Force Feedback Control of a Manipulator with a Compliant Wrist Force Sensor. *Mechanism and Machine Theory* 18 (1983), S. 57–62.

[217] PACK, R. T.: *IMA: The Intelligent Machine Architecture*. Dissertation, Vanderbilt University, 1999.

[218] PALIS, F.; RUSIN, V.: Adaptive Impedanzregelung von Robotersystemen mit überlagerten mechanischen Verkopplungen. In: *Proc., 9. Symposium Maritime Elektronik - Universität von Rostock*, 1998.

[219] PARDOWITZ, M.; ZÖLLNER, R.; DILLMANN, R.: Learning Sequential Constraints of Tasks from User Demonstrations. In: *Proc., IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, S. 424–429, 2005.

[220] PAUL, R.: Robot Manipulators: Mathematics, Programming, and Control. *MIT Press* (1981).

[221] PETRI, C. A.: *Kommunikation mit Automaten*. Dissertation, Universität Bonn, 1962.

[222] RAIBERT, M.; CRAIG, J.: Hybrid Position / Force Control of Manipulators. *Transaction of the ASME, J. Dynamic Systems, Measurement and Control* 102 (1981), S. 126–133.

[223] RAY, K. S.; MAJUMDER, D. D.: Application of Circle Criteria for Stability Analysis of Linear SISO and MIMO Systems Associated with Fuzzy Logic Controller. *IEEE Transactions on Systems, Man, and Cybernetics* 14(2) (1984) 2, S. 345–349.

[224] READMANN, M.: Flexible Joint Robots. *CRC Press* (1994).

[225] REINHARDT, H.: *Automatisierungstechnik - Theoretische und gerätetechnische Grundlagen, SPS*. Berlin: Springer, 2007.

[226] REINISCH, K.: Systemanalyse und Steuerung komplexer Prozesse: Probleme, Lösungswege und nichtindustrielle Anwendungen. *MSR* 29(5) (1986), S. 194–207.

[227] REINSCHKE, K.: *Zuverlässigkeit von Systemen*. Berlin: Verlag der Technik, 1973.

[228] REISCHL, M.: *Ein Verfahren zum automatischen Entwurf von Mensch-Maschine-Schnittstellen am Beispiel myoelektrischer Handprothesen*. Dissertation, Universität Karlsruhe, Universitätsverlag Karlsruhe, 2006.

[229] RÖFER, T.; LANKENAU, A.: Architecture and Applications of the Bremen Autonomous Wheelchair. In: *Proc., 4th Joint Conference on Information Systems*, S. 365–368, 1998.

[230] ROGERS, T.; WILKES, M.: The Human Agent: A Work in Progress Toward Human-Humanoid Interaction. *IEEE Transactions on Systems, Man and Cybernetics* 32(2) (2002), S. 864–869.

[231] ROJO GUERRA, D.: *Weiterentwicklung des Halsgelenkes eines humanoiden Roboters mit vier Freiheitsgraden*. Diplomarbeit, FH Karlsruhe, 2004.

[232] ROSENBLATT, J. K.: DAMN: A Distributed Architecture for Mobile Navigation. In: *Proc., AAAI Spring Symposium on Lessons Learned from Implememted Software Architectures for Physical Agents*, S. 167–178, 1995.

[233] ROSHEIM, M. E.: *Robot Evolution: The Development of Anthrobotics*. Wiley-IEEE, 1994.

[234] RÖSSLER, P.; HANEBECK, U. D.: Telepresence Techniques for Exception Handling in Household Robots. In: *Proc., IEEE International Conference on Systems, Man and Cybernetics (SMC´04),*, S. 53–58, 2004.

[235] RUSSEL, S.; NORVIG, P.: *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

[236] SAKAGAMI, Y.; WATANABE, R.; AOYAMA, C.; MATSUNAGA, S.; HIGAKI, N.; FUJIMURA, K.: The intelligent ASIMO: System overview and integration. *Intelligent Robots and System, 2002. IEEE/RSJ International Conference* 3 (2002), S. 2478 – 2483.

[237] SAUTER, D.; DUBOIS, G.; LEVRAT, E.; BRÉMONT, J.: Fault Diagnosis in Systems Using Fuzzy Logic. In: *Proc., TOOLDIAG '93*, 1993.

[238] SCHABLOWSKI-TRAUTMANN, M.: *Konzept zur Analyse der Lokomotion auf dem Laufband bei inkompletter Querschnittlähmung mit Verfahren der nichtlinearen Dynamik*. Dissertation, Universität Karlsruhe, Universitätsverlag Karlsruhe, 2006.

[239] SCHLEGL, T.: *Diskret-kontinuierliche Regelung mehrfingriger Roboterhände zur robusten Manipulation von Objekten*. Dissertation, TU München, VDI-Verlag, Düsseldorf, 2002.

[240] SCHLEMMER, M.; HILLER, M.: Online Verfahren zur Bahnplanung kinematisch redundanter Manipulatoren. *Zeitschrift für Angewandte Mathematik und Mechanik* 3 (2001), S. 175–184.

[241] SCHMIDT, R.; THEWS, G.: *Physiologie des Menschen*. Berlin: Springer, 1993.

[242] SCHNEIDER, H.: Implementation of a Fuzzy Concept for Supervision and Fault Detection of Robots. In: *Proc., 1st European Congress on Fuzzy and Technologies*, S. 775–780, 1993.

[243] SCHNEIDER, J.: *Risiko und Sicherheit technischer Systeme*. Birkhäuser Verlag, Basel, 1991.

[244] SCHOLL, K.-U.: Modular Controller Architecture Version 2 (MCA2).

[245] SCHRAFT, R. D.; GRAF, A.; TRAUB, D.; JOHN, D.: A Mobile Robot Platform for Assistance and Entertainment. *Industrial Robot Journal* 28 (2001), S. 29–34.

[246] SCHRÖDER, J.; STEINHAUS, P.; GOCKEL, T.; DILLMANN, R.: Design of a Holonomous Platform for a Humanoid Robot using MCA2. In: *Proc., 8th Conference on Intelligent Autonomous Systems*, S. 836–843, 2004.

[247] SCHULTE, H.: The Characteristics of the McKibben Artificial Muscle. *The Application of External Power in Prosthetics and Orthotics* (1961), S. 94–115.

[248] SCHULZ, S.: *Eine neue Adaptiv-Hand-Prothese auf der Basis flexibler Fluidaktoren*. Dissertation, Universität Karlsruhe, Shaker-Verlag, 2004.

[249] SCHULZ, S.; PYLATIUK, C.; BRETTHAUER, G.: A New Class of Flexible Fluidic Actuators and their Applications in Medical Engineering. *at - Automatisierungstechnik* 47(8) (1999), S. 390–395.

[250] SCHULZ, S.; PYLATIUK, C.; BRETTHAUER, G.: Entwicklung einer künstlichen Hand mittels flexibler Fluidaktoren. *Proc., GMA-Kongress (VDI Bericht Nr. 1680)* (2001), S. 143ff.

[251] SCHULZ, S.; PYLATIUK, C.; BRETTHAUER, G.: A New Ultralight Anthropomorphic Hand. In: *International Conference on Robotics and Automation, Seoul, Korea*, 2001.

[252] SCHULZ, S.; PYLATIUK, C.; MARTIN, J.; BRETTHAUER, G.: Die Anthropomorphe FZK-Hand. In: *Proc., Robotik 2002 (VDI-Berichte 1679)* (DILLMANN, R.; SCHRAFT, R.; WÖRN, H., Hg.), S. 531–536, VDI/VDE- Gesellschaft Mess- und Automatisierungstechnik, 2002.

[253] SCHULZ, S.; PYLATIUK, C.; REISCHL, M.; MARTIN, J.; MIKUT, R.; BRETTHAUER, G.: A Lightweight Multifunctional Prosthetic Hand. *Robotica* 23(3) (2005), S. 293–299.

[254] SCIAVICCO, L.; SICILIANO, B.: *Modelling and Control of Robot Manipulators*. London: Springer, 2001.

[255] SHIBATA, S.; INOOKA, H.: Psychological Evaluation of Robotic Motions. *International Journal of Industrial Ergonomics* 21 (1998), S. 483–494.

[256] SIMMONS, R. G.: Structured Control for Autonomous Robots. *IEEE Transactions on Robotics and Automation* 10(1) (Feb. 1994), S. 34–43.

[257] SOFGE, D.; PERZANOWSKI, D.; SKUBIC, M.; BUGAJSKA, M.; TRAFTON, J. G.; CASSIMATIS, N.; BROCK, D.; ADAMS, W.; SCHULTZ, A.: Cognitive Tools for Humanoid Robots in Space. In: *Proc., 16th IFAC Symposium on Automatic Control in Aerospace*, 2004.

[258] SOM, F.: Sichere Robotersteuerung für den personensicheren Betrieb ohne trennende Schutzeinrichtungen. *VDI Berichte 1841 Deutsche Gesellschaft für Robotik (DGR)* (2004), S. 745–753.

[259] SOMMER, H. J.; HAHN, H.: Ein einfaches Verfahren zum Test der BIBO-Stabilität bei Systemen mit Polynomialen Nichtlinearitäten und Fuzzy-Komponenten. In: *Proc., 3. Workshop Fuzzy-Control (GMA-UA 1.4.2)*, S. 13–26, 1993.

[260] SPONG, M.: Modelling and Control of Elastic Joint Robots. *IEEE Journal of Robotics and Automation* (1987), S. 291–300.

[261] STARKE, P.: *Analyse von Petri-Netz-Modellen*. Stuttgart: B.G. Teubner, 1990.

[262] STEINHORST, W.: *Sicherheitstechnische Systeme: Zuverlässigkeit und Sicherheit kontrollierter und unkontrollierter Systeme*. Vieweg, Braunschweig, 1999.

[263] STIEFELHAGEN, R.: Tracking Focus of Attention in Meetings. In: *Proc. IEEE International Conference on Multimodal Interfaces*, S. 273–280, 2002.

[264] STIEFELHAGEN, R.; FUEGEN, P.; GIESELMANN, P.; HOLZAPFEL, H.; NICKEL, K.; WAIBEL, A.: Natural Human-Robot Interaction using Speech, Gaze and Gestures. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2004).

[265] STOPP, A.; BALDAUF, S.; HORSTMANN, S.; KRISTENSEN, S.: A Safety Concept for Robot Assistants in Manufacturing. *VDI Berichte 1841 Deutsche Gesellschaft für Robotik (DGR)* (2004), S. 753–761.

[266] STOPP, A.; BALDAUF, T.; HANTSCHE, R.; HORSTMANN, S.; KRISTENSEN, S.; LOHNERT, F.; PRIEM, C.; RÜSCHER, B.: The Manufacturing Assistant: Safe, Interactive Teaching of Operation Sequences. In: *Proc., IEEE International Workshop on Robot-Human Interactive Communication*, S. 386–391, 2002.

[267] STROBEL, H.: *Experimentelle Systemanalyse*. Akademie Verlag, 1975.

[268] STRUBE, G.: *Wörterbuch der Kognitionswissenschaft*. Klett-Cotta Verlag, 1996.

[269] SUGANO, S.; KATO, I.: WABOT-2: Autonomous Robot with Dexterous Finger-Arm–Finger-Arm Coordination Control in Keyboard Performance. In: *Proc., International Conference on Robotics and Automation*, S. 90–97, 1987.

[270] TATE, A.; DRABBLE, B.; KIRBY, R.: *O-Plan2: An Architecture for Command, Planning and Control*. Morgan-Kaufmann Publishing, 1994.

[271] THOMALLA, C.: An Approach for Detection of Exceptional Situations and Recovery in Robot-Human Cooperation. In: *Proc., 3rd International Symposium on Robotics and Automation*, 2002.

[272] TRAVER, V.; DEL POBIL, A.; PEREZ-FRANCISCO, M.: Smart Safe Strategies for Service Robots Interacting with People. *Proc., 6th International Conference on Intelligent Autonomous Systems* (2000), S. 323–330.

[273] TRIESCH, J.; VON DER MALSBURG, C.: Robust Classification of Hand Postures against Complex Backgrounds. In: *Proc., 2nd International Conference on Automatic Face Recognition and Gesture Recognition 1996, Killington, Vermont, USA*, 1996.

[274] TURING, A. M.: Computing Machinery and Intelligence. *Mind* 59 (1950), S. 433–460.

[275] UDE, A.; ATKESON, M.; RILEY, M.: Automatic Generation of Kinematic Models for the Conversion of Human Motion Capture Data in Humanoid Robot Motion. In: *Proc., IEEE-RAS International Conference on Humanoid Robots*, 2000.

[276] UDE, A.; ATKESON, M.; RILEY, M.: Planning of Joint Trajectories for Humanoid Robots using B-Spline Wavelets. In: *Proc., IEEE International Conference on Robotics and Automation*, S. 2223–2228, 2000.

[277] VETTER, T.; WEBER, H.; WEHHOFER, J.: Erhöhte Wirtschaftlichkeit in der Qualitätsprüfung von Elektromotoren. *QZ Qualität und Zuverlässigkeit* 5 (1991).

[278] VISINSKY, M. L.; CAVALLARO, J. R.; WALKER, I. D.: Robotic Fault Detection and Fault Tolerance: A survey. *Reliability Engineering and System Safety* 46 (1994), S. 139–158.

[279] WANG, F.-Y.; SARIDIS, G. N.: A Coordination Theory for Intelligent Machines. *Automatica* 26(5) (1991), S. 833–844.

[280] WANG, L.-X.: Stable Adaptive Fuzzy Control of Nonlinear Systems. *IEEE Transactions on Fuzzy Systems* 1 (1993) 2, S. 146–155.

[281] WELD, D. S.: An Introduction to Least Commitment Planning. In: *Proc., International Conference on Artificial Intelligence*, 1994.

[282] WERNSTEDT, J.: *Experimentelle Prozeßanalyse*. München: Oldenbourg, 1989.

[283] WILLISKY, A. S.: A Survey of Design Methods for Failure Detection in Dynamic Systems. *Automatica* 12 (1976), S. 601–611.

[284] WILLISKY, A. S.; JONES, H. L.: A Generalized Likelihood Ratio Testing Approach for Detection and Estimation of Junps in Linear Systems. *IEEE Transactions on Automation and Control* 21 (1976), S. 108–112.

[285] WOLF, S.; LOOSE, T.; SCHABLOWSKI, M.; DÖDERLEIN, L.; RUPP, R.; GERNER, H. J.; BRETTHAUER, G.; MIKUT, R.: Automated Feature Assessment in Instrumented Gait Analysis. *Gait & Posture* 23(3) (2006), S. 331–338.

[286] WÜNNENBERG, J.: *Observer-Based Fault Detection in Dynamic Systems*. Dissertation, Universität Duisburg, 1990.

[287] YAMADA, Y.; SUITA, K.: A Failure-to-Safety Robot System for Human-Robot-Coexistance. *Robotics and Autonomous Systems* 18 (1996), S. 283–291.

[288] YIGIT, S.; BURGHART, C.; WÖRN, H.: Specific Combined Control Mechanisms for Human Robot Co-operation. In: *Proc., 15th. International Ship and Offshore Structures Congress*, 2003.

[289] YIGIT, S.; BURGHART, C.; WÖRN, H.: Applying Reflexes to Enhance Safe Human-Robot-Co-Operation with a Humanlike Robot Arm. *Proc., 35th International Symposium on Robotics* (2004).

Bildverzeichnis

1.1 Direktes und inverses kinematisches Problem . . . 3
1.2 Schematische Darstellung einer deliberativen Steuerungsarchitektur (modifiziert) . . . 4
1.3 Schematische Darstellung einer reaktiven Steuerungsarchitektur (modifiziert) . . . 5
1.4 Schematische Darstellung einer verhaltensbasierten Steuerungsarchitektur (modifiziert) . 6
1.5 Schematische Darstellung einer hybriden Steuerungsarchitektur (modifiziert) . . . 7
1.6 Einordnung verschiedener Steuerungsarchitekturen (modifiziert) . . . 9
1.7 Prozess der Sicherheitsüberwachung (modifiziert) . . . 10
1.8 Übersicht über Verfahren zur Fehlerdiagnose . . . 11
1.9 Humanoide Roboter des Sonderforschungsbereichs 588 . . . 17

2.1 Kognitive Steuerungsarchitektur des Sonderforschungsbereichs 588 . . . 22
2.2 Detailansicht einer möglichen Variante der Aufgabenplanung . . . 23
2.3 Detailansicht der Aufgabenkoordination . . . 24
2.4 Detailansicht der Aufgabenausführung . . . 24
2.5 Detailansicht der diskret-kontinuierlichen Systemstruktur für Elementaroperationen . . . 28
2.6 Detailansicht der diskret-kontinuierlichen Systemstruktur für Elementaraktionen . . . 29
2.7 Beispiel für ein modulares, hierarchisches Petri-Netz . . . 33
2.8 Beispiele für platzberandete Teilnetze . . . 35
2.9 Koordinierungsnetz der Aktionsfolge ’Greif- und Transportvorgang’ . . . 36
2.10 Kognitive Steuerungsarchitektur erweitert um Überwachungskomponenten . . . 38
2.11 Allgemeine Struktur der modellbasierten Fehlerdiagnose . . . 39
2.12 Modellstruktur der Fuzzy-Überwachung . . . 42
2.13 Beispiele für Zugehörigkeitsfunktionen zur Beurteilung des Stabilitätsgrads . . . 43
2.14 Beispiel zur Umsetzung des neuen Korridorkonzepts . . . 45
2.15 Modellstruktur der regelbasierten Auswertung zur Stabilitätsüberwachung . . . 46
2.16 Beispiele für platzberandete Teilnetze mit integrierter Ausnahmesituationsbehandlung . . 48
2.17 Verwaltungsnetz . . . 49
2.18 Beispiel für eine Aktion mit drei ”virtuellen” Unterstellen zur Ausnahmebehandlung . . 51

3.1 Zusammenhänge zwischen Entwicklungs- und Zielplattform . . . 59
3.2 Testumgebung zum Entwurf, zur Simulation und zur Analyse von Petri-Netzen . . . 61
3.3 Umsetzung der Roboter-Steuerungsarchitektur . . . 62
3.4 Statusnetz . . . 63
3.5 Beispiele für MCA-Komponenten . . . 65
3.6 Visualisierung des Robotergreifers mit MCAGUI und MCAadmin . . . 66
3.7 Schematische Darstellung der Portierungsstrategie . . . 67
3.8 Eingangs- und Ausgangsgrößen des hybriden Kraft-Positionsregelungsalgorithmus . . . 67
3.9 Umsetzung der Regler-Testumgebung auf der Entwicklungsplattform . . . 68
3.10 Umsetzung der Regler-Testumgebung auf der Zielplattform . . . 69

4.1 Kinematik des Robotergreifers mit wichtigen Koordinatensystemen 72
4.2 Teilsysteme des Antriebs für hydraulischen und pneumatischen Betrieb 73
4.3 Auftretende Einzelmomente am Aktor 74
4.4 Koordinatensysteme im Kontaktfall 78
4.5 Flächenkontakt und Punktkontakt des Robotergreifers mit einem verformbaren Objekt . 79
4.6 Freischnitte eines Fingers unter Einwirkung einer externen Kraft 80
4.7 Schematische Darstellung von Regler und Regelstrecke für den Robotergreifer 82
4.8 Detaillierte Struktur der Admittanzregelung 83
4.9 Detaillierte Struktur der hybriden Kraft-Positionsregelung 85
4.10 Zustandsautomat des Regelungsalgorithmus 87
4.11 Simulationsergebnisse für das Beispiel 'Greifergesten' (Szenario 1) 90
4.12 Simulationsergebnisse für das Beispiel 'Greifergesten' (Szenario 2) 90
4.13 Simulationsergebnisse für das Beispiel 'Greifergesten' (Szenario 3) 91
4.14 Umsetzung eines Präzisionsgriffs auf der Entwicklungsplattform 92
4.15 Simulationsergebnisse für das Beispiel 'Greifvorgang' 94
4.16 Bewegungen des menschlichen Halses 96
4.17 Kinematisches Grundkonzept der Roboterhalseinheit 97
4.18 Ersatzschaltbild des Motors mit Last 97
4.19 Vergleich von Messung und Simulation für das untere Nicken 101
4.20 Vergleich von Messung und Simulation für das Neigen 101
4.21 Vergleich von Messung und Simulation für das Drehen 101
4.22 Vergleich von Messung und Simulation für das obere Nicken 102
4.23 Modell der Lochkamera 103
4.24 Schematische Darstellung von Regler und Regelstrecke für die Roboterhalseinheit 104
4.25 Ergebnis der experimentellen Evaluierung der Roboterhalseinheit (Szenario 1) 104
4.26 Ergebnis der experimentellen Evaluierung der Roboterhalseinheit (Szenario 2) 105
4.27 Schematische Darstellung von Regler und Regelstrecke für die Roboterhalseinheit 105
4.28 Simulationsergebnisse für das Beispiel 'Kopfgesten' 107
4.29 Umsetzung der 'Aufmerksamkeitssteuerung' auf der Entwicklungsplattform 109
4.30 Simulationsergebnisse für das Beispiel 'Aufmerksamkeitssteuerung' 110

5.1 Simulationsergebnisse zur Stabilitätsüberwachung (Szenario 1) 115
5.2 Simulationsergebnisse zur Stabilitätsüberwachung (Szenario 2) 116
5.3 Simulationsergebnisse zur Stabilitätsüberwachung (Szenario 3) 117
5.4 Simulationsergebnisse zur Stabilitätsüberwachung (Szenario 4) 118
5.5 Aufbau der Versuchs- und Entwicklungsplattform am Fraunhofer Institut IITB [72] . . . 118
5.6 Experimentelle Ergebnisse zur Stabilitätsüberwachung 119
5.7 Simulationsergebnisse zur Hardwareüberwachung (Pumpendefekt) 121
5.8 Simulationsergebnisse zur Hardwareüberwachung (Leckage) 122
5.9 Simulationsergebnisse zur Problemlösung (Szenario 1) 124
5.10 Simulationsergebnisse zur Problemlösung (Szenario 2) 125
5.11 Simulationsergebnisse zur Problemlösung (Szenario 3) 126
5.12 Simulationsergebnisse zur Problemlösung (Szenario 4) 126
5.13 Simulationsergebnisse zur Problemlösung (Szenario 5) 127
5.14 Simulationsergebnisse zur Problemlösung (Szenario 6) 128
5.15 Simulationsergebnisse zur Problemlösung (Szenario 7) 129
5.16 Simulationsergebnisse zur Prioritätsverwaltung (Szenario 1) 130
5.17 Simulationsergebnisse zur Prioritätsverwaltung (Szenario 2) 131

5.18 Simulationsergebnisse zur Prioritätsverwaltung (Szenario 3) 131

A.1 Beispiel Petri-Netz mit zwei Plätzen und einer Transition 135
A.2 Beispiele zur Modularisierung von Petri-Netzen 136
A.3 Beispiel zur Verzweigung, Synchronisation und Zusammenführung von Petri-Netzen . . 136

B.1 Position und Ausrichtung eines Objektes im Raum 137
B.2 DENAVIT-HARTENBERG-Parameter 139

Tabellenverzeichnis

1.1 Vergleichstabelle humanoider Roboter 18

2.1 Beispiele für Elementaraktionen (EA) von humanoiden Robotern 30
2.2 Abstraktionseinheiten zur Beschreibung von Handlungen mit Petri-Netzen 32
2.3 Hardware-Ressourcen für das Beispiel 'Greif- und Transportvorgang' 35
2.4 Fuzzy-Regelbasis zur Beurteilung des Stabilitätsgrads 43
2.5 Bedeutung der Plätze und Transitionen des Verwaltungsnetzes 50
2.6 Problemlösungsstrategien von humanoiden Robotern 53
2.7 Vorrangregeln für Transitionen im Verwaltungsnetz 55
2.8 Aktionsfolge 'Aufmerksamkeitssteuerung mit Ausnahmebehandlung' 55
2.9 Aktionsfolge 'koordiniertes Annähern mit Ausnahmebehandlung' 56

3.1 Bedeutung der Plätze und Transitionen des Statusnetzes 64

4.1 Vergleich zwischen menschlicher Hand und Robotergreifer 73
4.2 Varianten des Robotergreifers in Abhängigkeit von der Sensorausstattung 77
4.3 Sollpositionen der Freiheitsgrade für die Zeigegeste 89
4.4 Aktionsfolge 'Greifbewegung' für den Präzisionsgriff 93
4.5 Vergleich zwischen menschlicher Halsbeweglichkeit und Roboterbeweglichkeit 96
4.6 Maximale bzw. minimale Sollpositionen der drei untersuchten Kopfgesten 107
4.7 Plätze und Transitionen der Aktion 'Aufmerksamkeitssteuerung' 109

5.1 Parameter der analysierten Szenarien 114
5.2 Vergleich der analysierten Szenarien 123
5.3 Zeitpunkte eintretender Ereignisses und aktivierter Transitionen (Szenarien 1, 2) 124
5.4 Zeitpunkte eintretender Ereignisses und aktivierter Transitionen (Szenarien 3, 4) 125
5.5 Zeitpunkte eintretender Ereignisses und aktivierter Transitionen (Szenarien 5, 6) 127
5.6 Zeitpunkte eintretender Ereignisses und aktivierter Transitionen (Szenario 7) 128

B.1 DH-Notation 140

C.1 Kinematische Modellparameter des Greifers 141
C.2 Modellparameter der Antriebskomponenten des Greifers 141
C.3 Kinematische Modellparameter der Halseinheit 142
C.4 Modellparameter der Antriebskomponenten der Halseinheit 142

D.1 MATLAB-m-Files zum Entwurf, zur Analyse und zur Simulation von Petri-Netzen . . . 143
D.2 MATLAB-m-Files zur Trajektoriengenerierung, Regelung und Modellbildung 144
D.3 MATLAB-m-Files zur Fehlerdetektion und Problemlösung 144
D.4 C-Files zur Ausführung auf der Entwicklungs- und Zielplattform 145

Index

Überwachungskonzept, 37, 123

Aktion, 31, 50, 55
Aktionsfolge, 31, 33, 52, 54, 55, 126
aktive Modelle, 23
Aktorik, 38
allgemeine Gasgleichung, 74
Analyse
    Deadlock-, 26
    Erreichbarkeits-, 26
    Lebendigskeits-, 26
ASIMO, 14
ATLANTIS, 7
Aufgabenausführung, 24, 34
Aufgabenkoordination, 24, 33
Aufgabenkoordinierung, 130
Aufgabenplanung, 23, 25, 31, 54
Aufgabenspeicher, 24, 63
Aufgabenwissen, 23, 133
Aufmerksamkeitssteuerung, 35, 50, 108, 127
AuRa, 7
Ausnahmebehandlungsstrategie, 19
autonome Planerstellung, 31
Autonomie, 38

Bediener-Interface, 62, 89
Benutzereingriff, 49, 124, 129
Benutzersicherheit, 19
Bewegungserfassungssystem, 2
Bewertungsfunktion, 41
    korregierte und gefilterte, 41
    Korrektur der, 41
    mittlere, 41
Bewertungsmaß, 41

Center of Singleton Methode, 43
Cog, 16

DAMN, 6
Datenbank, 133
DB, 16
Defuzzifizierung, 43
Dekomposition, 54, 133
Denavit-Hartenberg-Notation, 137
Detektion, 10
DH-Parameter, 140
Dialog-Manager, 23
Direct force control, 82
Diskret-kontinuierliche Systemstruktur, 25
Drucksensorik, 77
Dynamik, 75, 99

Eigenautonomie, 44
Einfluss der Sensorausstattung, 77
Elektromotor, 97
Elementaraktion, 28, 55, 89, 108
Elementaroperation, 27
Entwicklungsplattform, 59, 62

Fähigkeiten
    kognitive, 14
    motorische, 13
Führungsgrößenintervall, 44, 46
Feder-Masse-System, 78
Fehler
    Antriebs-, 10
    Material-, 10
    Sensor-, 10
Fehlerdetektion, 54
    modellbasiert, 39, 113
Fehlerdiagnose, 10, 37
Fluidaktor, 74, 120
FMEA, 11
    Konstruktions-, 11
    Prozess-, 11
    System-, 11
FTA, 11
Fuzzy-Überwachungsebene, 12
Fuzzy-Mode-Selector, 28
Fuzzy-System, 43

Generierung von Aufgabenwissen, 25
globale Datenbank, 22
Graphentheorie, 11
Greifergesten, 88
Greifvorgänge, 92

H7, 14
Halseinheit, 95, 127
Hand-Automatik-Umschaltung, 27
Handlung, 25, 33
Hardwareüberwachung, 38, 120
Hardwarefehler, 54
Harmonic Drive Getriebe, 98
Harmonische Balance, 12
HERMES, 17
HUBO, 15
Hybride Kraft-Positionsregelung, 84
Hydraulik, 73

Indirect force control, 82
Industrierobotik, 12
Interaktion, 19
ISAC, 17
Isolation, 10
Istzustand, 25, 37

Jack, 16
Johnnie, 15

künstliche Intelligenz, 1, 5
Kamerasystem, 102
Kante, 26, 135
Kinematik, 72, 96
    direkte, 3
    inverse, 3
kompressibles Medium, 74
KONDISK, 12
Kontaktkraft, 78
Kontaktmodellierung, 79
Kooperation, 19
Koordinatensystem
    kartesisch, 3
Koordinierungsnetz, 34, 54, 93, 110
Kopfgesten, 106
Korridorkonzept, 45

Leckage, 120
Leichtbau, 47
Lernen durch Imitation, 2
linguistischer Wert, 43

Ljapunov
    -Funktion, 41
    -Theorie, 11

manuelles Teaching, 31
MCA2, 8
MCAadmin, 65
MCAGUI, 65
Mehrgrößensystem, 44
Methode der Kleinsten Quadrate, 71
modellbasierte Fehlerdiagnose, 38, 39
modellbasierte Regelung, 81, 103
Modellbildung, 71, 95
multimodal, 12

Nachgiebigkeit
    aktiv, 13
    passiv, 13
Nachgiebigkeitsregelung, 83
Netlab, 60
Netztopologie, 49
Neuinitialisierung des Integrators, 28

Objekteigenschaften, 78
Objektmasse, 78
Objektmodellierung, 78
Objektverformung, 79

Petri-Netz, 26, 60, 135
    Modularisierung, 31
    Synchronisation, 32, 136
    Verzweigung, 32, 136
    Zusammenführung, 32
Planer
    fallbasiert, 4
    generativ, 4
    IxTeT, 4
    O-Plan, 4
    POP, 4
    transformationell, 4
    XFRM, 4
Planrevision, 4
Platz, 26, 60, 135
Pneumatik, 74
Popov-Kriterium, 12
Portierungskonzept, 65
Positionsregelung, 103
    bildbasiert, 105
Prioritätsverwaltung, 24, 34, 64, 129
Problemlösung, 10, 37, 48, 54, 123

Problemlösungsstrategie, 52
Programmieren durch Vormachen, 2
Pumpe, 68, 120
Pumpenzustand, 87

Raum
    Gelenkwinkel-, 3, 103
    Konfigurations-, 3
Redundanz
    analytisch, 11
    wissensbasiert, 11
Refinement, 4
Reflex, 52
Reflexe, 113
regelbasierte Auswertung, 52
Regelung
    Admittanz-, 83
    Impedanz-, 83
    Steifigkeits-, 83
Regelungsalgorithmus, 127
Residuengenerator, 40
Residuengenerierung, 40
Ressource
    Hardware-, 24, 129
    Software-, 24
Ressourcenkonflikt, 129
Ressourcenverwaltung, 62, 110
Retraction, 4
Roboter-Funktionaleinheit, 26, 30, 31
Roboterassistenten, 13
Roboteraufgabe, 25
Robotergreifer, 71
Robotersystem
    autonomes, 111
    humanoides, 21
Routineablauf, 48

Saika, 14
SDR-4X, 15
sensomotorische Steuerung, 21
Sensorfehler, 47
Sensorik, 38
Sicherheitsüberwachung, 12
Sollzustand, 25
Stützstellen, 43
Stabilitätsüberwachung, 38, 40, 113
Stabilitätsgrad, 41, 46
Statusnetz, 62
Steuerspannung, 87
Steuerungsarchitektur
    deliberative, 3
    hybride, 7
    reaktive, 5
    verhaltensbasierte, 6
Stoßfreiheit, 27
STRIPS, 4
Subsumption Architecture, 5
Szenengraphen, 23

taktile Sensorik, 77
TCA, 7
Teilnetz
    platzberandet, 31
    transitionsberandet, 31
Teilsysteme des Antriebs, 73, 97
Telemanipulation, 52
Teleoperator, 52
Trajektoriengenerierung, 2
Transition, 26, 60, 135
Trend, 41

Umweltkarten, 23
Unterstelle, 50

Ventil, 68
Verfahren
    modellgestützt, 10
    signalgestützt, 10
Verfeinerung, 31, 136
Vergröberung, 31, 136
Verwaltungsnetz, 49

WABOT, 16
Winkelsensorik, 77

Zahnriementrieb, 99
Zeitüberschreitung, 54
Zielplattform, 65
Zielsystem, 59
Zugehörigkeitsfunktionen, 43
Zustandsraummethode, 11