Untersuchungen zur Sturzerkennung zweibeiniger Roboter mit Hilfe von Verfahren zur Mustererkennung

Diplomarbeit

zur Erlangung des Grades eines Diplom-Wirtschaftsingenieurs
der Wirtschaftswissenschaftlichen Fakultät
der Universität Hannover

vorgelegt von

Martin Bretschneider

Erstprüfer:	Prof. Dr.-Ing. W. Gerth Institut für Regelungstechnik, Fakultät für Elektrotechnik und Informatik
Betreuer:	Dipl.-Ing. O. Höhn (Institut für Regelungstechnik)
Hannover, den	10.11.2005

Nutzungsbedingungen

Diese Arbeit darf nur für folgende Zwecke unentgeltlich genutzt werden:

- das Lesen in PDF-Betrachtern und Ausdrucken durch natürliche Personen für die eigene nicht-kommerzielle Nutzung,
- die Indizierung in – auch kommerziellen – Suchmaschinen zur Abwicklung von Suchanfragen bei gleichzeitigem Verweis auf die unten genannte URL.

Verboten ist jegliche andere Nutzung besonders

- die kommerzielle Verwendung,
- das Verändern, Kopieren, Bereitstellen oder Verlegen von Teilen oder der ganzen Arbeit

oder bedürfen meiner ausdrücklichen schriftlichen Erlaubnis.

Textsatz und Vorlagen

Diese Arbeit ist mit dem Textsatzprogramm LaTeX und der Dokumentenklasse KOMA-Skipt erstellt worden. Dabei hat pdfTeX die PDF-Datei erstellt. Unter `http://www.bretschneidernet.de/tips/thesislatex.html` ist die Vorlage dieser Diplomarbeit und eine Herangehensweise für das Verfassen von längeren Arbeiten mit LaTeX hinterlegt. Die Prinzipskizzen sind mit dem Vektorgraphikprogramm Ipe und die Plots mit MATLAB erstellt worden.

Adressen

URL: `http://www.bretschneidernet.de/publications/Martin_Bretschneider__Untersuchungen_zur_Sturzerkennung_zweibeiniger_Roboter_mit_Hilfe_von_Verfahren_zur_Mustererkennung.pdf`

bibTeX-Datei: `http://www.bretschneidernet.de/publications/Martin_Bretschneider__Untersuchungen_zur_Sturzerkennung_zweibeiniger_Roboter_mit_Hilfe_von_Verfahren_zur_Mustererkennung.bib`

Vorwort

Ich habe diese Diplomarbeit im März 2010 auf meiner Website veröffentlicht, damit die Inhalte und Ergebnisse gelesen, verwendet und weiterentwickelt werden.

Sie wird bis auf unbestimmte Zeit unter der oben genannten Adresse verfügbar sein und ist somit uneingeschränkt zitierfähig. Ich freue mich, wenn auf diese Arbeit durch ein Zitat verwiesen wird und weise vorsorglich darauf hin, dass Plagiate heutzutage immer leichter nachgewiesen werden können, vor allem, weil diese Arbeit bei Google sehr gut indiziert ist. Inzwischen wird diese Arbeit bei Begriffen wie *Roboter und Musterkennung* bei Google in den ersten Suchergebnissen gelistet. Sie ist inzwischen mehrere hundertmal heruntergeladen worden.

Dr.-Ing. Martin Bretschneider im August 2011

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Allgemeine Symbole

Symbol	Bedeutung
a	der Skalar a
$\vec{x}$	der Vektor $\vec{x}$
$\mathbf{A}$	die Matrix $\mathbf{A}$
t_p	Schrittphase
*Wortende	regulärer Ausdruck für eine beliebige Anzahl von Zeichen vor dem *Wortende*.
SWF	Der Schwingfuß ist derjenige Fuß, der den Boden beim Gehen nicht berührt.
STF	Der Standfuß ist derjenige Fuße, den den Boden beim Gehen berührt.

Symbole für die Prototypenklassifikation

Symbol	Bedeutung
k	Klasse
K	Anzahl der Klassen
j	Merkmal
J	Anzahl der Merkmale
$\vec{l}_k$	Merkmalsvektor l der Klasse k
L_k	Anzahl der verschiedenen Merkmalsvektoren l für eine Klasse k
$\vec{m}_k$	Protoypenvektor der Klasse k
$\vec{y}$	Merkmalsvektor
$c(y)$	Klassifizierungsfunktion, die y einer Klasse zuordnet
$\mathrm{PK}^{\mathrm{ws}}$	Prototypenklassifikation, die aus Merkmalen angelernt worden ist, die *während des Stoßes* aufgenommen worden sind.
$\mathrm{PK}^{\mathrm{ns}}$	Prototypenklassifikation, die aus Merkmalen angelernt worden ist, die *nach dem Stoß* aufgenommen worden sind.

Symbole für die Hidden Markov Models

Symbol	Bedeutung
N	Anzahl der Zustände
T	Anzahl der Beobachtungen
λ_i	Hidden Markov Model (HMM) i
s_i	Zustand i
$\vec{S} = (s_1, s_2, \ldots, s_N)$	Zustandssequenz
o_i	Beobachtung
$\vec{O} = (o_1, o_2, \ldots, o_T)$	Beobachtungssequenz aller Beobachtungen bis zum Zeitpunkt T
a_{ij}	Übergangswahrscheinlichkeit von Zustand i zu Zustand j
$\mathbf{A}$	Übergangswahrscheinlichkeitsmatrix für alle a_{ij}
b_{io_j}	Ausgabewahrscheinlichkeit der Beobachtung o_j im Zustand s_i
$\mathbf{B}$	Ausgabewahrscheinlichkeitsmatrix für alle Beobachtungen $\vec{O}$ und Zustände $\vec{S}$
π_i	Startwahrscheinlichkeit des Zustands i
$\vec{\pi}$	Startwahrscheinlichkeitsvektor der Zustände $\vec{S}$
$p(\cdot)$	Wahrscheinlichkeit
$\tilde{p}(\cdot) = \log(p(\cdot))$	logarithmierte Wahrscheinlichkeit
${}_{\mathrm{p}}\mathrm{HMMK}$	HMM-Klassifikation Prototyp-Hidden Markov Model
${}_{\mathrm{r}}\mathrm{HMMK}$	HMM-Klassifikation Richtungs-Hidden Markov Model
$\mathrm{HMMK}_{3,4,5}$	HMM-Klassifikation, bei dem jedes HMM die Zustände $s \in [3,4,5]$ hat
$\mathrm{HMMK}^{\mathrm{ws,ns}}$	HMM-Klassifikation, bei dem die $\mathrm{PK}^{\mathrm{ws}}$ oder $\mathrm{PK}^{\mathrm{ns}}$ zur Beobachtungsgenerierung benutzt wird
${}_{\mathrm{p,r}}\mathrm{HMMK}^{\mathrm{ws,ns}}_{3,4,5}$	alle möglichen Kombinationen der HMM-Klassifikation

Symbole für die Stabilitätsaussage anhand der Fuß-Kippwinkelgeschwindigkeit

Symbol	Bedeutung
$\dot{q}_y$	aktuelle Fuß-Kippwinkelgeschwindigkeit in y-Richtung (vorne/hinten)
$\dot{q}_x$	aktuelle Fuß-Kippwinkelgeschwindigkeit in x-Richtung (seitliche Richtung)
$\dot{q}_{max}^{vor,rueck,STF,SWF}$	für die Stabilität maximal erlaubte Fuß-Kippwinkelgeschwindigkeit über die vier Kanten des Standfußes

Symbole für die Benchmarks

Symbol	Bedeutung
SoR	Simulation ohne Reaktion
SmR	Simulation mit der von dem verwendeten Mustererkennungsverfahren bestimmten Reaktion.
SmgR	Simulation mit einer gesteuerten Standardreaktion 120 ms nach dem Beginn des Stoßes.
σ	Sturzvermeidungsgröße
ρ	Robustheitsgröße
γ	Roboter ist am Ende der Simulation gefallen
$\bar{\gamma}$	Roboter ist am Ende der Simulation *nicht* gefallen
ϵ	Erfolgsquote
τ	Reaktionszeit

1 Einleitung

Roboter werden heutzutage zum großen Teil als Industrieroboter in der Fertigung verwendet. In diesem Umfeld haben sie einen festgelegten Arbeitsraum und sind meist fest auf dem Untergrund fixiert. Serviceroboter sollen dem Menschen Dienstleistungen bereitstellen, die ihm das Leben erleichtern oder die er nicht selber ausführen will. Sie können beispielsweise Aufgaben im Haushalt übernehmen, für Unterhaltung sorgen oder auch Bomben entschärfen.

Damit Serviceroboter die Aufgaben im Umfeld von Menschen erledigen können, müssen sie sich dort auch bewegen können. Es gibt Ansätze, Serviceroboter mit Rädern oder Beinen zur Fortbewegung auszustatten. Radgebundene Systeme haben den Vorteil einer hohen Stabilität und einer weniger komplexen Struktur. Damit radgebundene Roboter Treppen steigen können, müssen sie jedoch besondere Eigenschaften wie eine flexible Struktur oder übergroße Räder haben. Roboter, die sich auf Beinen fortbewegen, können dem Menschen nachgeahmt werden. Sogenannte bipedale Roboter haben – wie der Mensch – zwei Beine und können zum Beispiel Treppen steigen. Da diese Roboter nur mit zwei Füßen Kontakt zum Boden haben und während des Gehens die meiste Zeit nur ein Fuß den Boden berührt, sind sie relativ instabil. Wenn sie stürzen, können sie sich selber und ihre Umgebung beschädigen und schlimmstenfalls Menschen verletzen. Daher sollte der Roboter in der Lage sein, einen drohenden Sturz zu erkennen und darauf entsprechend zu reagieren.

Es hat sich gezeigt, dass bei Stürzen oder instabilen Haltungen charakteristische Muster in den Sensordaten des Roboters auftreten. In dieser Arbeit werden verschiedene Mustererkennungsverfahren daraufhin untersucht, inwiefern es mit ihnen möglich ist, die charakteristischen Signale zu klassifizieren und so die drohenden Stürze durch eine entsprechende Reaktion des Roboters zu vermeiden.

In Kapitel 2 werden der bipedale Roboter BARt-UH und ein simuliertes Robotermodell vorgestellt. Weiterhin werden Störungen und die daraus folgenden Auswirkungen mit Hilfe des simulierten Roboters analysiert. In Kapitel 3 wird die Prototypenklassifikation als ein mögliches Mustererkennungsverfahren zur Vorhersage von Stürzen untersucht. Ein weiteres Verfahren auf Basis von stochastischen Modellen wird mit den Hidden Markov Models in Kapitel 4 beschrieben. Kapitel 5 führt in einen Algorithmus ein, der anhand der Fuß-Kippbewegung Aussagen zur Stabilität des Roboters liefert. Diese drei Verfahren werden in Kapitel 6 anhand verschiedener Sturzszenarien des simulierten Roboters miteinander verglichen und in Kapitel 7 auf BARt-UH übertragen. Die Zusammenfassung und ein Ausblick finden sich in Kapitel 8.

2 Sturzvermeidung durch Reflexreaktionen

In diesem Kapitel werden der Roboter BARt-UH[1] und das simulierte Robotermodell beschrieben. Die Störungsarten, die den Roboter zu Fall bringen können, werden dargestellt, ebenso wie die Reaktionen, die der Roboter ausführen kann, um einen Sturz zu vermeiden. Eine Stabilitätsanalyse zeigt auf, dass die Eigenstabilität des Roboters von vielen Parametern abhängt.

2.1 Der zweibeinige Roboter BARt-UH

Die Verifikation der in dieser Arbeit untersuchten Verfahren erfolgte im Wesentlichen mit Hilfe des zweibeinigen Roboters BARt-UH, der 1999 am Institut für Regelungstechnik der Universität Hannover entwickelt worden ist [HS99]. Dieser verfügt in jedem Bein ein Hüft-, Knie- und Fußgelenk und somit insgesamt sechs Freiheitsgrade. Hiermit ist BARt-UH das Gehen in der sagitalen Ebene, also Vorwärtsgang und Rückwärtsgang, möglich. Auf der Plattform (Torso), an der sich die beiden Hüftgelenke befinden, sind die Leistungselektronik für die Gelenkmotoren, der Mikrocontroller und die Akkus für die Stromversorgung untergebracht. BARt-UH hat eine Masse von etwa 25 kg und eine Höhe von etwa 80 cm. Mit einem Akkusatz ist der Roboter vollständig autonom für eine Betriebsdauer von etwa 30 Minuten.

Auf dem Mikrocontroller vom Typ MPC555 läuft das Echtzeitbetriebssystem RTOS-UH[2]. Über einen CAN[3]-Bus können Sensordaten von auf BARt befindlichen Sensoren auf einen

1 **B**ipedal **A**utonomous **R**obo**t** - **U**niversität **H**annover.
2 **R**eal-**T**ime **O**perating **S**ystem - **U**niversität **H**annover.
3 **C**ontroller **A**rea **N**etwork.

PC übertragen werden. In dieser Arbeit wurden die Daten von Fußkraft-Sensoren, Beschleunigungssensoren auf dem Torso und zwei IMU[4] ausgewertet. Eine große IMU [See03] befindet sich auf dem Torso für die Gewinnung von Sensordaten wie Winkel und Winkelgeschwindigkeiten des Torsos und eine kleinere μIMU auf einem Fuß für die des Fußes. Die IMU können verschiedene Sensordaten wie die Position, Lage (also Winkel) und Winkelgeschwindigkeiten im Raum bestimmen.

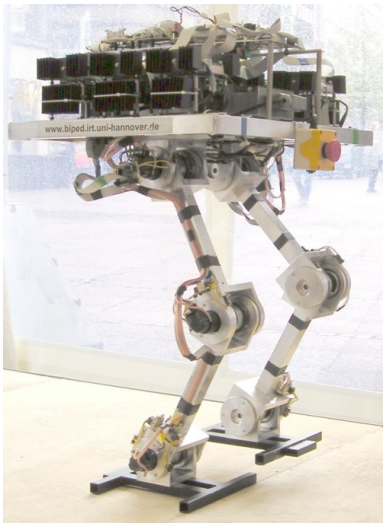


Abbildung 2.1: Der zweibeinige Roboter BARt-UH

Für die in dieser Arbeit untersuchten Sturzszenarien ist die Plattform auf dem Torso mit der Leistungselektronik, den Akkus und dem Mikrocontroller entfernt worden, um mögliche Beschädigungen zu vermeiden.

2.2 Simulation des zweibeinigen Roboters

Damit BARt-UH bei den Sturzuntersuchungen nicht unnötig gefährdet wird, ist ein Simulator entwickelt worden. Dieser gewährleistet neben der Schonung des Roboters auch eine Zeitersparnis. Der simulierte Roboter hat im Vergleich zu BARt-UH zwei zusätzliche Freiheitsgrade in jedem Fuß- und Hüftgelenk, so dass er flexiblere Bewegungen wie Schritte zur Seite ausführen kann.

4 Die **I**nertial **M**essure **U**nit kann als Gleichgewichtsorgan bezeichnet werden.

Der Simulator ist mit der Entwicklungsumgebung Delphi der Firma Borland für das Betriebssystem Microsoft Windows entwickelt worden. Die Visualisierung ist über den 3D-Standard OpenGL realisiert. Die graphische Oberfläche des Simulation-Tools (siehe Abbildung 2.2) ermöglicht das Einstellen von Bodenbeschaffenheiten wie Neigung und Bodenunebenheiten. Es können Störungsarten ausgewählt und sofort danach in der Simulation ausführt werden. Um den Verlauf eines Sturzes genauer betrachten zu können, ist es mit einem Schieberegler möglich, jeden berechneten Zeitpunkt bis auf eine halbe Millisekunde aus allen Ansichten genau zu betrachten. Außerdem können die in der Simulation berechneten Sensordaten in einem Graph dargestellt werden. Diese Sensordaten können danach in eine Datei zur weiteren Verarbeitung mit dem Mathematikprogramm MATLAB exportiert werden. In verschiedenen nachladbaren Bibliotheken (DLL) können die Fuß-Trajektorien, Regler der Gelenke und andere Funktionen geladen werden.

Das kinetische Modell des Roboters hat die Gleichung

$$\mathbf{M}(\vec{q})\ddot{\vec{q}} + \vec{c}(\vec{q},\dot{\vec{q}}) + \vec{g}(\vec{q}) + \vec{Q}_{Reibung}(\vec{q},\dot{\vec{q}}) = \vec{Q}_{Motor} + \vec{Q}_{extern} \tag{2.1}$$

mit der Massenmatrix $\mathbf{M}(q)$, den Coriolis- und Eulerkräften in dem Vektor $\vec{c}(\vec{q},\dot{\vec{q}})$, den Gravitationskräften $\vec{g}(q)$, den nichtkonservativen Reibkräften $\vec{Q}_{Reibung}(\vec{q},\dot{\vec{q}})$, der durch die Motoren erzeugten Kräfte $\vec{Q}_{Motor}$ und den von außen zugeführten Kräften $\vec{Q}_{extern}$ [HGG05]. Im Vektor $\vec{q}$ sind die Gelenkwinkel sowie die Positions- und Winkelkoordinaten des Roboters enthalten. Diese Gleichung (bzw. das Gleichungssystem) wird nach $\ddot{\vec{q}}$ umgestellt, wobei für das Invertieren der Massenmatrix $\mathbf{M}$ die Cholesky-Zerlegung benutzt wird. Dann wird $\ddot{\vec{q}}$ mit Hilfe des Runge-Kutta-Verfahrens zweimal numerisch differenziert, so dass sich $\vec{q}$ ergibt. Die Trajektorien der Füße beim normalen Laufen und bei anderen Schrittbewegungen als Reaktion zur Sturzvermeidung werden durch Splines realisiert. Die Regelung der Gelenkwinkel erfolgt durch einen PD-Regler.

Um dem simulierten Roboter reproduzierbare Störungen zuzufügen, ist eine Skriptsprache entwickelt worden, in der Parameter wie Stoßzeitpunkt, Stoßrichtung, Stoßstärke, Reaktion, Reaktionszeit, Aufnahmezeitpunkte der Merkmalsvektoren (siehe Abschnitt 3.3), das Einbinden von DLLs für Trajektorien Mustererkennungsverfahren sowie die Dateipfade für das Speichern der beim Stoßen aufgezeichneten Pfade festgelegt werden können.

Ein Problem des Simulators ist, dass er bei jeder Simulation Speicher alloziert, diesen nach

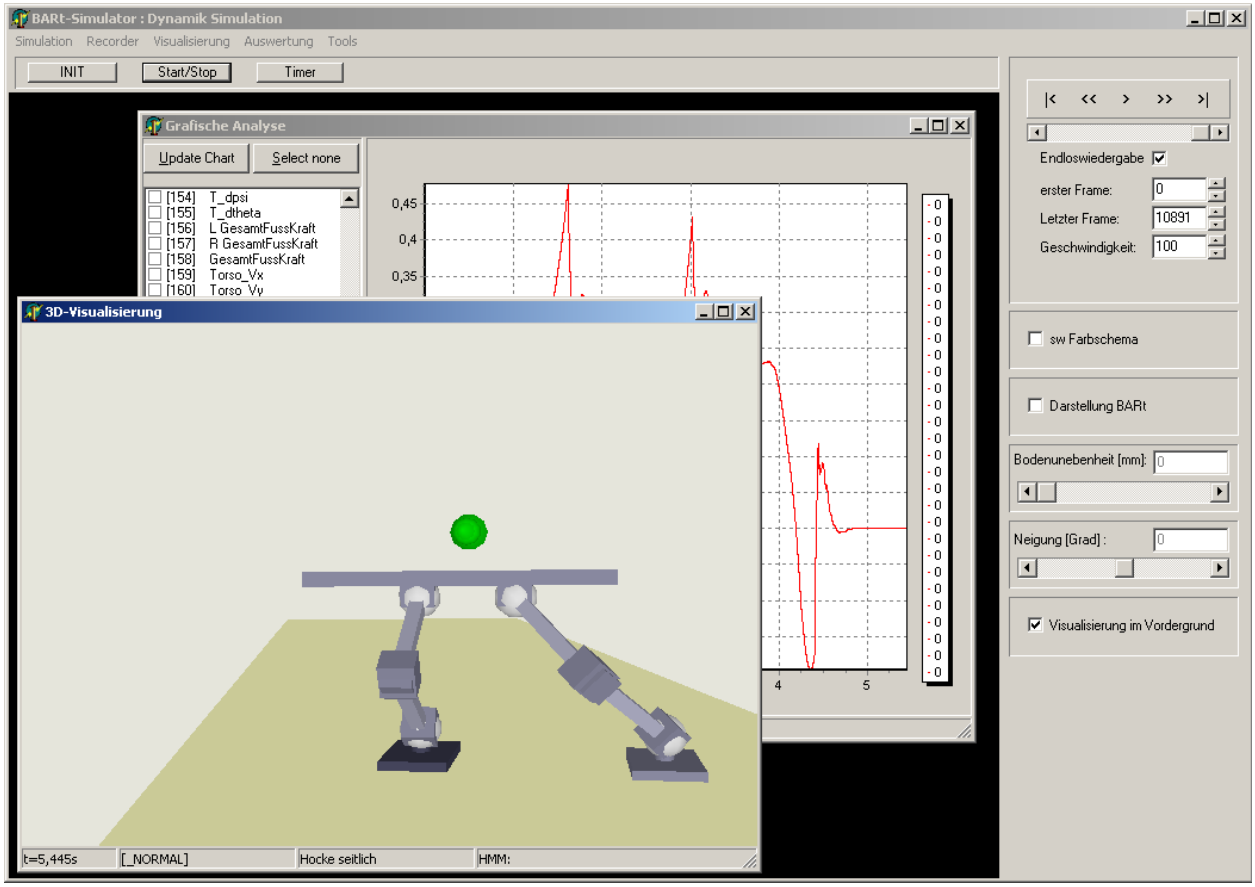


Abbildung 2.2: Graphische Oberfläche des BARt-UH-Simulators: Im Vordergrund ist die Visualisierung des Roboters zu sehen, in der der simulierte Roboter die seitliche Hocke ausgeführt hat und nun steht.

der Simulation aber nicht wieder frei gibt und nach einigen Simulationen abstürzt. Deshalb ist in dieser Arbeit für die Skriptsprache ein Perl[5]-Skript entwickelt worden, welches es ermöglicht, die Simulationsskripte zu erzeugen. Hiermit können die Parameter für mehrere Simulationen festgelegt werden und der Simulator wird von dem Perl-Skript gesteuert. Dieses lässt den Simulator nur wenige Simulationen durchführen, der Simulator beendet sich anschließend, das Perl-Skript wertet den Rückgabewert des Simulators aus und startet darauf die Simulation gegebenenfalls erneut.

5 Perl ist in einer - wenn auch älteren Version von 1999 - bei der MATLAB-Installation unter Windows im Pfad `Installationspfad\sys\perl\win32\bin\perl\` installiert.

2.3 Störarten

In der Simulation können dem Roboter verschiedene Störungen zugefügt werden. Einerseits kann der Roboter Stöße gegen den Torso erhalten, anderseits kann die Ebene, auf der der Roboter läuft, modifiziert werden. So kann eine Stufe eingebaut oder der Boden entfernt werden. Der Untergrund kann auch Unebenheiten erhalten, was unwegsamem Gelände entspricht. Oder mit einem niedrigen Reibwert μ kann glatter Untergrund simuliert werden.

In dieser Arbeit liegt der Schwerpunkt auf der Untersuchung von Stößen. Diese können dem Roboter aus vier Richtungen zugefügt werden: Von vorne, von hinten und von den beiden Seiten. Diese seitlichen Stöße werden nach der Beinkonfiguration benannt: Wird der Roboter an der Seite gestoßen, an der sich der Schwingfuß befindet, heißt der Stoß *Stoß über den Standfuß*. Derjenige Stoß, der an der Seite des Standfußes angewandt wird, wird als *Stoß über den Schwingfuß* bezeichnet (STF[6]).

Die Intensität eines Stoßes p ergibt sich aus der Kraft $\vec{f}$, die eine Zeitdauer t einwirkt:

$$\vec{p} = \vec{f} \cdot t \qquad [\mathrm{Ns}]. \tag{2.2}$$

2.4 Reaktionsarten

Auf die in Abschnitt 2.3 genannten vier Stoßarten kann der Roboter mit verschiedenen Reaktionen antworten, um ein Fallen zu vermeiden. Bei einem Stoß von vorne kann der Roboter zum Beispiel mit dem Schwingfuß einen Schritt nach hinten durchführen. Wird der Stoß von hinten appliziert, sollte der Roboter mit einem Schritt nach vorne antworten. Der Stoß über den Schwingfuß sollte eine seitliche Hocke des Roboters nach sich ziehen. Bei dem Stoß über den Standfuß kann der Roboter mit einem seitlichen Schritt reagieren. Diese Reaktionen werden *Standardreaktion* genannt und sind nocheinmal in Tabelle 2.1 aufgeführt.

Außerdem kann der Roboter auch eine Hocke ausführen, die er einnehmen soll, wenn der Sturz nicht mehr vermieden werden kann. Durch diese Hocke soll die Rotationsgeschwindigkeit verringert werden, mit der den Roboter auf den Boden aufkommt. Diese

6 Im Folgenden werden die Abkürzungen *SWF* und *STF* vor allem in Abbildungen verwandt.

Stoßart	Standardreaktion
Stoß von hinten	Schritt nach vorne
Stoß von vorn	Schritt nach hinten
Stoß über den Standfuß	Hocke zur Seite
Stoß über Schwingfuß	Schritt zur Seite

Tabelle 2.1: Standardreaktionen zu den Stoßrichtungen

Reaktion wird im Folgenden *Schutzreaktion* genannt und ist unabhängig von der Stoßrichtung.

Stoßart	Schutzreaktion
starke Stöße und unvermeindbarer Sturz	Hocke

Tabelle 2.2: Schutzreaktion zu den Stoßrichtungen

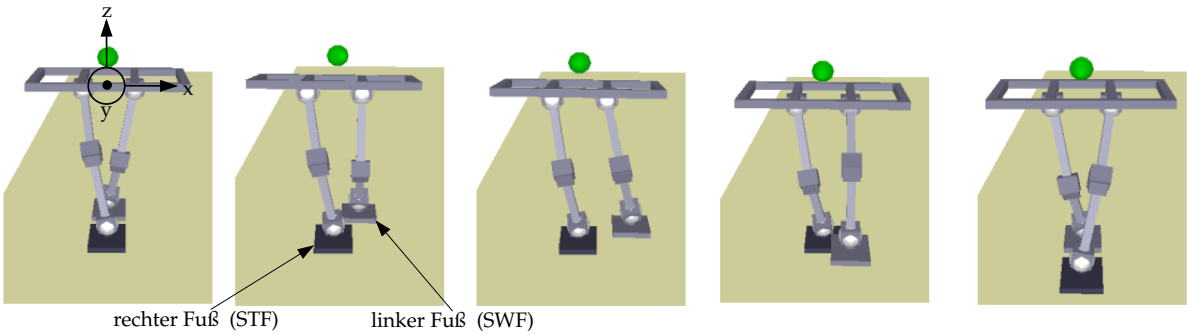
2.5 Stabilitätsanalyse

Der Begriff *Stabilitätsanalyse* beschreibt die Auswertung der Stabilität des Roboters, nachdem er zu unterschiedlichen Zeitpunkten, mit unterschiedlicher Stärke und aus unterschiedlichen Richtungen gestört wurde. Dabei wird für jede Stoßrichtung eine Abbildung erstellt, in der für jede Kombination von Stoßimpuls und Stoßzeitpunkt dargestellt wird, ob der Roboter gefallen ist oder nicht (siehe Abbildung 2.4). In Kapitel 6 dient die Stabilitätsanalyse zum Vergleichen der Mustererkennungsverfahren.

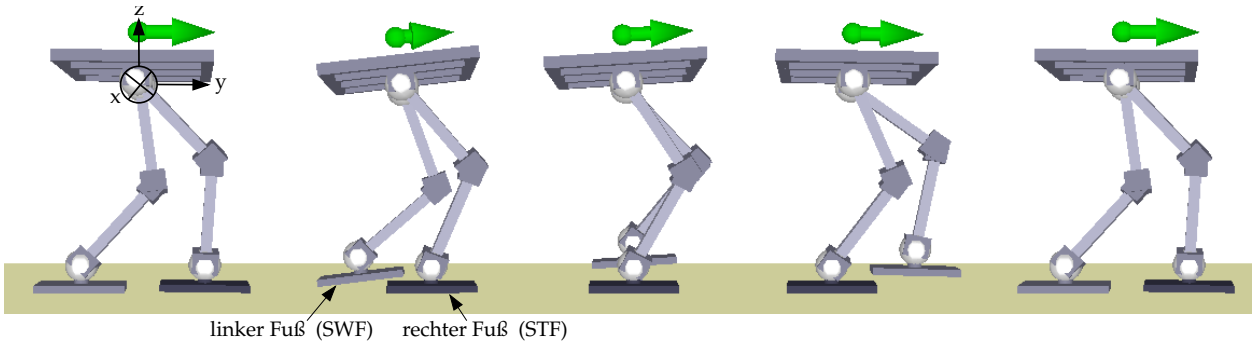
Der Stoßzeitpunkt kann absolut in Sekunden oder relativ innerhalb einer dimensionslosen Schrittphase t_p angegeben werden. Bei der absoluten Zeitpunktangabe in Sekunden wird die Periodizität der Laufbewegung des Roboters nicht betrachtet[7], bei der Angabe des Zeitpunktes in der Schrittphase hingegen schon. Mit der Schrittphase kann ein Zeitpunkt relativ angegeben werden, was die Konfiguration unabhängig von dem bisherigen Verlauf eindeutig erklärt. Die Schrittphase t_p läuft während eines Schrittes von -1 bis $+1$ (siehe

7 Es ist unerheblich bei dieser Fragestellung, mit welchem seitlichen Stoß der Roboter gestört wird. Aufgrund der seitlichen Symmetrie des Roboters verhält er sich bei gleicher Beinkonfiguration – also gleicher Schrittphase t_p – gleich: Wird er auf dem rechten Bein stehend von rechts angestoßen, verhält er sich genauso, wie wenn er von links auf dem linken Bein stehend angestoßen wird.

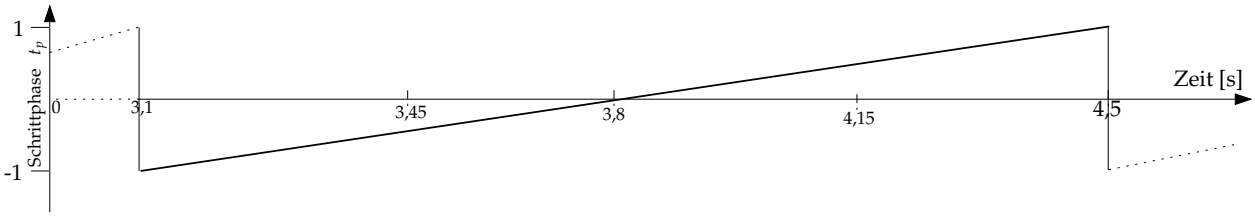
Abbildung 2.3). Berühren beide Füße den Boden ist $t_p = -1$. Der Schwingfuß hebt sich und bei $t_p = 0$ befindet er sich neben dem Standfuß. Bei $t_p = +1$ setzt der Schwingfuß vor dem Standfuß auf, so dass wieder beide Füße den Boden berühren. Der Schritt ist durchgeführt, beide Füße wechseln die Aufgabe und der Standfuß wird zum Schwingfuß, der Schwingfuß zum Standfuß.



(a) Ansicht von vorne (der Pfeil über dem Roboter zeigt die Geschwindigkeit des Torsos an)



(b) Ansicht von der rechten Seite mit Koordinatensystem



(c) Ein Schritt mit Angabe der Zeit in Sekunden sowie der Schrittphase t_p

Abbildung 2.3: Ein Schritt des Roboters beim normalen menschenähnlichen Vorwärtsgang

Nachdem der Roboter – zum Beispiel mit Stößen – gestört worden ist, sind für seine Stabilität vor allem drei Faktoren entscheidend:

die Position des Schwingfußes: Je nach Position des Schwingfußes kann der Roboter

Stöße auch ohne explizite Reaktion abfangen.

die Position des Schwerpunktes: Je weiter der auf die Laufebene projizierte Schwerpunkt von der Mitte des Roboters entfernt ist, desto deutlicher ändert sich die Stabilität in bestimmte Fallrichtungen.

die kinetische Energie: Der Roboter hat durch die Vorwärtsbewegung kinetische Energie inne. Durch einen Stoß wird dem Roboter weitere Energie zugefügt. Aus der Überlagerung ergibt sich je nach Stoßrichtung und Stoßstärke eventuell eine Fallbewegung.

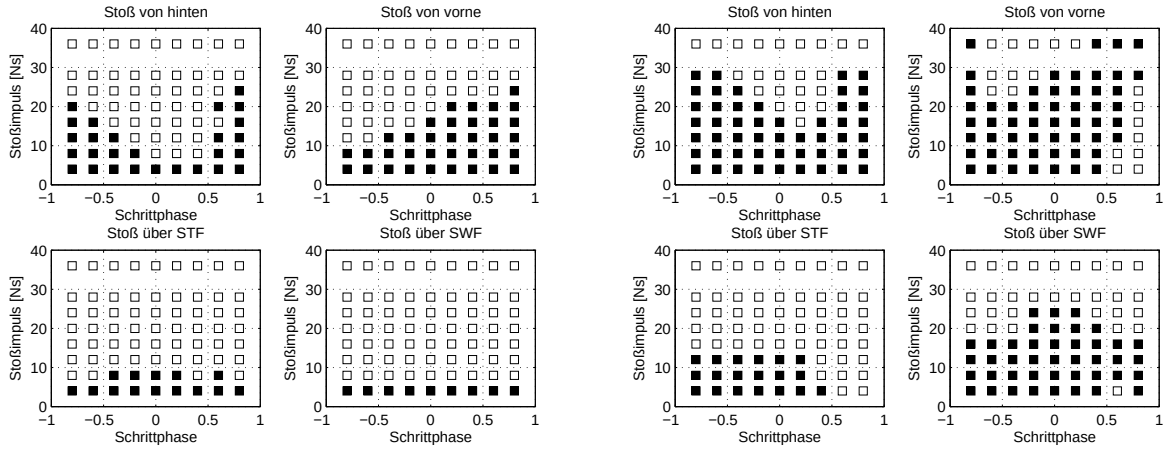
In Abbildung 2.4 ist eine Stabilitätsanalyse bei Stößen ohne Reaktion des Roboters und mit gesteuerter Standardreaktion 120 ms nach Beginn des Stoßes gegenübergestellt. Diese gesteuerte Reaktion zeigt das optimale Verhalten nach einem Stoß mit einer angenommen Reaktionszeit von 120 ms. Die Stoßdauer beträgt jeweils 60 ms. An diesen beiden Abbildungen lässt sich einerseits die Eigenstabilität des Roboters an der Stabilitätsanalyse ohne Reaktion und anderseits das bestmögliche Ergebnis bei der Stabilitätsanalyse mit gesteuerter Reaktion erkennen: Somit ist die Eigenstabilität bei Stößen von vorne und hinten stärker als bei den seitlichen Stößen.

Bei einem Stoß von vorne ist die Eigenstabilität besonders hoch, da die kinetische Energie, die der Roboter durch das Laufen hat, groß im Vergleich zu der Stoßenergie ist. Die Eigenstabilität nimmt bei diesem Stoß von vorne mit steigender Schrittphase zu. Diese Tatsache ist für die Schrittphasen ab $t_p = 0$ erklärbar, weil der Roboter beim Bewegen des Schwingfußes nach vorne seine Geschwindigkeit erhöht (vergleiche auch die Geschwindigkeitspfeile in Abbildung 2.3 und Abbildung A.1(d) auf Seite 97) und der Schwerpunkt auch nach vorne wandert.

Bei einem Stoß von hinten ist die Eigenstabilität in der Schrittphase $0 \leq t_p \leq 0{,}4$ besonders gering. Zu diesen Zeitpunkten hat der Schwingfuß den größten Abstand zum Boden. Bei Stößen zu späteren Zeitpunkten wird der Schwingfuß durch den Stoß schneller auf den Boden bewegt, so dass der Roboter auf diese Weise stabilisiert wird. In der Schrittphase $-1 \leq t_p \leq 0{,}2$ hat der Roboter eine geringe Geschwindigkeit und kann den Impuls noch mit seiner durch die von der Geschwindigkeit gegebene kinetische Energie aufnehmen; allerdings vergrößert sich der Abstand des Schwingfußes auch, so dass dieser Einfluss Überhand gewinnt.

Seitliche Stöße kann der Roboter jedoch nur mit kleinem Stoßimpuls ertragen, ohne zu fallen. Denn hier kann keine vorteilige Position des Schwingfußes den Stoß abfangen. Bei dem

Stoß über den Standfuß befindet sich der Schwerpunkt in der mittleren Schrittphase $-0{,}4 \leq t_p \leq 0{,}2$ in Richtung des Schwingfußes und ist so etwas günstiger, weshalb der Roboter in dieser Konstellation bei Impulsen von 8 Ns noch nicht fällt.



(a) ohne Reaktion (b) mit gesteuerter Reaktion

Abbildung 2.4: Stabilitätsanalyse bei verschiedenen Stoßstärken aus verschiedenen Richtungen mit und ohne Reaktion. Kasten mit schwarzem Inhalt heißt *Roboter ist nicht gefallen*, Kasten mit weißer Fläche heißt *Roboter ist gefallen*

Bei den gesteuerten Reaktionen ist der Erfolg – wie zu erwarten – in der Regel höher als bei der Simulation ohne Reaktion. Es fällt besonders fällt auf, dass der Roboter bei dem Stoß über den Schwingfuß seine Erfolge verbessern konnte. Hier ist der Ausfallschritt des Schwingfußes besonders in den Schrittphasen $-0{,}4 \leq t_p \leq 0{,}6$ erfolgreich, wenn der Schwingfuß sich in der Mitte unter dem Torso befindet. Bei sehr frühen und späten Schrittphasen und starken Stößen ist dieser Ausfallschritt allerdings nicht mehr erfolgreich, weil der Schwingfuß durch den Ausfallschritt versetzt neben den Standfuß gesetzt wird und der Roboter dann über die Fußkanten fällt.

Bei dem anderen seitlichen Stoß, dem Stoß über den Standfuß, ist zwar auch eine Verbesserung erkennbar, allerdings fällt diese nicht sehr groß aus. Der Grund ist darin zu sehen, dass der Ausfallschritt des Schwingfußes bei stärkeren Stoßimpulsen den Roboter nicht stabilisieren kann.

Beim Vergleich zu der Analyse ohne Reaktion fällt auf, dass der Roboter bei Störungen zu späten Schrittphasen $t_p \geq 0{,}4$ beim Stoß von vorne sogar häufiger gefallen ist. Hier hat der

gesteuerte Ausfallschritt zu dem ungünstigen Zeitpunkt den Fall erst ausgelöst. Denn in diesen späteren Schrittphasen hat sich der Schwingfuß vor den Torso bewegt und ist kurz vor dem Aufsetzten auf den Boden. Nun führt der Roboter die Standardreaktion – den Schritt nach hinten – mit dem Schwingfuß aus und der Roboter kippt über den Standfuß nach vorne. Hätte der Roboter keine Reaktion ausgeführt, hätte sich der Schwingfuß auf den Boden bewegt und hätte so den Stoß abgefangen. Hieran ist ersichtlich, dass die Reaktionen den Roboter nicht immer vor dem Sturz bewahren.

Die Stabilität nach dem Stoß von hinten hat sich bei allen Stoßimpulsen mit der gesteuerten Standardreaktion verbessert . Hier ist – wie bei der Analyse ohne Reaktion – der Misserfolg bei den Schrittphasen $0 \leq t_p \leq 0{,}4$ sichtbar. Bei den Stößen von vorne ist auch eine beträchtliche Steigerung erkennbar: Fast jeder Stoß von vorne kann durch den Schritt nach hinten abgefangen werden. Interessant sind hier wieder die Verschlechterungen in der späten Schrittphase bei den geringen Stößen: Hier hat erneut der gesteuerte Schritt das Fallen ausgelöst.

Es ist also notwendig, die Störung genau zu klassifizieren, um die sinnvollste zur Verfügung stehende Reaktion auszuführen. Bei manchen Stößen ist eine Reaktion gar nicht nötig, weil der Roboter eine ausreichende Eigenstabilität besitzt. Sie kann im Gegenteil erst dazu führen, dass der Roboter zu Fall kommt.

3 Prototypenklassifikation

3.1 Grundlagen der Mustererkennung

Die Mustererkennung ist ein Oberbegriff für Verfahren, die Mustern eine greifbare Bedeutung zuweisen. Dabei können die Muster beliebige Signale sein, die von einem System erzeugt werden. Anwendungsgebiete der Mustererkennung sind beispielsweise die Spracherkennung bei der Fahrplanauskunft oder Handschrifterkennung als Anwendung der Bildverarbeitung. Im Folgenden werden die Schritte der Mustererkennungsverfahren an Hand der Spracherkennung aufgezeigt:

1. Die von einem Menschen gesprochene Sprache (die Muster) wird von einem Mikrophon und einem AD-Wandler in verarbeitbare Daten umgewandelt.
2. In der Vorverarbeitung werden Hintergrundgeräusche herausgefiltert.
3. In der Merkmalsgewinnung werden aus den Mustern mit vielen Informationen *Merkmale* mit weniger aber prägnanteren Informationen erzeugt. Hier können Zeit-Frequenz-Transformationen durchgeführt werden oder die Sprache in die Grundbestandteile, die *Phoneme* zerschnitten werden.
4. Die Merkmalsauswahl filtert unnötige Merkmale heraus, so dass nur relevante Frequenzbänder oder Zeitabschnitte weiterverwendet werden. Diese Merkmale werden in einem Merkmalsvektor $\vec{y}$ zusammengefasst.
5. Die Klassifikation der Phoneme wird durchgeführt und die Phoneme werden zu Wörtern unter Zuhilfenahme eines Wörterbuches mit bekannten Wörtern zusammengesetzt.

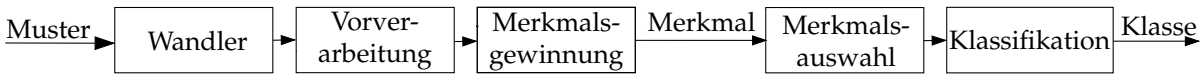


Abbildung 3.1: Übersicht über die Schritte der Mustererkennung von den Mustern bis zur erkannten Klasse

In der Reihenfolge dieser Schritte werden die Informationen reduziert, aber die Bedeutung dieser Informationen gesteigert.

Der Klassifizierungschritt ist der eigentliche Kern der Mustererkennung. Die Klassifizierungsvorschrift $c(\vec{y})$ ordnet ein Merkmalsvektor $\vec{y}$ einer Klasse k zu.

Es gibt verschiedene Klassifizierungsverfahren, die sich in der Regel in zwei Kategorien einteilen lassen:

1. In der **numerischen Mustererkennung** wird einem unbekannten Merkmalsvektor eine Klasse unter Zuhilfenahme einer Diskriminazfunktion oder Unterscheidungsfunktion d_k zugewiesen. Diese Funktion muss die Eigenschaft haben, die Zugehörigkeit eines Merkmalsvektors zu einer Klasse auszudrücken.

2. Die **syntaktische Mustererkennung** basiert auf Beziehungen von Merkmalen untereinander. Dabei werden Zeichenketten in der Form einer Sprache mit semantischen und syntaktischen Eigenschaften oder einer mit einer Baumstruktur ausgewertet. Dieser syntaktischen Mustererkennung kann die numerische Mustererkennung vorgeschaltet sein. Diese Kombination wird dementsprechend numerische-syntaktische Mustererkennung genannt.

Die Mustererkennung lässt sich in der Durchführung in zwei Schritte einteilen (siehe auch Abbildung 3.2):

1. In einem Anlernen-Schritt werden viele Trainings-Muster so verarbeitet, dass die gewonnenen Merkmale die Zuordnung zu Klassen ermöglichen. Diese Klasseninformationen werden in bestimmten Datentypen gespeichert. Dieser Schritt kann je nach Aufgabe offline geschehen.

2. In dem Klassifizierungs-Schritt wird das aktuelle Problem-Muster unter Zuhilfenahme der im Schritt 1 gewonnenen Informationen mit der Klassifizierungsvorschrift einer bestimmten Klasse zugeordnet. Dieser Schritt muss online durchgeführt werden.

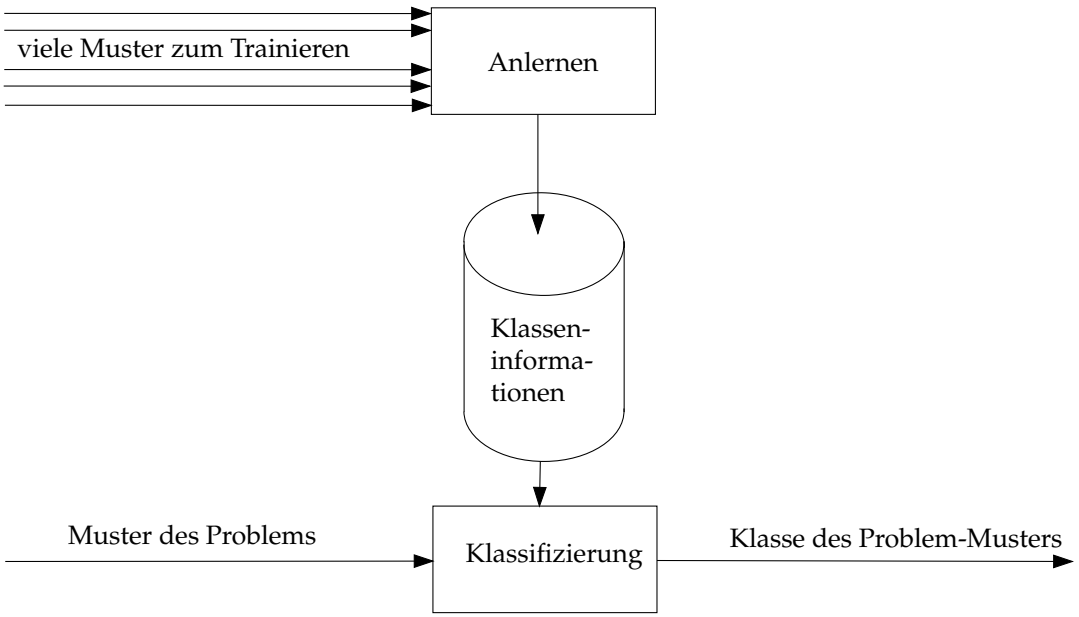


Abbildung 3.2: Anlern- und Klassifizerungsschritt in der Mustererkennung

3.2 Grundlagen der Prototypenklassifikation

Die Prototypenklassifikation [LE89, S. 97] gehört zur Kategorie der numerischen Mustererkennungsverfahren. Es wird aus J verschiedenen Merkmalen, die für die Beschreibung eines zu klassifizierenden Systems zur Verfügung stehen, ein Merkmalsvektor $\vec{y} = (y_1 \dots y_J)^T \in \Re^J$ zusammengestellt. Für jede Klasse $k \in \{k_1 \dots k_K\}$ wird ein repräsentativer Prototyp $\vec{m}_k = (m_{k1} \dots m_{kJ})^T \in \Re^J$ erstellt, der die Klasse k am besten und möglichst alle anderen Klassen schlecht beschreibt. Der Prototyp für diese Klasse kann durch Mittelwertbildung aus den L_k für die Klasse k gehörenden Merkmalsvektoren erstellt werden. In Gleichung 3.1 wird der Mittelwert $\bar{y}_{kj}$ für die gegebene Klasse k und das gegebene Merkmal j berechnet, indem alle L_k Merkmale elementweise addiert und danach durch die Anzahl der L_k Merkmale, die zur Klasse k gehören, dividiert werden:

$$\vec{m}_k = \begin{pmatrix} m_{k1} \\ m_{k2} \\ \vdots \\ m_{kJ} \end{pmatrix} = \begin{pmatrix} \bar{y}_{k1} \\ \bar{y}_{k2} \\ \vdots \\ \bar{y}_{kJ} \end{pmatrix} \qquad \text{mit} \qquad \bar{y}_{kj} = \frac{1}{L_k} \sum_{l=1}^{L_k} y_{kjl}. \tag{3.1}$$

Da die Merkmale jedoch unterschiedliche physikalische Eigenschaften und Dimensionen besitzen können, bekommt ein Merkmal mit hohen Zahlenwerten ein viel stärkeres Gewicht bei der Klassifikation als ein Merkmal mit geringen Werten. Deshalb ist die Skalierung jedes Merkmals j mit seiner Standardabweichung σ_j sinnvoll. Es wird ein weiterer Mittelwert $\bar{y}_j$ bestimmt, der nun aber alle K Klassen umfasst. Mit Hilfe dieses Mittelwertes $\bar{y}_j$ kann die quadrierte Standardabweichung σ_j^2 für jedes Merkmal j berechnet werden:

$$\sigma_j^2 = \frac{1}{L_1 + \cdots + L_k} \sum_{k=1}^{K} \sum_{l=1}^{L_k} (y_{kjl} - \bar{y}_j)^2 \qquad \text{mit} \qquad \bar{y}_j = \frac{1}{L_1 + \cdots L_k} \sum_{k=1}^{K} \sum_{l=1}^{L_k} y_{kjl}. \tag{3.2}$$

Die in den Merkmalsvektoren zusammengefassten Merkmale werden beim Anlernen der Prototypen und Klassifizieren durch ihre Standardabweichung dividiert und es ergeben sich die skalierten Merkmalsvektoren $\tilde{m}_k$:

$$\vec{\tilde{m}}_k = \begin{pmatrix} \tilde{m}_{k1} \\ \tilde{m}_{k2} \\ \vdots \\ \tilde{m}_{kJ} \end{pmatrix} = \begin{pmatrix} \tilde{\bar{y}}_{k1} \\ \tilde{\bar{y}}_{k2} \\ \vdots \\ \tilde{\bar{y}}_{kJ} \end{pmatrix} \qquad \text{mit} \qquad \tilde{\bar{y}}_{kj} = \frac{1}{L_k} \sum_{l=1}^{L_k} \frac{y_{kjl}}{\sigma_j}. \tag{3.3}$$

Während der Klassifizierung wird der Merkmalsvektor $\vec{y}$, den das zu klassifizierende System produziert, mit den Prototypen verglichen. Die Klasse desjenigen Prototypen, der den geringsten Abstand zu dem Merkmalsvektor hat, wird ausgewählt. Ein gebräuchliches Abstandsmaß ist der Minkowski-Abstand d_g, wobei g ein Parameter der natürlichen Zahlen ist, der die Bewertung der Abstände gewichtet: Ist d groß, werden größere Unterschiede zwischen den Merkmalen stärker bestraft und kleine Abstände belohnt.

$$d_g(\vec{m}_k, \vec{y}) = \left| \sum_{j=1}^{J} (|m_{kj} - y_j|)^g \right|^{\frac{1}{g}} \tag{3.4}$$

Für $d = 2$ ergibt sich der Euklidische Abstand

$$d_2(\vec{m}_k, \vec{y}) = \left| \sum_{j=1}^{J} (|m_{kj} - y_j])^2 \right|^{\frac{1}{2}} = \sqrt{\sum_{j=1}^{J} (|m_{kj} - y_j|)^2}. \tag{3.5}$$

Für die skalierten Prototypen $\tilde{\vec{m}}_k$ muss auch hier die Skalierung angewandt werden:

$$d_2(\tilde{\vec{m}}_k, \tilde{\vec{y}}) = \sqrt{\sum_{j=1}^{J} (|\tilde{m}_{kj} - \tilde{y}_j|)^2}. \tag{3.6}$$

Die Klassifizierungsvorschrift $c(\vec{y})$ lautet dann:

$$c(\vec{y}) = \arg\min_k d(\tilde{\vec{m}}_k, \tilde{\vec{y}}). \tag{3.7}$$

3.3 Der Merkmalsvektor

Der Merkmalsvektor ist die Grundlage der Prototypenklassifikation. Die darin zusammengefassten Merkmale sollen bei der in dieser Arbeit untersuchten Fragestellung einen Zustand, in dem der Roboter nicht mehr stabil ist, möglichst schnell und genau anzeigen. Diese Merkmale werden im Folgenden beschrieben; in Abschnitt A.1 sind die zeitlichen Verläufe der Merkmale während eines Stoßes von hinten gegen den Roboter abgebildet.

Center of Pressure: Der *Center of Pressure* (CoP) bezeichnet denjenigen Punkt unterhalb des Fußes, an dem die resultierende äußere Reaktionskraft $\vec{F}_R$ wirkt [Alb02, S. 50 ff.]. Dieser Punkt muss innerhalb eines Stabilitätsgebietes liegen. Diese Stabilitätsgebiet liegt auf der Ebene, auf der sich der Roboter befindet, und wird durch die Fußecken aufgespannt. Bei einem normalen Schritt bewegt sich der CoP in y-Richtung (CoP_y) nach vorne, da sich der Roboter über dem Standfuß nach vorne bewegt, während der Standfuß auf dem Boden bleibt (siehe auch Abbildung A.1(b)). Der CoP in x-Richtung (CoP_x) ist beim Zeitpunkt $t_p = -1$ nahe bei 0, weil der Roboter mit beiden Füßen den Boden berührt. Hebt sich der Schwingfuß, muss der Standfuß die Gewichtskraft des Roboters tragen, der Schwerpunkt des Roboters wandert in die Richtung des Schwingfußes und der CoP_x wird größer. Nähert sich der Schwingfuß wieder der Roboter-Mitte beim Absetzen, verkleinert sich auch der CoP_x. Bei einem Stoß nach vorne erreicht der CoP_y sehr schnell den Rand des Stabilitätsgebietes bei $CoP_y = 1$.

Torsotranslationsgeschwindigkeit: Es ist leicht ersichtlich, dass bei einem Stoß gegen den Torso die Torsotranslationsgeschwindigkeit einen Sturz gut aufzeigen kann. Beim normalen Gehen ist die Torsotranslationsgeschwindigkeit in x-Richtung (v_{T_x})[1] relativ gering, bei einem seitlichem Stoß würde sie größer werden. Die Torsotranslationsgeschwindigkeit in y-Richtung (v_{t_y}) ist bei einem Stoß nach vorne oder hinten aussagekräftig. In der normalen Schrittperiode schwankt diese aber schon relativ stark. Sie wächst während der Schrittphase mit einem kleinen lokalen Maximum bei $t_p = 0$ und nimmt zum Ende der Schrittphase – kurz bevor der Schwingfuß wieder den Boden berührt – rapide zu . Bei einem Sturz wächst die Torsotranslationsgeschwindigkeit allerdings schneller und ist so von dem normalen Gehen unterscheidbar.

Torsorotationsgeschwindigkeit: Die Torsorotationsgeschwindigkeit $d\phi$ beschreibt die Rotation um die x-Achse. Bei Stößen nach vorne oder hinten wächst die Rotationsgeschwindigkeit stärker als die Translationsgeschwindigkeit, weil der Roboter in der Tat stärker rototiert. Die Rotationsgeschwindigkeit $d\psi$ beschreibt die Rotation um die y-Achse und steigt bei seitlichen Stößen an. Die Rotationsgeschwindigkeit $d\eta$ um die z-Achse ist nur zu späteren Zeitpunkten verwendbar, wenn der Roboter fällt, mit einem Fuß aufschlägt und sich dann um die z-Achse dreht.

Fußwinkel und Fußwinkelgeschwindigkeit: Die Fußwinkel (q_{F_x} und q_{F_y}) und Fußwinkelgeschwindigkeiten (dq_{F_x} und dq_{F_y}) in x- und y-Richtung können ähnlich wie die Rotationsgeschwindigkeiten die Stürze beschreiben.

${}_{(\mathrm{T})}\vec{\mathbf{g}}$-Vektor Der ${}_{(\mathrm{T})}\vec{g}$ -Vektor beschreibt die Erdbeschleunigung $\vec{g}$ innerhalb des körperfesten Koordinatensystems ${}_{(T)}K$ des Torsos.

Schrittphase: Die Schrittphase wird in Abschnitt 2.5 erläutert und läuft während eines Schrittes von $-1 \ldots +1$. Mit der Schrittphase als Merkmal wird der zeitliche Aspekt eines Sturzes berücksichtigt, weil an manchen Zeitpunkten die Stürze je nach Beinkonfiguration anders verlaufen.

1 Das Koordinatensystem des Roboters ist in Abbildung 2.3 zu sehen. Die y-Koordinate beschreibt die Richtung nach vorne und die x-Koordinate die Richtung zur Seite.

3.4 Klassenerstellung

Die Klassen mit ihren Prototypen sollen die verschiedenen Fall-Vorgänge gut beschreiben. Da die Stöße in der Simulation senkrecht auf den Seiten des Roboter-Torso angebracht werden, sind die Vorzugsfallrichtungen die Fußkante des Standfußes an der entgegengesetzten Seite des Roboters. So fällt der Roboter bei einem Stoß von vorne über die hintere Fußkante des Standfußes. Deshalb werden vier verschiedene Fall-Richtungen festgelegt:

$$\text{Fall-Richtungen} \in \{\text{Fallen nach vorn}, \text{Fallen zurück}, \text{Fallen über STF}, \text{Fallen über SWF}\}.$$

Da die Stöße unterschiedliche Intensitäten haben, können stärkere Stöße den Roboter auch schneller zu Fall bringen. Deshalb werden zwei Fall-Intensitäten festgelegt:

$$\text{Fall-Intensität} \in \{\text{OK}, \text{Sturz}\}.$$

Bei der OK-Intensität kann der Roboter mit der Standardreaktion (siehe Tabelle 2.1) den Fall vermeiden. Die Sturz-Intensitäten beschreiben im Unterschied zu den OK-Intensitäten die Fallszenarien, bei denen mit der Standardreaktion der Sturz nicht mehr vermieden werden kann.

In der Kombinationen der vier Fall-Richtungen und zwei Fall-Intensitäten werden acht Klassen gebildet. Zu diesen Fall-Klassen wird noch eine *normal*-Klasse hinzugefügt, in der für den Roboter keine Gefahr besteht. Der normal-Klasse werden die normale ungestörte Laufbewegung und Bewegungen nach geringe Stoßstärken zugerechnet, die dem Roboter nichts anhaben können.

In Abbildung 3.3 sind die Kombinationen dieser Klassen zum besseren Verständnis dargestellt.

Nachdem die Klassen und ihre Bedeutung nun festgelegt sind, gilt es die Prototypen der Klassen anzulernen. Dazu werden aus mehreren Stoßszenarien kurz nach dem Stoßbeginn mehrere Merkmalsvektoren aufgezeichnet. Diese Merkmalsvektoren werden je nach Fall-Verlauf den definierten Klassen zugeordnet. Aus diesen Merkmalsvektoren werden wie in Abschnitt 3.2 die Prototypen errechnet.

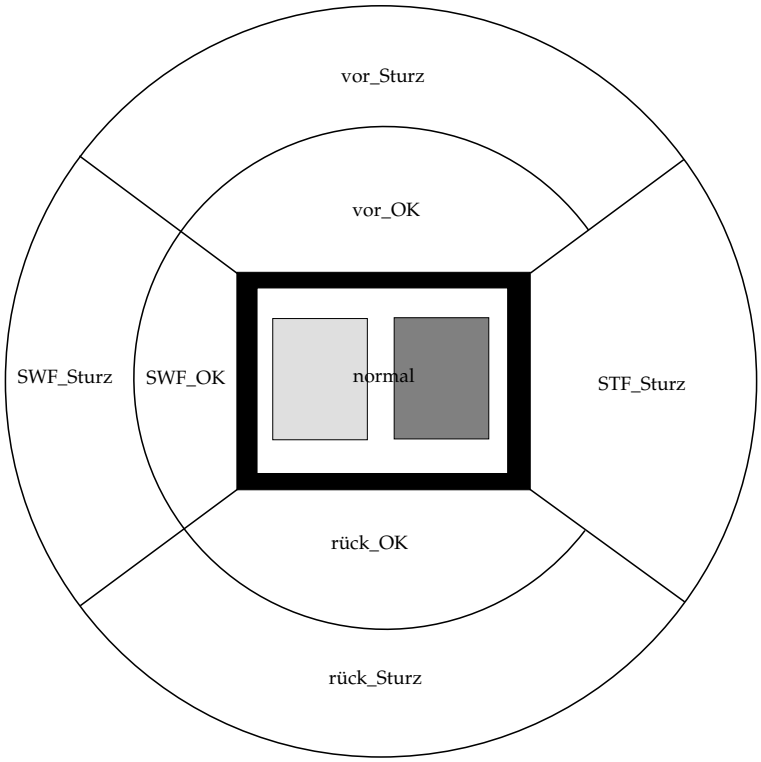


Abbildung 3.3: Diese Abbildung visualisiert die Klassendefinitionen. Der zweibeinige Roboter ist schematisch von oben dargestellt. Um den Roboter befinden sich die Fall-Klassen. Je weiter die Klasse von dem Roboter entfernt ist, desto intensiver ist die Fall-Bewegung, mit der die jeweilige Klasse korrespondiert. In dem ersten Kreissegment um den Roboter befinden sich die OK-Klassen, bei denen die Standardreaktion des Roboters das Fallen verhindern kann. In dem äußeren Ring sind die Sturz-Klassen, bei denen der Sturz nicht mehr vermieden werden kann.

- Die schon in Abschnitt 2.5 genannten Gedanken zur Eigenstabilität werden hier wieder aufgegriffen. Bei Stößen, bei denen der Roboter auch ohne Reaktion nicht fällt (Abbildung 2.4(a)), muss der Roboter auch keine Reaktion ausführen. Dazu wird beim Anlernen die Ebene, auf der der Roboter läuft, um die Winkel $\alpha \in \{-4^\circ \ldots +4^\circ\}$ gekippt und es werden so hohe Unebenheiten auf der Ebene simuliert, so dass der Roboter gerade noch nicht fällt. Die Merkmalsvektoren, die während der simulierten Gehbewegung aufgenommen worden sind, werden der **normal-Klasse** zugerechnet, um den Roboter robuster gegenüber solchen geringen Störungen zu machen. Wenn die Prototypenklassifikation diese normal-Klasse als die wahrscheinlichste Klasse befindet, soll der Roboter keine Reaktion mit den Füßen ausführen.

- Nun wird die Stabilitätsanalyse der SmgR[2] betrachtet (Abbildung 2.4(b)). Für jede Stoßrichtung wird der größte Stoßimpuls festgehalten, bei dem mit der gesteuerten Standardreaktion der Sturz noch vermieden werden kann. Je nach Schrittphase kann dies ein anderer Wert sein. Die Merkmalsvektoren, die während der dazu gehörenden Simulation aufgezeichnet worden sind, werden den **OK-Klassen** zugeordnet. Bei der online-Anwendung der Prototypenklassifikation soll der Roboter die jeweilige Standardreaktion ausführen, wenn die jeweilige OK-Klasse die wahrscheinlichste ist. Mit dieser Reaktion sollte der Sturz vermeidbar sein. Da der Roboter nach einem Stoß über den Standfuß sehr labil ist, wird die Klasse STF_OK nicht vergeben.

- Für die **Sturz-Klassen** werden die Merkmalsvektoren der Simulationen zum Anlernen benutzt, bei denen der Roboter in der SmgR den Sturz nicht vermeiden konnte. An diese Klassen wird die Schutzreaktion Hocke gebunden.
  Beim Anlernen der Klassen in einer Fallrichtung muss mit Umsicht unterschiedenen werden, ob der Merkmalsvektor einer Simulation der OK- oder der Sturz-Klasse zugeordnet werden soll. Deshalb sollte für das Anlernen der Sturz-Klassen Merkmalsvektoren aus Simulationen mit hohen Stoßimpulsen gewählt werden. Somit ist gewährleistet, dass der Roboter die OK- und Sturz-Klassen zuverlässig differenzieren kann.

Aus diesen Überlegen ergeben sich die Stoßimpulse, mit denen die Prototypen der Klassen angelernt werden. In Tabelle 3.1 sind diese Stoßimpulse aufgeführt. Dazu werden die Zeitpunkte $t_p \in \{-0{,}9 \quad -0{,}8 \quad -0{,}7 \quad \ldots \quad 0{,}7 \quad 0{,}8 \quad 0{,}9\}$ für das Einleiten der Stöße gewählt. Somit sind Merkmalsvektoren von einer Vielzahl von Schrittphasen vorhanden.

In Abschnitt 2.5 hat sich gezeigt, dass der Stoßzeitpunkt ein wichtiger Parameter eines möglichen Sturzes ist. So ist zum Beispiel bei einem Stoß von vorne mit der SmgR mit einem Stoßimpuls $p = 36\,\mathrm{Ns}$ in späteren Schrittphasen der Sturz mit der Standardreaktion häufiger vermeidbar als in frühen (siehe Abbildung 2.4(b)). Deshalb werden alle Klassen in zwei Versionen für die frühe ($-1 \leq t_p \leq 0$) und späte Schrittphase ($0 \leq t_p \leq 1$) aufgeteilt. Die Klassen, die mit Merkmalsvektoren aus Simulationen mit Stößen aus der frühen Schittphase angelernt werden, erhalten als Präfix vor dem Klassennamen ein ünd die späten ein "+". Somit ergeben sich 16 Klassen (siehe Tabelle 3.2).

2 Bei der Simulation mit gesteuerten Reaktion führt der Roboter die jeweilige Standardreaktion immer 120 ms nach dem Beginn des Stoßes aus.

		Stoßrichtungen			
		von hinten	von vorne	über STF	über SWF
Klassen	normal	3 Ns	5 Ns	4 Ns	4 Ns
	OK	18 Ns	24 Ns	–	12 Ns
	Sturz	34 Ns	42 Ns	20 Ns	20 Ns

Tabelle 3.1: Die Simulationen werden mit den aufgeführten Stoßimpulsen durchgeführt. Mit den Merkmalsvektoren aus diesen Simulationen werden die Klassen angelernt. Außerdem wird die normale ungestörte Laufbewegung der normal-Klasse zugewiesen.

			Fallrichtungen			
			nach vorne	zurück	über STF	über SWF
Klassen	$t_p > 0$	normal	+normal			
		OK	+vor_OK	+rück_OK	–	+SWF_OK
		Sturz	+vor_Sturz	+rück_Sturz	+STF_Sturz	+SWF_Sturz
	$t_p < 0$	normal	-normal			
		OK	-vor_OK	-rück_OK	–	-SWF_OK
		Sturz	-vor_Sturz	-rück_Sturz	-STF_Sturz	-SWF_Sturz

Tabelle 3.2: Die in der Prototypenklassifikation genutzten Klassen mit der Unterscheidung von zwei Schrittphasen.

Während der Simulation sind die Merkmalsvektoren zu verschiedenen Zeitpunkten gespeichert worden. Es sind die Messwerte in dem Zeitraum 20 ms vor, während der Stoßdauer von 60 ms und 15 ms nach dem Stoß aufgezeichnet worden. Es wurden nun zwei verschiedene Klassifikatoren angelernt:

- Ein Set aus Prototypen ist den Merkmalsvektoren 30 ms, 35 ms und 40 ms nach dem Beginn des Stoßes – also während des Stoßes – angelernt worden. Die Prototypenklassifikation, die diesen Prototypen benutzt, wird im Folgenden mit $\mathbf{PK^{ws}}$ (**P**rototypen-**K**lassifikation **w**ährend des **S**toßes) abgekürzt.
- Ein zweites Set an Prototypen ist den Merkmalsvektoren 70 ms und 75 ms ms nach dem Beginn des Stoßes – also nach dem Ende des Stoßes – angelernt worden. Auch die Prototypenklassifikation mit diesen Prototypen hat eine Abkürzung, nämlich $\mathbf{PK^{ns}}$ (**P**rototypen-**K**lassifikation **n**ach dem **S**toß).

Diese beiden Prototypenklassifikationen werden in dem Benchmark in Abschnitt 6.3 ver-

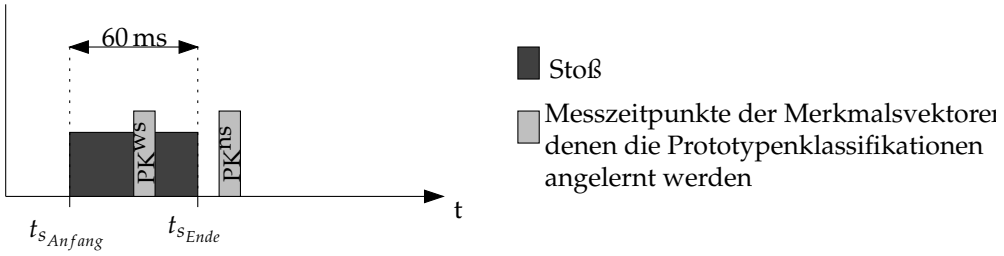


Abbildung 3.4: Diese Abbildung zeigt ein Zeitdiagramm mit dem Stoß und den Messzeitpunkten für die Merkmalsvektoren der PK^{ws} und PK^{ns}.

glichen.

3.5 Anlernen der Prototypen

In Abschnitt 3.4 sind die Parameter für die Simulationen erläutert worden, mit denen die Prototypen der Klassen angelernt werden. In Folgenden wird erklärt, wie das Anlernen durchgeführt wird.
Die Robotersimulation wird durch das Perl-Skript (Abschnitt 2.2) mit den verschiedenen Stoßparametern Stoßimpuls, Stoßrichtung und Stoßzeitpunkt gestartet. Die Simulation wird durchgeführt und nach der Simulation werden die Merkmalsvektoren, die Stoßrichtung und die Sturzvermeidung gespeichert und werden nach dem Ende der Simulationen mit einem MATLAB-Skript geladen. Hier können die Messzeitpunkte der Merkmalsvektoren ausgewählt und diese Merkmalsvektoren werden je nach Stoßrichtung, Schrittphase und Sturzvermeidung den Klassen zugeordnet. Die Prototypen werden für jede Klasse durch Mittelwertbildung in MATLAB erstellt; außerdem wird die Standardabweichung errechnet. Diese errechneten Werte werden in eine Textdatei geschrieben, die der Simulator einliest und für die online-Berechnung der Abstände zu den Prototypen der Klassen verwendet.

3.6 Klassifikation in der Roboter-Simulation

Die Prototypenklassifikation wird in der Roboter-Simulation alle 10 ms durchgeführt. Dabei findet die Abstandstandsberechnung von dem in der Simulation aktuell generierten Merk-

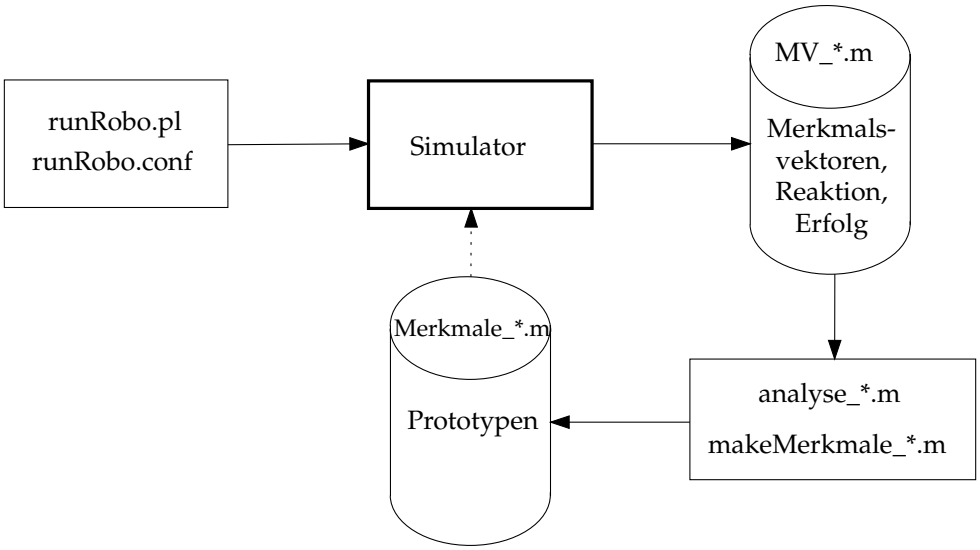


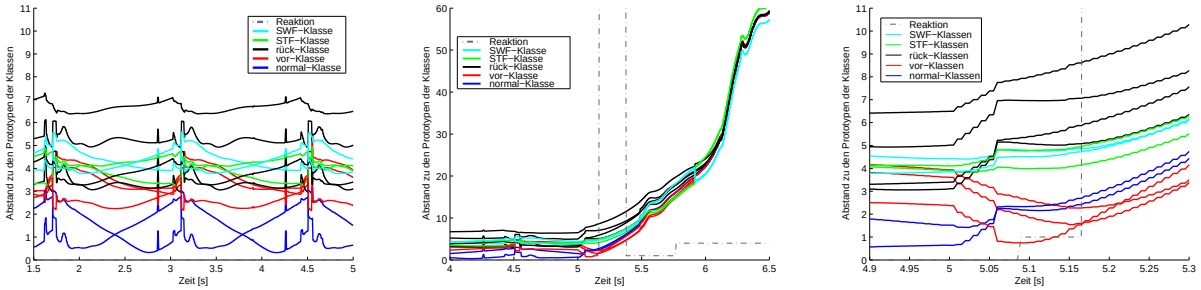
Abbildung 3.5: Übersicht über die Dateien, mit denen die Prototypen angelernt werden.

malsvektor und dem skalierten Prototypenvektor jeder Klasse nach dem Euklidischen Abstandsmaß Anwendung. Nun sind folgende zwei Modi möglich:

- Im normalen Gehen ohne Störung haben die beiden normal-Klassen *-normal* und *+normal* abwechselnd in der Schrittperiode den geringsten Abstand (siehe Abbildung 3.6(a)). Die OK- und Sturz-Klassen haben einen größeren Abstand. Dabei haben die vor-Klassen einen relativ geringen Abstand, weil der Roboter im normalem Laufen auch eine Vorwärtsbewegung durchführt. Die rück-Klassen haben den größten Abstand, weil die Bewegung beim Fallen nach hinten die entgegengesetzte Richtung hat. Innerhalb jeder Schrittperiode sind diese Abstandsverläufe sehr ähnlich.

- In Abbildung 3.6(b) sind die Abstandsverläufe der Klassen nach einem Stoß von hinten bei 5.0 s zu sehen. Sobald der Roboter gestoßen wird, sinkt der Abstand zumindest einer OK- oder Sturz-Klasse (in Abbildung 3.6(c) die rote Kurve einer vor-Klasse). Gleichzeitig wächst der Abstand der normal-Klassen und der anderen Klassen, die nicht diese Fall-Richtung beschreiben. Es hat sich herausgestellt, dass erst vier Mal hintereinander (40 ms lang) die normal-Klassen nicht den geringsten Abstand haben sollten, bevor die Standardreaktion der jeweiligen Klasse mit dem geringsten Abstand ausgeführt wird. Ansonsten gibt häufig Fehlklassifizierungen, wenn eine Klasse für einen sehr kurzen Zeitraum die Klasse mit dem geringsten Abstand ist. Bei 5.09 s hat eine vor_OK-Klasse den geringsten Abstand und es würde die Standardreaktion

ausgeführt werden. Da in diesem Beispiel keine Reaktion ausgeführt wird, fällt der Roboter weiter. Die Abstände aller Klassen werden größer bis auf die vor_Sturz-Klassen. So würde bei 5.16 s die Schutzreaktion ausgeführt werden.



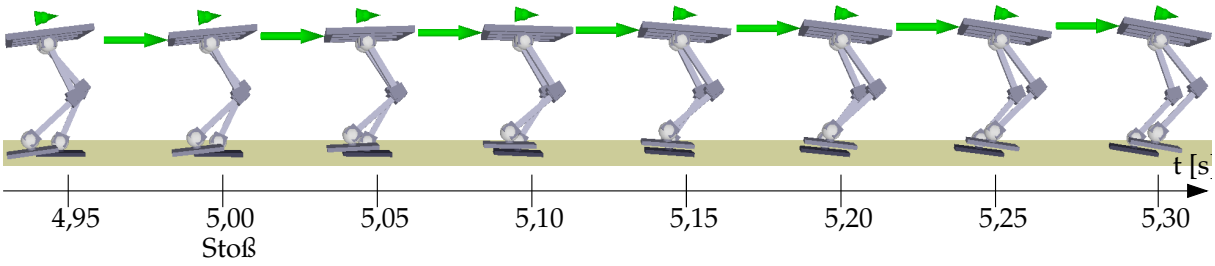
(a) normaler Schritt

(b) Stoß von hinten bei 5.0 s ($t_p = -0{,}3$) mit einem Impuls $p = 18\,\mathrm{Ns}$

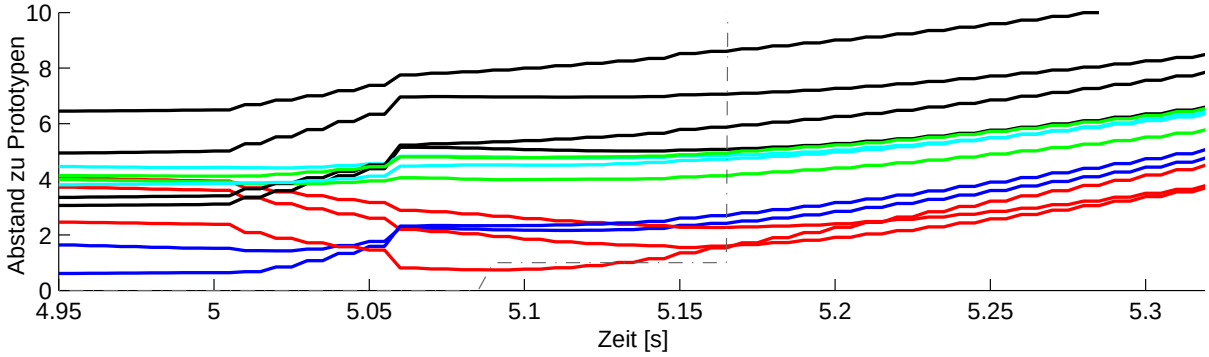
(c) Stoß von hinten bei 5.0 s ($t_p = -0{,}3$) mit einem Impuls $p = 18\,\mathrm{Ns}$

Abbildung 3.6: Prototypenklassifikation mit $\mathrm{PK}^{\mathrm{ns}}$. Die Abstände der Klassen zu dem jeweiligen Merkmalsvektor sind in in Abbildungen abgetragen.

Die Bedingung, dass die normal-Klassen vier Mal hintereinander nicht den kleinsten Abstand haben sollen, bevor eine Reaktion ausgeführt wird, ist ein Erfahrungswert. Ist dieser Zeitraum zu klein, ist die Prototypenklassiflation nicht sehr robust gegenüber geringen Stoßimpulsen. Ist sie zu groß, wird die Reaktionszeit erhöht und eine Reaktion kann den Sturz nicht mehr vermeiden, weil der Roboter in der Zwischenzeit schon zu lange gefallen ist.



(a) Bildsequenz des modellierten Roboters



(b) Abstände der Prototypen der Klassen zu dem aktuellen Merkmalsvektor

Abbildung 3.7: Bildsequenz und Verläufe der Abstände der Prototypen der Klassen zum aktuellen Merkmalsvektor nach einem Stoß von hinten bei 5.0 s ($t_p = -0{,}3$) mit einem Impuls $p = 18$ Ns. Die Farben haben folgende Klassen-Bedeutung: blau: normal, rot: Fallen nach vorn, schwarz: Fallen nach hinten, grün: Fallen über SWF und cyan: Fallen über STF.

4 Hidden Markov Models

Hidden Markov Models sind stochastische Modelle, die verborgene Zustände zu erklären versuchen[1]. Die Grundlage für HMM (Hidden Markov Model wird im Folgenden auch mit *HMM* abgekürzt) sind zwei stochastische Prozesse. Der erste Prozess beschreibt das Wechseln von einem Zustand s_i in einen anderen Zustand s_j und der zweite beschreibt die Zuordnung von Beobachtungen $\vec{O}$ zu dem jeweiligen Zustand. Der erste Prozess ist nicht bekannt[2] und wird mit der Übergangswahrscheinlichkeitsmatrix **A** beschrieben, der zweite Prozess ist hingegen bekannt und wird mit der Ausgabewahrscheinlichkeitsmatrix der Beobachtungen **B** beschrieben. Die HMM werden im Folgenden mit dem Symbol λ abgekürzt.

Die Markov-Eigenschaft, die auch den Markov-Prozessen erster Ordnung ihren Namen gibt, ist Grundlage des ersten stochastischen Prozesses, der das Wechseln von einem Zustand in einen anderen Zustand beschreibt. Hierbei ist die Wahrscheinlichkeit zum Zeitpunkt t im Zustand s_i zu sein nur vom vorherigen Zustand s_j zum Zeitpunkt $t-1$ anhängig:

$$p(s_t = s_i | s_{t-1} = s_j, s_{t-2} = s_h, \dots) \quad = \quad p(s_t = s_i | s_{t-1} = s_j). \tag{4.1}$$

Die älteren Zustände ab $s_{t-2} = s_h$ beeinflussen die Wahrscheinlichkeit nicht mehr.

HMM werden seit langem erfolgreich in der Spracherkennung eingesetzt (eine Übersicht ist in [Rab89, S. 275] zu finden).

1 Der Namensgeber ist der russische Mathematiker Andrei Andrejewitsch Markow, dessen Familienname in der englischen Literatur *Markov* geschrieben wird und diese Schreibweise in der deutschen Literatur übernommen wurde.

2 deshalb das Adjektiv *hidden* im *Hidden Markov Model*.

4.1 Grundlagen

Anhand eines einfachen Beispiels sollen nun Grundelemente eines HMM aufgezeigt werden: Das HMM λ_G[3] beschreibt den Gemütszustand eines Menschen. Der Mensch kann normal, fröhlich oder traurig sein. Die möglichen Zustände $\vec{S} = (s_1, s_2, s_3)$ sind somit

$$\vec{S} = (s_1, s_2, s_3) = (\text{normal, fröhlich, traurig}). \tag{4.2}$$

Die möglichen Beobachtungen sind

$$\vec{O} = (o_1,\ o_2,\ o_3,\ o_4) = (\text{normaler Gesichtsausdruck, lachen, weinen, grinsen}). \tag{4.3}$$

Es ist ersichtlich, dass der Gemütszustand eines Menschen kompliziert sein kann und nur innerhalb des Menschen vorliegt. Zu seiner Umgebung gibt der Mensch unbewusst und bewusst von ihm steuerbare Signale – die Beobachtungen – aus. Die bewusst ausgesandten Signale können jedoch von ihm auch bewusst verändert werden: So lacht ein Mensch zum Beispiel bewusst über einen Witz, den er gar nicht lustig findet. Er lacht nämlich aus Gefälligkeit gegenüber dem Witzerzähler, ist in Wirklichkeit aber gelangweilt.

In der Übergangswahrscheinlichkeitsmatrix **A** werden die Wahrscheinlichkeiten des Übergangs von einem Zustand zu einem anderen Zustand angegeben:

$$\mathbf{A}_G = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 0{,}7 & 0{,}2 & 0{,}1 \\ 0{,}4 & 0{,}6 & 0{,}0 \\ 0{,}3 & 0{,}0 & 0{,}7 \end{pmatrix}. \tag{4.4}$$

Die Wahrscheinlichkeit, dass der Zufallsprozess von Zustand $s_2 =$ fröhlich zu Zustand $s_1 =$ normal wechselt ist $a_{21} = 0{,}4$. Die Wahrscheinlichkeit a_{23} ist hingegen 0, weil ein Wechsel von *fröhlich* zu *traurig* praktisch nicht eintritt. In Abbildung 4.1 sind die Zustände mit den Übergangswahrscheinlichkeiten dargestellt.

In der Ausgabewahrscheinlichkeitsmatrix **B** wird die Wahrscheinlichkeit festgelegt, dass die Beobachtung bei einem Zustand auftritt:

3 Im Folgenden werden Beispiel-Parameter für das Beispiel-HMM angegeben. Die Parameter haben den Index G für Gemüt.

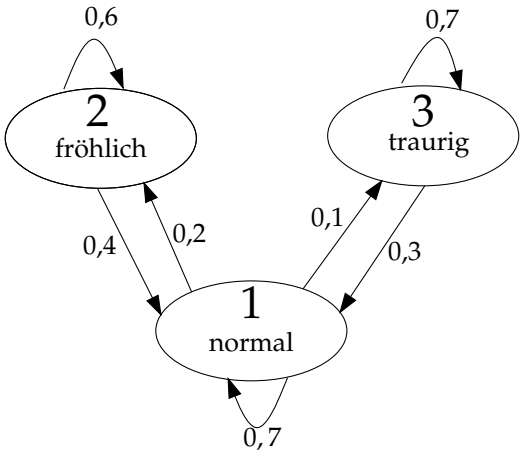


Abbildung 4.1: Graph des Gefühls-HMM mit den Übergangswahrscheinlichkeiten an den Kanten

$$\mathbf{B}_G = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{pmatrix} = \begin{pmatrix} 0{,}6 & 0{,}2 & 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}7 & 0{,}0 & 0{,}2 \\ 0{,}1 & 0{,}0 & 0{,}8 & 0{,}1 \end{pmatrix}. \tag{4.5}$$

Die Ausgabewahrscheinlichkeit b_{13} ist die Wahrscheinlichkeit, dass die Beobachtung o_3 im Zustand s_1 auftritt, was in diesem Beispiel $b_{13} = 0{,}1$ ist. Die Wahrscheinlichkeit im Zustand $s_1 =$ normal die Beobachtung $o_3 =$ weinen zu sehen ist 0,1. Diese Kombination ist also nicht sehr wahrscheinlich.

Außerdem gibt es als besonderen Fall der Übergangswahrscheinlichkeiten die Startwahrscheinlichkeiten der Zustände:

$$\vec{\pi}_G = \begin{pmatrix} \pi_1 & \pi_2 & \pi_3 \end{pmatrix} = \begin{pmatrix} 0{,}8 & 0{,}1 & 0{,}1 \end{pmatrix}. \tag{4.6}$$

Zum Startzeitpunkt des HMM soll also der normale Zustand mit der Wahrscheinlichkeit $\pi_1 = 0{,}8$ vorliegen.

Die Wahrscheinlichkeiten in einer Zeile dieser Matrizen müssen sich zu 1 summieren:

$$\sum_{j}^{n} a_{ij} = 1 \qquad \text{für jeden Zustand i} \tag{4.7}$$

$$\sum_{j}^{n} b_{ij} = 1 \qquad \text{für jede Beobachtung i} \tag{4.8}$$

$$\sum_{i}^{n} \pi_i = 1. \tag{4.9}$$

Ein HMM wird also vollständig mit diesen drei Matrizen beschrieben:

$$\lambda = (\mathbf{A}, \mathbf{B}, \vec{\pi}). \tag{4.10}$$

4.2 Die drei Probleme der Hidden Markov Models

Für die Benutzung von HMM müssen viele Berechnungen sowohl offline und online durchgeführt werden. In Abhängigkeit von der Aufgabenstellung müssen bis zu drei Probleme nach [Rab89] gelöst werden. Die folgenden Ausführungen sind nach [Rab89] und [MS01] geschrieben:

Problem 1: Die Wahrscheinlichkeit $p(\vec{O}|\lambda)$ einer Beobachtungssequenz $\vec{O} = (o_1, o_2, \ldots, o_T)$ bei einem gegebenen HMM $\lambda = (\mathbf{A}, \mathbf{B}, \vec{\pi})$ muss berechnet werden.

Problem 2: Die Zustandssequenz $\vec{S} = (s_1, s_2, \ldots, s_T)$ ist zu finden, die die Beobachtungssequenz $\vec{O} = (o_1, o_2, \ldots, o_T)$ bei einem gegebenen HMM $\lambda = (\mathbf{A}, \mathbf{B}, \vec{\pi})$ am besten erklärt.

Problem 3: Die Parameter $(\mathbf{A}, \mathbf{B}, \vec{\pi})$, die ein HMM λ beschreiben, müssen auf die jeweilige Anwendung optimiert werden.

4.2.1 Forward- und Backwardvariable

Auf dem Weg zum Lösen dieser drei Probleme hat sich gezeigt, dass viel Rechenzeit eingespart werden kann, wenn für die Wahrscheinlichkeitsberechnungen Hilfsvariablen eingeführt werden, die Zwischenberechnungen der Wahrscheinlichkeiten speichern. Diese Hilfsvariablen werden in zweidimensionale Arrays gespeichert. Die Dimensionen dieser Arrays sind Zeit und Zustand. In solch einem Array wird also eine bestimmte Wahrscheinlichkeit in Abhängigkeit von Zustand und Zeit gespeichert.

Die **Forwardvariable** $\alpha_t(i) = p(o_1, o_2, \cdots, o_t, s_t = i|\lambda)$ beschreibt bei einem gegebenen HMM λ die Wahrscheinlichkeit einer Beobachtungsteilsequenz $\vec{O} = (o_1, o_2, \cdots, o_t)$ nicht bis zum Ende T, sondern nur bis zum Zeitpunkt $t < T$, dass der Zustand $s_t = i$ zu diesem Zeitpunkt t vorliegt. Die Forwardvariable wird folgendermaßen berechnet:

1. Zuerst wird die Forwardvariable für den Zeitpunkt $t = 1$ initialisiert. Dafür wird die Wahrscheinlichkeit der ersten Beobachtung o_1 mit der Startwahrscheinlichkeit π_i für jeden Zustand $1 \leq i \leq N$ multipliziert und gespeichert:

$$\alpha_1(i) = \pi_i b_i(o_1), \qquad \text{für} \quad 1 \leq i \leq N \tag{4.11}$$

2. Als zweites folgt der Induktionsschritt (Gleichung 4.12), in dem die eigentliche Berechnung durchgeführt wird. In dem Summationsterm wird die alte Forwardvariable mit der Übergangswahrscheinlichkeit vom alten Zustand i zum neuen Zustand j multipliziert, so dass Zustand j erreicht wird. Dieser Term wird für alle möglichen alten Zustände i aufsummiert. Diese Summe wird mit der Beobachtungswahrscheinlichkeit für die neue Beobachtung o_{t+1} multipliziert, die im neuen Zustand s_j beobachtet wird. Hieraus ergibt sich die Forwardvariable für den nächsten Zeitpunkt $t + 1$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}) \qquad \text{für} \quad 1 \leq t \leq T-1 \qquad \text{und für } 1 \leq j \leq N. \tag{4.12}$$

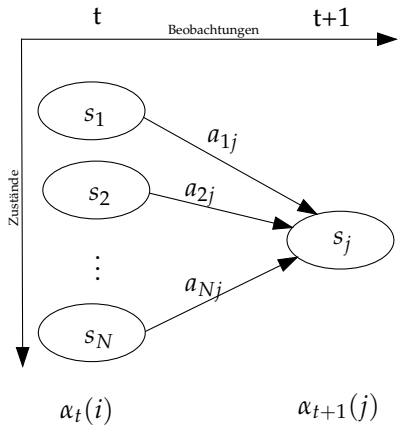


Abbildung 4.2: Induktionsschritt zur Berechnung der Forwardvariablen α

3. In einem abschließenden Schritt können die letzten Forwardvariablen zum Zeitpunkt

T addiert werden und es ergibt sich die bedingte Wahrscheinlichkeit

$$p(\vec{O}|\lambda) = \sum_{i=1}^{N} a_T(i). \tag{4.13}$$

Die Berechnung der **Backwardvariablen** $\beta_t(i) = p(o_{t+1}, o_{t+2}, \cdots, o_T|s_t = i, \lambda)$ verläuft analog zu der Berechnung der Forwardvariablen, allerdings wird hier die Beobachtungssequenz $\vec{O}$ in der Zeit rückwärts durchgearbeitet. Die Backward-Variable $\beta_t(i)$ beschreibt die bedingte Wahrscheinlichkeit der Beobachtungssequenz $\vec{O} = (o_{t+1}, o_{t+2}, \cdots, o_T)$ von t bis zum Ende T bei gegebenem Zustand s_T und HMM λ.

1. Die Initialisierung ist willkürlich festgelegt:

$$\beta_T(i) = 1 \qquad \text{für} \qquad 1 \leq i \leq N. \tag{4.14}$$

2. Im Induktionsschritt wird von späteren Beobachtungen auf frühere Beobachtungen geschaut und dabei werden die Beobachtungs- und Übergangswahrscheinlichkeiten im Übergang vom späteren zum früheren Zustand durch Multiplikation miteinander verbunden:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \qquad \text{für} \quad t = T-1, T-2 \cdots 1 \quad \text{und für } 1 \leq i \leq N. \tag{4.15}$$

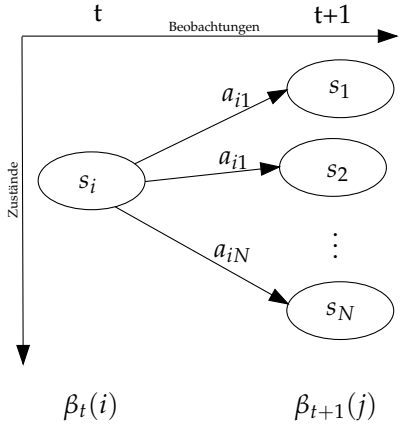


Abbildung 4.3: Induktionsschritt zur Berechnung der Backwardvariablen β

4.2.2 Lösung von Problem 1: Wahrscheinlichkeitsberechnung

Problem 1 kann direkt mit den Forward- und Backwardvariablen gelöst werden. Die bedingte Wahrscheinlichkeit der Beobachtungssequenz $\vec{O}$ und dem Zustand i zum Zeitpunkt t $s_t = i$ bei gegebenem HMM λ lässt sich durch Aufteilen der Beobachtungsseqzenz in zwei Teile durch die Forward- und Backward-Variablen ausrücken:

$$\begin{aligned} p(\vec{O}, s_t = i|\lambda) &= P(o_1, o_2, \cdots, o_T, s_t = i|\lambda) \\ &= p(o_1, o_2, \cdots, o_t, s_t = i|\lambda) \cdot p(o_{t+1}, o_{t+2}, \cdots, o_T|s_T = i, \lambda) \\ &= \alpha_t(i) \cdot \beta_t(i). \end{aligned} \tag{4.16}$$

Hieraus ergibt sich dann:

$$p(\vec{O}|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \cdot \beta_t(i) \quad \text{für} \quad 1 \leq t \leq T+1. \tag{4.17}$$

4.2.3 Lösung von Problem 2: Der Viterbi-Algorithmus

Problem 2 war, die beste Zustandssequenz $\vec{S}$ bei einer gegebenen Beobachtungssequenz $\vec{O}$ und einem gegebenen HMM λ zu finden. Da es mehrere Möglichkeiten geben kann, die *beste* Zustandssequenz zu finden, kann dieses Problem auf verschiedene Weisen gelöst werden. Es hat sich der Viberbi-Algorithmus als sinnvolle Lösungsmöglichkeit herausgestellt:

Dazu werden zwei Hilfsvariablen erstellt: $\vec{\eta}$ ist die im Algorithmus bestimmte Zustandssequenz, die die größte Wahrscheinlichkeit ϕ bei der gegebenen Beobachtungssequenz $\vec{O}$hat.

Bei der Initialisierung werden ϕ und $\vec{\eta}$ folgende Werte zugewiesen:

$$\phi_i = \pi_i \qquad \eta_i = 0 \qquad \text{für alle} \qquad 1 \leq i \leq N. \tag{4.18}$$

Im Induktionsschritt wird für alle Beobachtungen und Zustandsübergangskombinationen von Zustand s_i zu s_j die Wahrscheinlichkeit ϕ maximiert. Ist diese für eine Beobachtung durchgeführt worden, wird die Wahrscheinlichkeit für den nächsten Durchlauf in ϕ_{i+1}

und der Zustand zu dieser höchsten Wahrscheinlichkeit in η_{i+1} gespeichert und es wird zur nächsten Beobachtung weitergegangen:

$$\begin{aligned} \phi_{i+1} &= \phi_i \cdot \max[b_i(o_{t+1}) \cdot a_{ij}] \qquad \text{für alle} \quad 1 \leq t \leq T+1; \quad 1 \leq i,j \leq N \\ \eta_{i+1} &= \arg\max(\phi_{i+1}). \end{aligned} \tag{4.19}$$

In dieser Berechnung müssen $T(2N)^2$ Berechnungen (Multiplikationen) durchgeführt werden. Da dieser Algorithmus online benötigt wird, ist es also für eine schnelle Berechnung zielführend, möglichst wenig Zustände in den HMM zu definieren, da diese quadratisch in die Rechenzeit eingeht.

In Gleichung 4.19 werden viele Multiplikationen mit Wahrscheinlichkeiten durchgeführt. Da die Wahrscheinlichkeiten der Faktoren $b_i(o_{t+1})$ und a_{ij} sehr klein sein können, wird das Produkt ϕ sehr schnell gegen 0 gehen können. Nun besteht die Gefahr, dass ϕ nicht mehr in einem üblichen Datentypen wie ein *Double* mit einem Bereich von $5{,}0 \cdot 10^{-324} \ldots 1{,}7 \cdot 10^{308}$ dargestellt werden kann. Deshalb kann der Viterbi-Algorithmus etwas abgewandelt werden, indem mit logarithmierten Wahrscheinlichkeiten gerechnet wird. Da die Logarithmus-Funktion eine monotone Funktion ist, wird die Reihenfolge der Wahrscheinlichkeiten nicht geändert. Anstelle der Wahrscheinlichkeit $p = 10^{-100}$ wird die logarithmierte Wahrscheinlichkeit $\tilde{p} = -100$ verwandt, so dass jetzt fast jeder Gleitkommazahl-Datentyp genutzt werden kann. Außerdem hat diese Berechnung einen weiteren Vorteil, da $\log(a \cdot b) = \log(a) + \log(b)$. Es müssen also online nun keine Multiplikationen durchgeführt werden, sondern nur noch Additionen, die in der Regel schneller ausgeführt werden können. Außerdem muss das Logarithmieren der Wahrscheinlichkeiten nur einmal durchgeführt werden und kann schon offline erfolgen. Die logarithmierten Wahrscheinlichkeiten werden mit einer Tilde mit dem jeweiligen Symbol wie zum Beispiel $\tilde{p}$ dargestellt. Es wird daher in dieser Arbeit der Viterbi-Algorithmus mit logarithmierten Wahrscheinlichkeiten angewandt:

$$\begin{aligned} \tilde{\phi}_i &= \tilde{\pi}_i \qquad \eta_i = 0 \qquad \text{für alle} \quad 1 \leq i \leq N \\ \tilde{\phi}_{i+1} &= \tilde{\phi}_i + \max[\tilde{b}_i(o_{t+1}) + \tilde{a}_{ij}] \qquad \text{für alle} \quad 1 \leq t \leq T+1 \quad 1 \leq i,j \leq N \\ \eta_{i+1} &= \arg\max(\tilde{\phi}_{i+1}). \end{aligned} \tag{4.20}$$

Der Viterbi-Algorithmus liefert als Ergebnis die wahrscheinlichste Zustand-Sequenz in $\vec{\eta}$,

den sogenannten Viterbi-Pfad durch alle möglichen Zustandskombinationen und außerdem die Wahrscheinlichkeit dieses Viterbi-Pfades in ϕ beziehungsweise $\tilde{\phi}$.

In diesem Zusammenhang ist zu beachten, dass $\log(0)$ nicht definiert ist. Für sehr geringe Wahrscheinlichkeiten in den Matrizen **A** und **B** muss also ein beliebig hoher negativer Wert eingesetzt werden.

4.2.4 Lösung von Problem 3: Parameterschätzung mit Hilfe des Baum-Welch-Algorithmus

Die Parameter eines HMM $\lambda = \lambda(\mathbf{A}, \mathbf{B}, \vec{\pi})$ müssen auf die jeweilige Anwendung angepasst werden. Die Beobachtungen sind aus den Zufallsprozessen produziert worden und liegen vor. Die Wahrscheinlichkeit, dass die Beobachtungen zu dem HMM gehören, sind zu maximieren, wobei über die Parameter maximiert wird:

$$\max_{\mathbf{A},\mathbf{B},\vec{\pi}} p(\vec{O}|\lambda(\mathbf{A}, \mathbf{B}, \vec{\pi})). \tag{4.21}$$

Leonard Baum und seine Kollegen haben hierfür auf Basis der Erwartungs-Maximierungs-Methode (EM-Methode) einen Algorithmus entwickelt, die iterativ die Parameter maximiert [Rab89, BPSW70]. Dieser Algorithmus wird nach seinen Entwicklern Baum-Welch-Algorithmus benannt.

Neben den Forward- und Backwardvariablen α und β werden hier zwei weitere Hilfsvariablen eingeführt:

$\xi_t(i,j) = p(s_t = i, s_{t+1} = j|\vec{O}, \lambda)$ ist die Wahrscheinlichkeit zum Zeitpunkt t im Zustand i und im nächsten Zeitpunkt $t+1$ in Zustand j zu sein. $\gamma_t(i) = p(s_t = i|\vec{O}, \lambda)$ ist die Wahrscheinlichkeit zum Zeitpunkt t im Zustand i zu sein. $\gamma_t(i)$ kann durch $\xi_t(i,j)$ ausgedrückt werden: $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$, denn $\xi_t(i,j)$ beschreibt die Wechsel von Zustand i nach j und $\gamma_t(i)$ nur das Betreten des Zustandes i, so dass über alle N Zustände j summiert werden kann.

$\xi_t(i,j)$ kann mit Hilfe der Forward- und Backward-Variablen α und β berechnet werden. Mit dem Zählerterm $\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)$ wird der Wechsel von Zustand i zu Zustand j

mit den daran beteiligten Übergangs- und Beobachtungs-Wahrscheinlichkeiten und den Forward- und Backward-Variablen beschrieben (siehe auch Abbildung 4.4):

$$\begin{aligned} \xi_t(i,j) &= p(s_t = i, s_{t+1} = j|\vec{O}, \lambda) \\ &= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(\vec{O}|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}. \end{aligned} \tag{4.22}$$

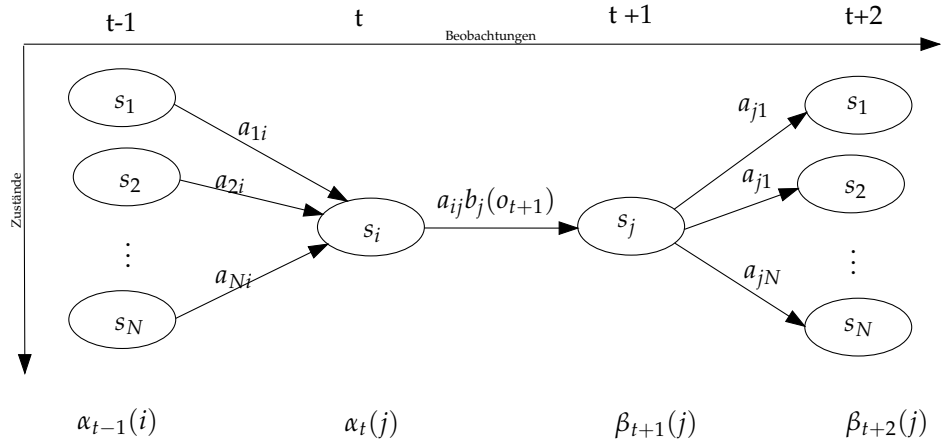


Abbildung 4.4: Berechnung von ξ mit Hilfe der Forward- und Backward-Variablen α und β

Wenn $\gamma_t(i)$ über alle Beobachtungen, also über alle Zeitpunkte, aufsummiert wird, ergibt sich ein Maß für die Anzahl aller Übergänge von Zustand i: $\sum_{t=1}^{T-1}\gamma_t(i)$. Dabei muss der letzte Zeitpunkt T ausgelassen werden, da von diesem ja nach der Definition kein Wechsel zu einem anderen Zustand mehr möglich ist, weil s_T der letzte Zustand ist. Es liegt nahe, das Gleiche mit $\xi_t(i,j)$ zu durchzuführen: $\sum_{t=1}^{T-1}\xi_t(i,j)$ beschreibt die Anzahl der Übergänge von Zustand i zu j.

Diese Summationsterme sind nun die Grundlage für die Parameterschätzungen.

Für die geschätzte Startwahrscheinlichkeit $\hat{\pi}_i$ der Zustände i wird $\gamma_{t=1}(i)$ nur zum Zeitpunkt $t = 1$ betrachtet:

$$\hat{\pi}_i = \gamma_{t=1}(i). \tag{4.23}$$

Für die geschätzten Übergangswahrscheinlichkeiten $\hat{a}_{ij}$ ergibt sich:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\text{erwartetet Anzahl von Übergängen von } s_i \text{ zu } s_j}{\text{erwartetet Anzahl von Übergängen von } s_i}. \tag{4.24}$$

Bei der geschätzten Beobachtungswahrscheinlichkeit wird nun wieder die ganze Beobachtungssequenz bis $t = T$ betrachtet, da im letzten Zustand – wie in jedem Zustand – eine Beobachtung produziert wird. Zähler- und Nennerterm unterscheiden sich nur in der Tatsache, dass im Zählerterm lediglich die Summanden addiert werden, bei der die Beobachtung k auftrat:

$$\hat{b}_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j) \qquad \text{bei Vorliegen der Beobachtung k}}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{4.25}$$

Leonard E. Baum [Rab89][BPSW70] zeigt, dass stets $p(\vec{O}|\hat{\lambda}) \geq p(\vec{O}|\lambda)$. Das HMM $\hat{\lambda}$ ist also mindestens so gut wie das ursprüngliche HMM.

Dieser Algorithmus wird nun mit den Beobachtungssequenzen, die von einem HMM möglichst gut beschrieben werden sollen, trainiert. Diese Trainingsschritte können fortlaufend mit den geschätzten Parametern aus dem vorherigen Trainingsschritt durchgeführt werden, bis sich die Parameterschätzung nicht mehr ändert und somit konvergiert ist.

Problematisch ist, dass dieser Algorithmus nur das lokale Maximum bestimmt, so dass das globale Maximum eventuell nicht erreicht wird. Deshalb ist es wichtig, schon die Startwerte für den Algorithmus sinnvoll für die Anwendung und die Struktur der HMM zu wählen.

4.3 Hidden Markov Models zur Sturzerkennung

Als Grundlage für die Klassifikation bei der Roboter-Simulation sollte zunächst ein großes HMM dienen, das verschiedene Zustände beim Gehen und Fallen des Roboters umfasst (siehe Abbildung 4.5): Die ersten vier Zuständen beschreiben im Idealfall das Gehen als

Schleife, die periodisch durchlaufen wird. Fällt der Roboter, wechselt der Viterbi-Pfad je nach Sturzrichtung zu einem der vier Äste, die die verschiedenen Fall-Richtungen beschreiben. Der letzte Zustand ist der gefallen-Zustand. Dieser Zustand ist nur bei dem Entwurf der Struktur eines möglichen HMM für die Beschreibung von Stürzen entwickelt worden. In der Anwendung ist der gefallen-Zustand nicht benutzt worden, da er nie erreicht werden soll. Der jeweilige Sturz-Zustand sagt schon implizit aus, dass der Roboter fallen wird. Außerdem müsste nach dem Bestimmen des Viterbi-Pfades der Pfad von hinten nocheinmal überprüft werden, um die Fall-Richtung zu bestimmen. Ein weiterer Nachteil ist, dass bei der online-Berechnung mit dem Viterbi-Algorithmus die Anzahl der Zustände, die durch das HMM beschrieben werden soll, quadratisch in die Rechenzeit eingeht (vergleiche Unterabschnitt 4.2.3).

Aus diesen Gründen ist dieses große HMM in mehrere kleine HMM aufgeteilt worden, die der jeweiligen Fallsituation entsprechen. Ein solches spezielles HMM für den Fall nach vorne ist in Abbildung 4.5 zu sehen.

Bei der Benutzung der HMM werden in dieser Arbeit nur diskrete Beobachtungen ausgewertet. Deshalb müssen die kinetischen und kinematischen Größen, die in der Simulation alle 0,5 ms berechnet werden, mit der Prototypenklassifikation alle 10 ms in eine Beobachtung umgewandelt werden (Vektorquantifizierung).

Es werden auch nicht alle Beobachtungen vom Anfang der Simulation benutzt, sondern nur ein Beobachtungsfenster der letzten 30 Beobachtungen. Dieses spart wiederum Rechenzeit und macht die HMM-Klassifikation auch unabhängiger von dem vorgehenden Teil der Simulation vor dem möglichen Fallen.

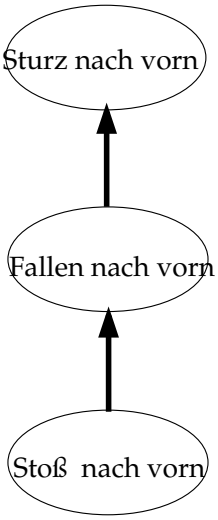
Bei der online-Anwendung des Viterbi-Algorithmus gibt es folgendes Problem: Es ist möglich, dass ein berechneter Viterbi-Pfad

$$\vec{\eta}_a = (\text{Stoß nach vorn} \quad \text{Fallen nach vorn} \quad \text{Gehen Phase 1}) \qquad \text{mit} \quad \phi_a$$

eine Störung aber dann einen normales Laufen beschreibt und dabei die höchste Wahrscheinlichkeit ϕ_a besitzt. Auf der anderen Seite gibt es einen nicht so wahrscheinlichen Viterbi-Pfad

$$\vec{\eta}_b = (\text{Stoß nach vorn} \quad \text{Fallen nach vorn} \quad \text{Sturz nach vorn}) \qquad \text{mit} \quad \phi_b < \phi_a,$$

(a) ein HMM in kompletter Ansicht



(b) ein spezielles HMM, das nur den Sturz nach vorne beschreibt

Abbildung 4.5: Graph eines Hidden Markov Models: Die Pfeile mit der dicken Strichstärke beschreiben den Zustandspfad, der durch das Modell beschrieben wird, wenn ein Sturz nach vorne auftritt. Die Pfeile mit der normalen Strichstärke beschreiben einige Möglichkeiten, wie das Modell andere Stürze beschreiben könnte.

der im Gegensatz zu $\vec{\eta}_a$ ein Stürzen beschreibt. Nun würde $\vec{\eta}_a$ wegen seiner hohen Wahrscheinlichkeit ϕ_a aber mit der nicht gewünschten Aussage dem $\vec{\eta}_b$ mit der geringeren Wahrscheinlichkeit $\phi_b < \phi_a$ und der gewünschten Aussage vorgezogen werden und es würde zum Beispiel keine Reaktion ausgeführt werden.

Deshalb wird im Folgenden der *bedingte Viterbi-Algorithmus* benutzt. Dieser benutzt einen zusätzlichen Vektor $\vec{S}_{Ende}$, der zuvor definiert werden muss. Der Viterbi-Pfad muss in einem der Zustände aus diesem Vektor $\vec{S}_{Ende}$, in dem sich zum Beispiel die Sturz-Zustände

befinden, enden:

$$\begin{array}{lcll} \tilde{\phi}_i & = & \tilde{\pi}_i \qquad \eta_i = 0 & \text{für alle} \quad 1 \leq i \leq N \\ \tilde{\phi}_{i+1} & = & \tilde{\phi}_i + \max[\tilde{b}_i(o_t) + \tilde{a}_{ij}] & \text{für alle} \quad 1 \leq t \leq T+1 \quad 1 \leq i,j \leq N, \quad s_T \in \vec{S}_{Ende} \; . \\ \vec{\eta}_{i+1} & = & \arg\max(\tilde{\phi}_{i+1}) & \end{array} \tag{4.26}$$

Im normalen Gehen beim Wechsel zwischen den Schritt-HMM gibt es leider Fehlklassifikationen. In Abbildung 4.7(a) sind die Wahrscheinlichkeiten der HMM-Klassifikation bei einem normalen Schritt zu sehen. Bei 2,7 s ist ein normal-HMM nicht mehr das wahrscheinlichste und es würde eine Reaktion ausgeführt werden. Außerdem kostet das Berechnen der HMM-Wahrscheinlichkeiten viel Rechenzeit und die HMM bieten keine Zusatzinformation. Deshalb wird ein Großteil der Rechenzeit eingespart, wenn die Berechnung des bedingten Viterbi-Algorithmus erst zu den gefährlichen Zeitpunkten gestartet wird, wenn eine Beobachtung erzeugt worden ist, die nicht dem normalen Gehen entspricht. Der bedingte Viterbi-Algorithmus berechnet für jedes HMM i die Wahrscheinlichkeit $p_i = p_i(\vec{S}|\vec{O}, \lambda)$ des bedingten Viterbi-Pfades und speichert diese Wahrscheinlichkeit für jedes HMM. Dann wird das wahrscheinlichste HMM ausgewählt und die an dieses HMM gebundene Reaktion ausgeführt.

Nun werden die Implementierungen von zwei verschiedene HMM-Klassifizierungen vorgestellt, die sich nur in der Generierung der Beobachtungen unterscheiden. Beide erhalten größtenteils die Beobachtungen aus der Prototypenklassifikation (Abschnitt 3.4). Außerdem haben beide HMM-Klassifizierungen die gleiche Struktur der HMM und können daher auf die gleiche Weise – wie in Abschnitt 4.4 beschrieben – angelernt werden.

4.3.1 Klassifikation mit den Prototyp-Hidden Markov Models

In Abbildung 4.6 ist eine Übersicht über die Struktur dieser Klassifizierung zu finden: Der Simulator erzeugt die Merkmalsvektoren, die in der Prototypenklassifikation mit den Prototypen verglichen werden. Der Prototyp mit dem geringsten Abstand wird der im Folgenden erklärten Beobachtungsgenerierung zur Verfügung gestellt:

- Für das normale Laufen wird eine Schrittperiode in vier gleichlange Zeitabschnitte geteilt. Für jeden Abschnitt wird eine Beobachtung erzeugt. Jede Schrittperiode

erzeugt also die Beobachtungen in der gleichen Reihenfolge. Die Beobachtungen für mehrere Schrittperiosen wären beispielsweise
$\vec{O}_{Laufen} = (1, 1, \ldots, 2, 2, \ldots, 3, 3 \ldots, 4, 4 \ldots, 1, 1 \ldots)$.

- Die jeweils wahrscheinlichste Prototypenklasse erzeugt eine Beobachtung. Durch diese Beobachtungen werden die Informationen der Prototypenklassifikation weiterverwendet.

- Hat eine Fall-Klasse der Prototypenklassifikation nicht den geringsten Abstand zum aktuellem Merkmalsvektor, wird die Torso-Beschleunigung nach vorne und zur Seite näher untersucht. Sind diese größer als ein Schwellwert, wird für hierfür eine Beobachtung erzeugt. Mit diesen Beobachtungen sollen kurze Stöße beschrieben werden, die den Roboter noch nicht so stark beeinflussen, dass er eine Fall-Bewegung ausführt.

Dann wird der bedingte Viterbi-Pfad für jedes HMM mit der aktuellen Beobachtungssequenz berechnet. Das HMM mit dem wahrscheinlichsten Viterbi-Pfad bestimmt, welche Reaktion der Roboter ausführt. Die Reaktion ist je nach HMM schon vorher festgelegt worden.

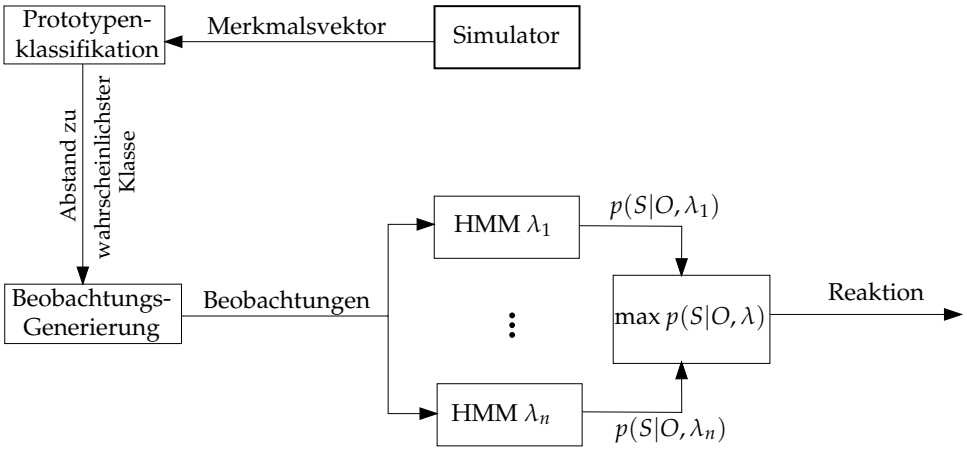
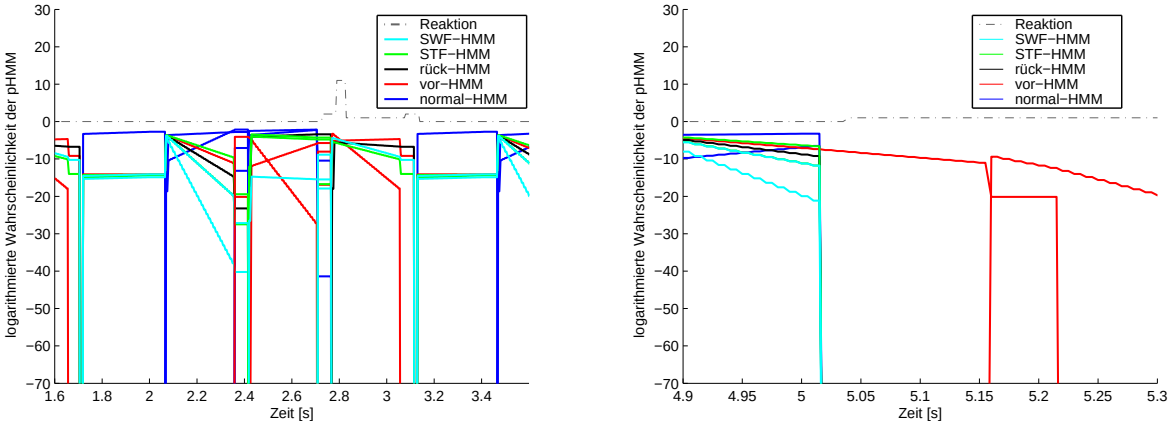


Abbildung 4.6: Blockdiagramm der $_{\text{p}}$HMMK

Dieses HMM ist somit sehr eng mit der Prototypenklassifikation verbunden. Deshalb wird die Klassifikation mit dieser Struktur im Folgenden mit $_{\mathbf{p}}\mathbf{HMMK}$ abgekürzt. Die Abkürzung $_{\mathbf{p}}\mathbf{HMMK}_5^{\mathbf{ns}}$ bedeutet, dass die Prototypenklassifikation mit dem Prototypen, der aus Merkmalsvektoren *nach dem Stoß* (*ns*) angelernt worden ist, für das *Prototyp-HMM* mit *5 Zuständen* benutzt wird.

In Abbildung 4.7(b) sind die Wahrscheinlichkeiten der HMM der $_{\text{p}}$HMMK nach einem Stoß von hinten zu sehen. 20 ms nach dem Stoß werden die Wahrscheinlichkeiten aller HMM der $_{\text{p}}$HMMK sofort sehr gering, während die Wahrscheinlichkeiten des vor-HMM der $_{\text{p}}$HMMK nur langsam geringer werden und die wahrscheinlichsten sind. Denn kurz nach dem Stoß hat eine Beobachtung, die in den anderen $_{\text{p}}$HMMK nicht antrainiert worden ist, diese sehr unwahrscheinlich gemacht. Auffällig ist, dass die Wahrscheinlichkeit der vor-$_{\text{p}}$HMMK im Zeitverlauf geringer wird, obwohl mehr antrainierte Beobachtungen generiert werden. Das liegt daran, dass bei jedem Durchlauf des Viterbi-Algorithmus zwei Multiplikationen mit Faktoren durchgeführt werden, die kleiner als 1 sind. Deshalb sinken die Wahrscheinlichkeiten. In Abbildung 4.7(a) sind die Wahrscheinlichkeiten der $_{\text{p}}$HMMK während eines ungestörten Schrittes dargestellt. In dem Zeitraum $0{,}4\,\text{s} \leq 0{,}7\,\text{s}$ wird ein vor-HMM der $_{\text{p}}$HMMK im Zeitverlauf wahrscheinlicher. Das begründet sich durch das begrenzte Beobachtungsfenster, das nur die letzten 29 Beobachtungen an den Viterbi-Algorithmus durchlässt und dabei die ältesten unwahrscheinlichen Beobachtungen verwirft.



(a) Wahrscheinlichkeiten der HMM während eines ungestörten Schrittes

(b) Wahrscheinlichkeiten der HMM nach einem Stoß von hinten bei 5.0 s ($t_p = -0{,}3$) mit einem Impuls $p = 18\,\text{Ns}$

Abbildung 4.7: Wahrscheinlichkeiten der verschiedenen HMM der $_{\text{p}}\text{HMMK}_3^{\text{ns}}$

An dem sehr schnellen Abfallen der Wahrscheinlichkeiten zeigt sich, wie stark die HMM Beobachtungen bestrafen, die ihnen nicht antrainiert worden sind. Die Schwere dieser Bestrafung kann mit den Ersatzwahrscheinlichkeiten (Abschnitt 4.4) festgelegt werden. Diese Ersatzwahrscheinlichkeiten einzelner Beobachtungs-Zustands-Kombinationen der

Ausgabewahrscheinlichkeitsmatrix **B** sind diejenigen Kombinationen, die in dem Anlernen der HMM nicht vorgesehen waren.

4.3.2 Klassifikation mit den Richtungs-Hidden Markov Models

Dieses HMM-Klassifikation HMM ähnelt der $_{\text{p}}$HMMK, allerdings wird hier nicht nur die wahrscheinlichste Klasse als Beobachtung ausgewertet, sondern auch die übrigen Klassen. Dafür werden den Klassen bei der Generierung der Klasseneigenschaften in dem Anlernschritt eine Winkel- und Betrags-Eigenschaft zugewiesen. Bei der Prototypenklassifikation werden diese Eigenschaften für jede Klasse mit dem Euklidischen Abstand gewichtet addiert, so dass der Sturz eine genauere Richtungsinformation bekommt. Durch die Gewichtung erhalten Prototypen mit einem größeren Abstand auch nur einen kleinen Anteil an der der Winkel- und Betragseigenschaft.

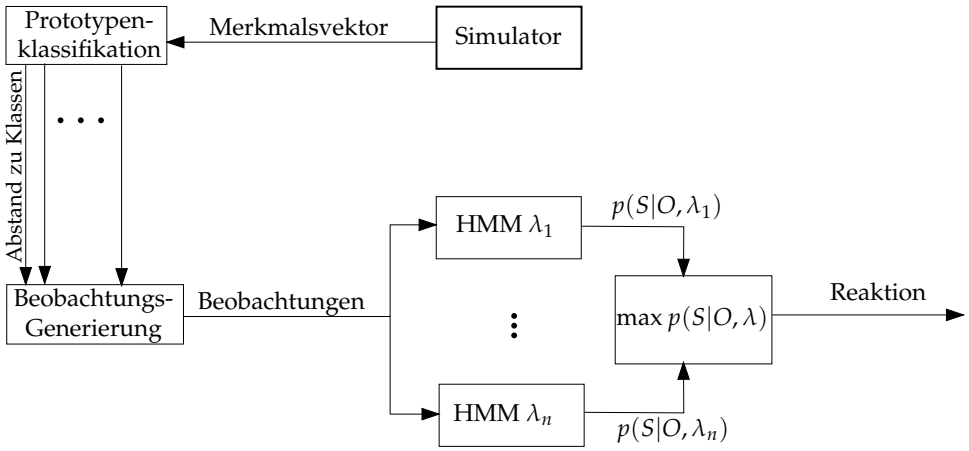


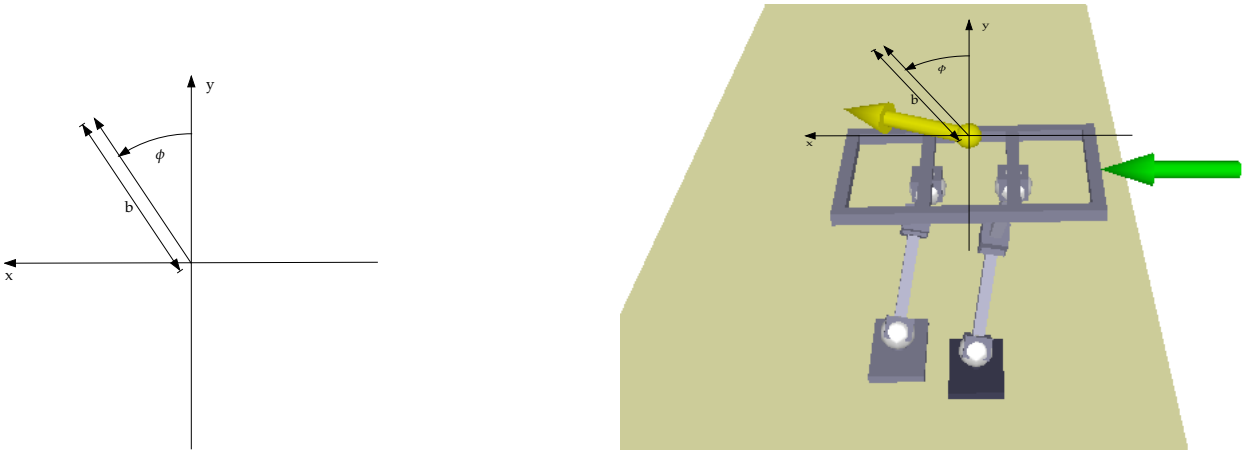
Abbildung 4.8: Blockdiagramm der $_{\text{r}}$HMMK

Die Koordinaten x und y (siehe Abbildung 4.9(b)) beschreiben eine Fallbewegung in einem zweidimensionalen Koordinatensystem, das auf den Roboter gelegt wird. r_k ist die Richtung des Fallens für die Klasse k und i_k die Intensität des Sturzes. d_2k ist der schon in der Prototypenklassifikation benutzte Euklische Abstand zwischen dem aktuellen Merkmalsvektor und dem Prototypen der Klasse k:

$$y = \sum_{k=1}^{K} \frac{i_k \cdot \cos(r_k)}{d_{2k}} \qquad x = \sum_{k=1}^{K} \frac{i_k \cdot \sin(r_k)}{d_{2k}}. \tag{4.27}$$

Die Winkel- und Betragsberechnung ergibt sich dann schnell:

$$\phi = \arctan\left(\frac{x}{y}\right) \qquad b = \sqrt{x^2 + y^2}. \tag{4.28}$$



(a) Winkel bei der $_{\mathrm{r}}$HMMK (b) Winkel bei der $_{\mathrm{r}}$HMMK mit Roboter

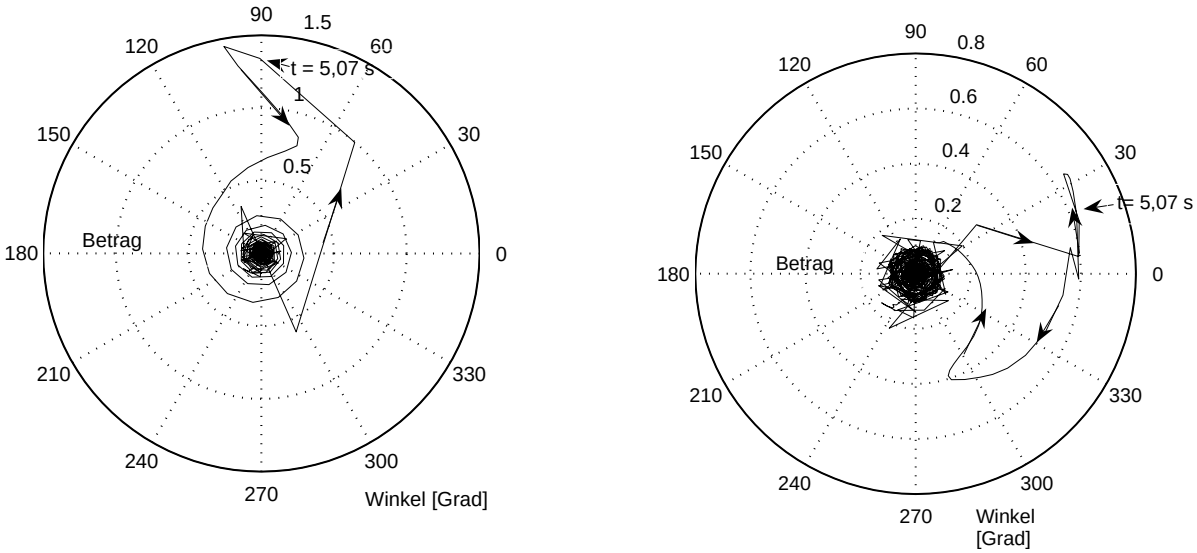
Abbildung 4.9: Darstellung der Berechnung des Winkels und des Betrages der $_{\mathrm{r}}$HMMK

Analog zu dem vorhergehenden HMM findet dieses Richtungs-HMM auch eine Abkürzung: $_{\mathbf{r}}\mathbf{HMMK}_2^{\mathbf{ws}}$ bedeutet, dass die Prototypenklassifikation mit dem Prototypen, der aus Merkmalsvektoren *während des Stoßes* (*ws*) angelernt worden ist, für das *Richtungs-HMM* mit *2 Zuständen* benutzt wird.

In Abbildung 4.10(a) und Abbildung 4.10(b) sind die Beträge und Winkel von zwei Simulationen mit verschiedenen Stoßrichtungen in Polarkoordinaten abgebildet. In beiden Abbildungen fällt der starke Ausschlag des Betrages bei dem Winkel für die jeweilige Fall-Richtung auf: In Abbildung 4.10(a) wird ein **Stoß über den Standfuß** bei $t = 5\,\mathrm{s}$ dargestellt. Nach dem Stoß wachsen sowohl der Winkel als auch der Betrag bis bei $t = 5{,}07\,\mathrm{s}$ der Betrag des Winkels $\phi = 90°$ am größten ist. Die seitliche Fallrichtung wird durch diesen Winkel also gut dargestellt und eine Beobachtung, die dieser Betrags-Winkel-Kombination entspricht, kann das Fallen gut beschreiben. Beim **Stoß von hinten** in Abbildung 4.10(b) ist der Winkel relativ konstant bei $\phi = 0°$ während der Betrag ansteigt. Bei $t = 5{,}07\,\mathrm{s}$ hat der Betrag auch hier sein Maximum erreicht und beschreibt das Fallen nach vorne gut. Auffällig ist, dass nach dem Fallen sowohl die Winkel als auch Beträge wieder sinken. Diese Tatsache begründet sich dadurch, dass während eines Stoßes die Abstände

der Prototypen aller Klassen sehr groß und die relativen Unterschiede geringer werden (siehe Abbildung 3.6(b)). Die Gewichtung mit dem Abstand hat somit keinen Einfluss mehr.

Die Beobachtungen werden nach den Kriterien erzeugt, wie hoch der Betrag und der Winkel ist.

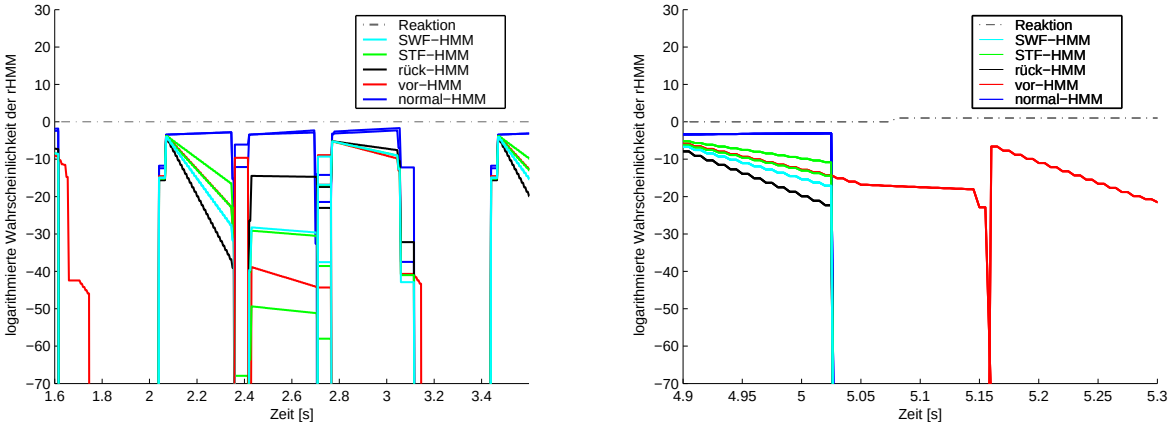


(a) Stoß über den Standfuß bei $t = 5{,}0\,\mathrm{s}$ ($t_p = -0{,}3$) mit einem Impuls $p = 18\,\mathrm{Ns}$

(b) Stoß von hinten bei $t = 5{,}0\,\mathrm{s}$ ($t_p = -0{,}3$) mit einem Impuls $p = 18\,\mathrm{Ns}$

Abbildung 4.10: Polardiagramme, die Winkel und Betrag der zusammengefassten Klassen zeigen, die der $_{\mathrm{r}}$HMMK als Beobachtungen dienen.

In Abbildung 4.11 werden die Wahrscheinlichkeiten der HMM der $_{\mathrm{r}}$HMMK während eines normalen Schrittes, nach einem Stoß und dem darauf folgenden Fallen dargestellt. Der Stoß wird bei $t = 5\,\mathrm{s}$ gegen den simulierten Roboter ausgeführt. Bis dahin ist ein normal-HMM der $_{\mathrm{r}}$HMMK das wahrscheinlichste HMM der $_{\mathrm{r}}$HMMK gewesen. Abbildung 4.11(b) zeigt, wie 30 ms nach dem Stoßbeginn die Wahrscheinlichkeiten aller HMM der $_{\mathrm{r}}$HMMK bis auf die vor-HMM der $_{\mathrm{r}}$HMMK rapide geringer werden. Ein vor-HMM der $_{\mathrm{r}}$HMMK ist nun das wahrscheinlichste und es könnte die entsprechende Reaktion ausgeführt werden.



(a) Wahrscheinlichkeiten der HMM der ${}_{\mathrm{r}}$HMMK während eines ungestörten Schrittes

(b) Wahrscheinlichkeiten der HMM der ${}_{\mathrm{r}}$HMMK nach einem Stoß von hinten bei 5.0 s ($t_p = -0{,}3$) mit einem Impuls $p = 18\,\mathrm{Ns}$

Abbildung 4.11: Die Wahrscheinlichkeiten der verschiedenen HMM der ${}_{\mathrm{r}}\mathrm{HMMK}_3^{\mathrm{ns}}$.

4.4 Anlernen der Hidden Markov Models

Das Anlernen kann manuell und automatisch durchgeführt werden. Beim manuellen Anlernen werden die gesammelten Beobachtungen nach jeder Simulation durch durch den Menschen einem zu trainierenden HMM zugewiesen. Das automatische Anlernen wird mit Programmen auf dem PC vorgenommen. Das manuelle Anlernen hat den Vorteil, dass viele Fallverläufe berücksichtigt werden können und auch spezielle Konstellationen erst durch Betrachten der Simulation erkenntlich werden. Nachteilig ist allerdings, dass mit dem manuellen Anlernen keine reproduzierbaren Ergebnisse möglich sind und verschiedene Mustererkennungsverfahren nicht verglichen werden können. Deshalb ist das automatische Anlernen auch wegen der Zeitersparnis zu bevorzugen.

Hierzu wird die Roboter-Simulation zweimal mit einem Benchmark-Skript mit den folgenden definierten Stoßzeitpunkten und Stoßrichtungen durchgeführt:

- Stoßrichtung $\in \{$Stoß von hinten Stoß von vorne Stoß über STF Stoß über SWF$\}$
- Stoßimpuls $p \in (12\,\mathrm{Ns} \quad 20\,\mathrm{Ns} \quad 28\,\mathrm{Ns})$
- Stoßzeitpunkt $t_p \in (-0{,}9 \quad -0{,}6 \quad -0{,}3 \quad 0 \quad +0{,}3 \quad +0{,}6)$.

In der einen Simulation wird keine Reaktion ausgeführt und in der anderen Simulation wird die gesteuerte Standardreaktion ausgeführt (siehe Abschnitt 2.4). Bei den Simulationen werden für die spätere Auswertung für jeden Stoß die Daten über die Stoßrichtung, die Stoßstärke, den Stoßzeitpunkt und die Information, ob der Roboter nach dem Stoß auf den Boden gefallen ist, in Dateien geschrieben. Außerdem werden bei beiden Simulationen die von der Simulation generierten Beobachtungen in zwei Dateien gespeichert. Mit diesen Beobachtungen werden die HMM angelernt. Da die beiden Simulationen bis auf die Reaktion identisch sind, können sie miteinander verglichen werden.

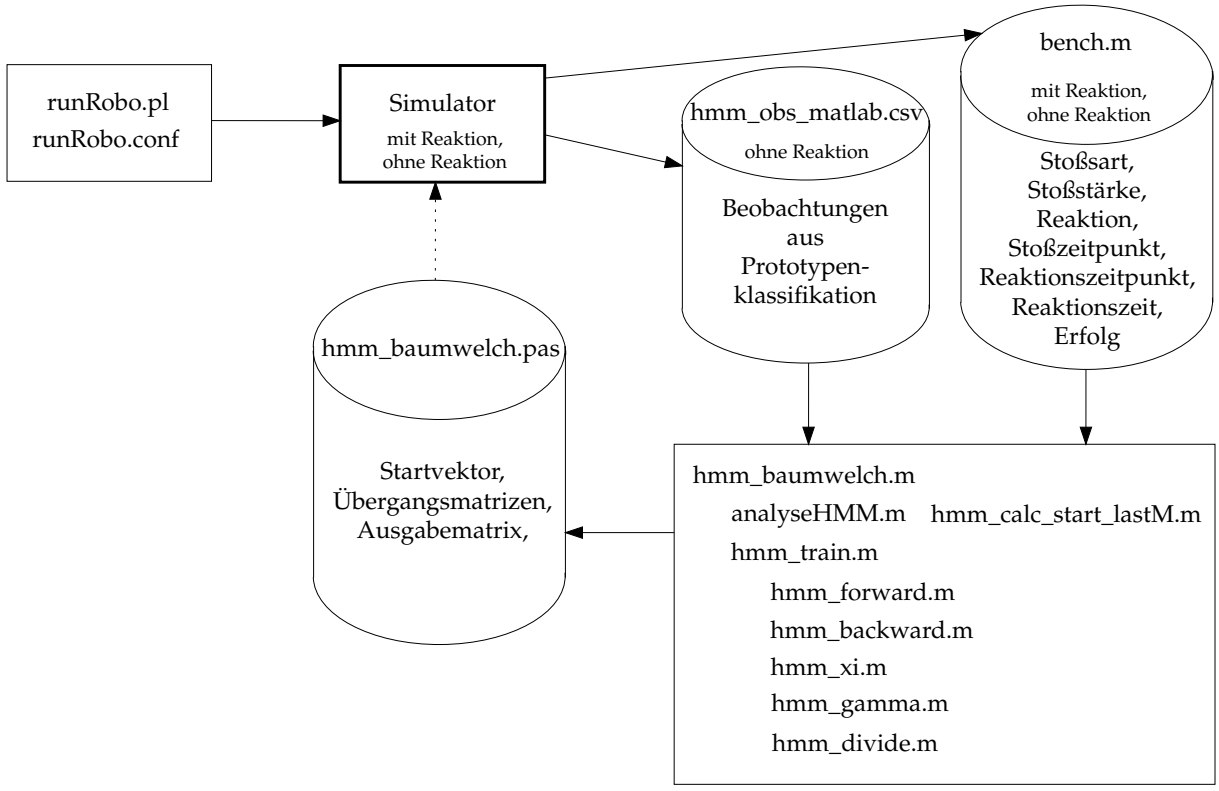


Abbildung 4.12: Übersicht über die Dateien, mit denen die HMM angelernt werden.

Ein MATLAB-Skript wertet diese Dateien aus und weist die Beobachtungen den HMM zu, wobei nur die Beobachtungen von der Simulation ohne Reaktion benutzt werden. Die Beobachtungen der anderen Simulation mit gesteuerter Standardreaktion werden nicht benutzt, weil sich die Beobachtungen während der Reaktion ändern. In der Mustererkennung mit den HMM sollen aber die Beobachtungen beim reinen Fallen die Grundlage sein.

Die lange Beobachtungssequenz jeder Simulation wird zunächst auf den interessanten

Bereich von 100 ms vor und 300 ms nach dem Stoß zusammengeschnitten. Der Stoßzeitpunkt muss nicht in den Beobachtungen erkannt werden, da in den Simulationen die Störzeitpunkte vorgegeben waren und auch in den Benchmark-Datei geschrieben wurden. Danach wird die restliche Beobachtungssequenz dem passenden HMM nach folgenden Kriterien zugewiesen:

- der Stoßzeitpunkt ist bekannt und wird entsprechend der Schrittphase dem frühen, mittleren oder späten HMM zugewiesen
- die Stoßrichtung ist bekannt
- die Zuweisung zu der Sturzintensität erfolgt nach der Tatsache, ob der Roboter nach der gesteuerten Reaktion den Sturz vermeiden konnte: Ist er hingefallen, wird die Beobachtungssequenz dem HMM mit der hohen Sturzstärke zugewiesen, bei der der Roboter eine Schadenvermeidungsreaktion wie die Hocke ausführen soll. War die Schrittausführung hingegen erfolgreich, wird die Beobachtungssequenz dem HMM mit der niedrigen Sturzstärke zugewiesen, an die die Standardreaktion gebunden wird.

Es werden die folgenden HMM erstellt:

- Sechs HMM beschreiben die normale ungestörte Gehenbewegung. Hier sind die Beobachtungen des normalen Gehens ohne Störung aufgenommen und in kleine gleichlange Teilsequenzen geschnitten, die unterschiedliche Zeiträume einer Schrittperiode beschreiben.
- Die HMM, die das Fallen beschreiben, werden mit den Beobachtungen der 24 Kombinationen aus den folgenden Parametern angelernt:
  - Fallrichtungen $\in \{$Fallen nach vorne Fallen nach hinten Fallen über STF Fallen über SWF$\}$
  - Sturzintensitäten $\in \{$OK Sturz$\}$
  - Schrittphasen $\in \{-\quad \bullet \quad +\}$

			Fallrichtungen			
			nach vorne	zurück	über STF	über SWF
Hidden Markov Model	$t_p \geq 0{,}3$	normal	+normal3, +normal3			
		OK	+vor_OK	+rück_OK	+STF_OK	+SWF_OK
		Sturz	+vor_Sturz	+rück_Sturz	+STF_Sturz	+SWF_Sturz
	$-0{,}6 < t_p < 0{,}3$	normal	• normal2, • normal2			
		OK	• vor_OK	• rück_OK	• STF_OK	• SWF_OK
		Sturz	• vor_Sturz	• rück_Sturz	• STF_Sturz	• SWF_Sturz
	$t_p \leq -0{,}6$	normal	-normal1, -normal1			
		OK	-vor_OK	-rück_OK	-STF_OK	-SWF_OK
		Sturz	-vor_Sturz	-rück_Sturz	-STF_Sturz	-SWF_Sturz

Tabelle 4.1: Die verschiedenen HMM mit der Unterscheidung von drei Schrittphasen.

Insgesamt werden für das Trainieren der Fallen-HMM in der Roboter-Simulation 72 Simulationen für 24 verschiedene HMM durchgeführt (siehe Tabelle 4.1).

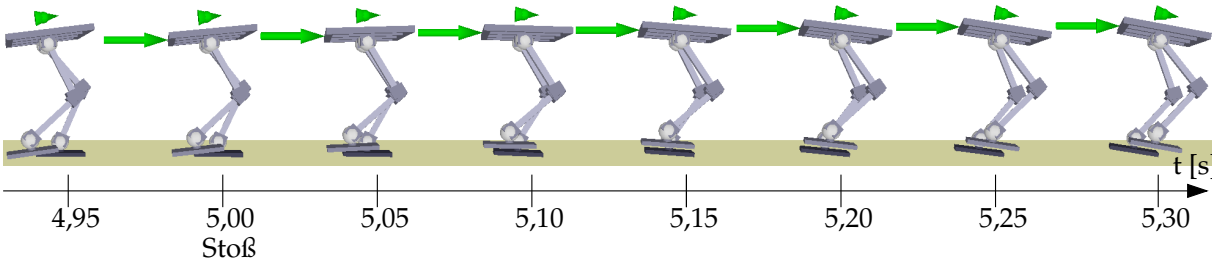
Dann werden die Startwerte von **A**, **B** und $\vec{\pi}$ für die HMM erstellt. Dabei wird für jedes HMM die gleiche Startmatrix erstellt, da in den kleinen Modellen immer die ungefähre Übergangsbewegung um die erste Diagonale beschrieben werden soll. Die Übergangsmatrizen haben die übliche quadratische Form, während die Ausgabewahrscheinlichkeitsmatrizen durch die große Anzahl der Beobachtungen mehr Spalten als Zeilen haben.

Nach diesen Vorbereitungen kann das eigentliche Trainieren der HMM durchgeführt werden. Es ist dabei die Anzahl der Trainingsschritte des Baum-Welch-Algorithmus zu wählen, mit der die eben trainierten Matrizen ein weiteres Mal trainiert werden. Es hat sich gezeigt, dass zehn Trainingsschritte in der Regel ausreichen und so das Trainieren innerhalb weniger Minuten abgeschlossen ist. Wenn alle Trainingsschritte durchgeführt worden sind, werden die Wahrscheinlichkeiten in **A**, **B** und $\vec{\pi}$ logarithmiert, damit die einfachere Berechnung des Viterbi-Pfades in der Simulation angewandt werden kann (siehe Unterabschnitt 4.2.3). Da einige Wahrscheinlichkeiten den Wert $p = 0$ haben, kann $p_{log} = \log(0)$ nicht berechnet werden, weil dieser Wert nicht definiert ist. Deshalb werden für $p_{log} = \log(0)$ sogenannte Ersatzwahrscheinlichkeiten benutzt:

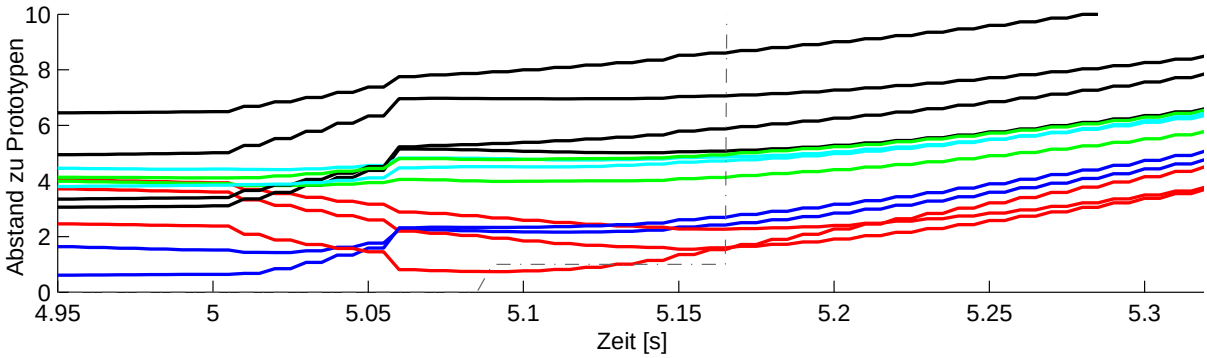
- In dem Startwahrscheinlichkeitsvektor $\vec{\pi}$ wird $\log(0) \equiv -2000$ festgelegt. Dieser Wert ist bewusst sehr niedrig – also unwahrscheinlich – gewählt worden.

- In der Übergangsmatrix **A** wird $\log(0) \equiv -400$ festgelegt. Dieser Wert ist höher festgelegt worden, damit untrainierte Zustandswechsel überhaupt möglich sind. Allerdings sollen diejenigen HMM, die solch einen Zustandwechsel von sich aus beschreiben, auch nicht künstlich übertroffen werden.
- In der Ausgabewahrscheinlichkeitsmatrix **B** wird $\log(0) \equiv -200$ festgelegt. Dieser Wert ist mit Absicht die höchste Ersatzwahrscheinlichkeit. Denn es kommt häufiger vor, dass eine Beobachtung generiert wird, die in dem HMM nicht vorgesehen ist.

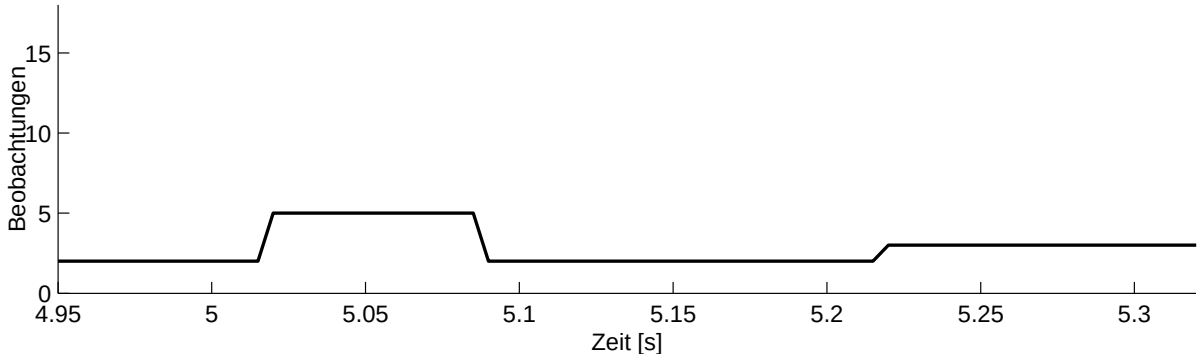
Danach werden die trainierten Matrizen in eine Pascal-Datei geschrieben, die von der Roboter-Simulation eingebunden wird. Nun kann der Simulator gestartet werden und lädt diese angelernten Parameter beim Initialisieren.



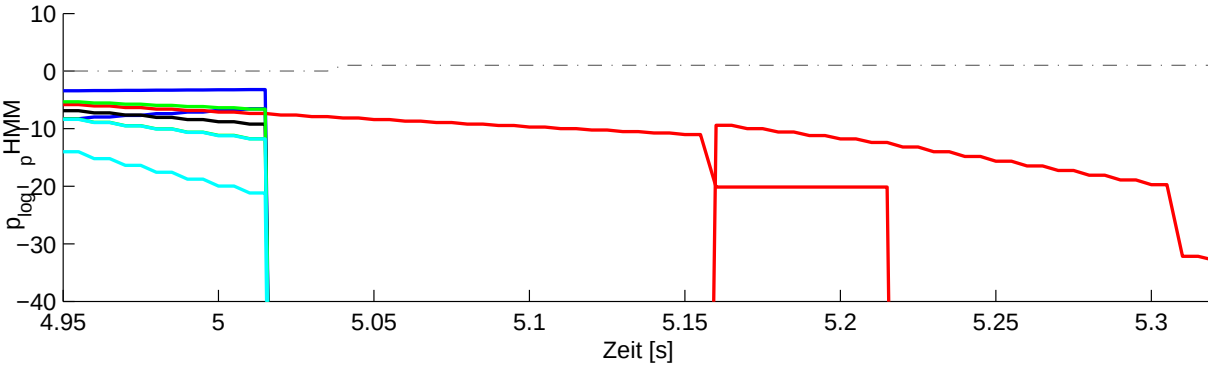
(a) Bildsequenz des modellierten Roboters



(b) Abstände der Prototypen der Klassen zu dem aktuellen Merkmalsvektor

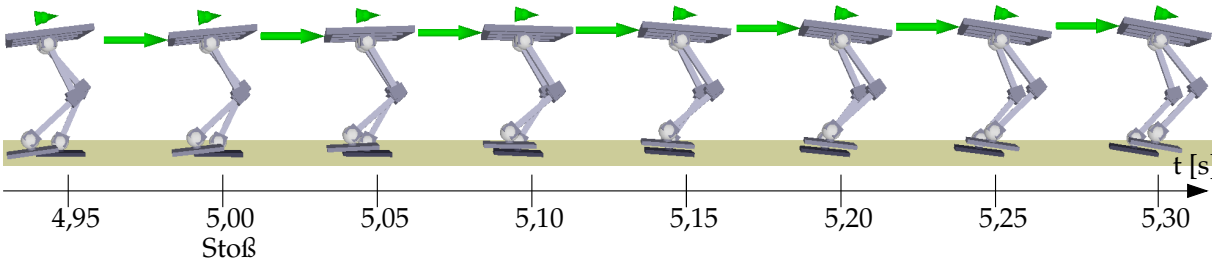


(c) aus den Prototypen generierte Beobachtungen für die $_{\mathrm{p}}$HMMK

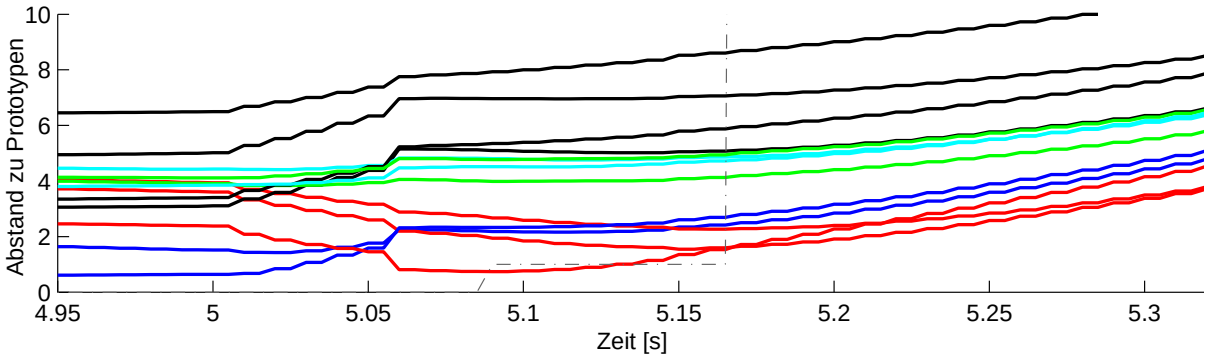


(d) logarithmierte Wahrscheinlichkeiten der verschiedenen HMM der $_{\mathrm{p}}$HMMK

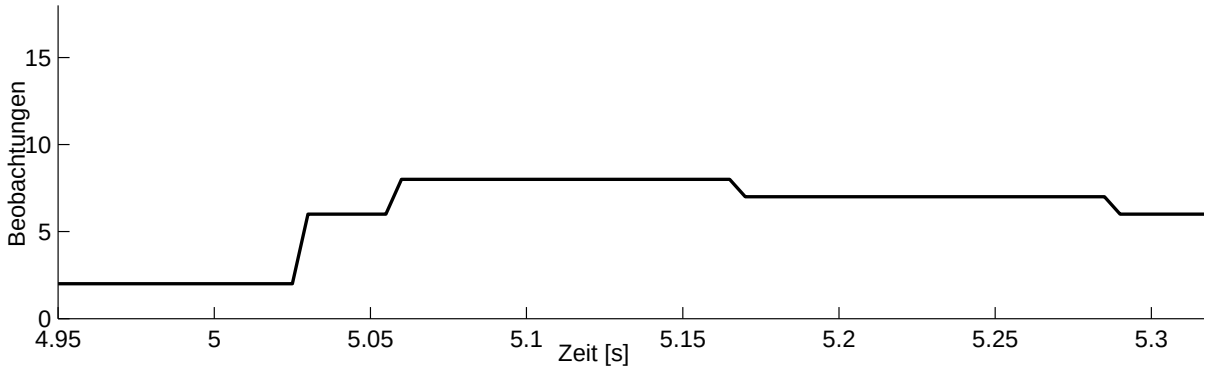
Abbildung 4.13: Bildsequenz und Verläufe von Graphen der HMM der $_{\mathrm{p}}$HMMK nach einem Stoß von hinten bei 5.0 s ($t_p = -0{,}3$) mit einem Impuls $p = 18$ Ns. Die Farben haben folgenden Klassen-Bedeutung: blau: normal, rot: Fallen nach vorn, schwarz: Fallen nach hinten, grün: Fallen über SWF und cyan: Fallen über STF.



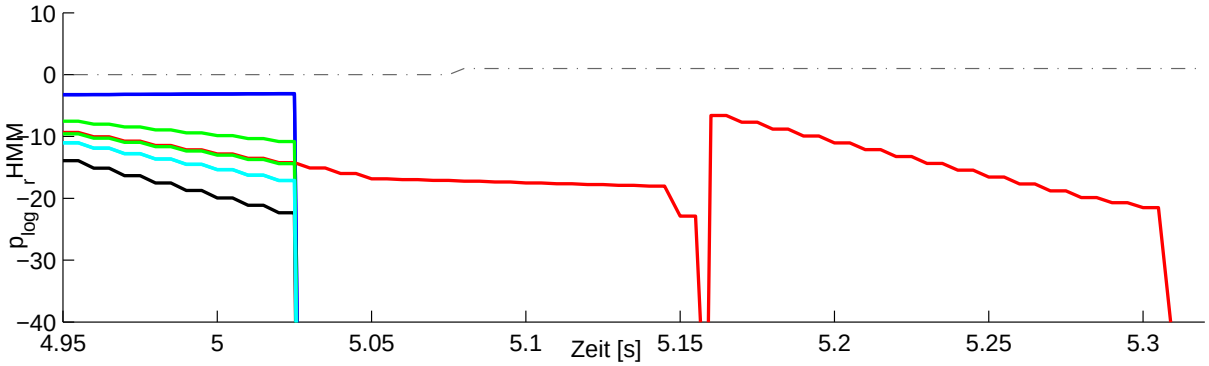
(a) Bildsequenz des modellierten Roboters



(b) Abstände der Prototypen der Klassen zu dem aktuellen Merkmalsvektor



(c) aus den Prototypen generierte Beobachtungen der $_{\mathrm{r}}$HMMK



(d) logarithmierte Wahrscheinlichkeiten der verschiedenen HMM der $_{\mathrm{r}}$HMMK

Abbildung 4.14: Bildsequenz und Verläufe von Graphen der HMM der $_{\mathrm{r}}$HMMK nach einem Stoß von hinten bei 5.0 s ($t_p = -0{,}3$) mit einem Impuls $p = 18$ Ns. Die Farben haben folgenden Klassen-Bedeutung: blau: normal, rot: Fallen nach vorn, schwarz: Fallen nach hinten, grün: Fallen über SWF und cyan: Fallen über STF.

5 Stabilitätsaussage anhand der Fuß-Kippbewegung

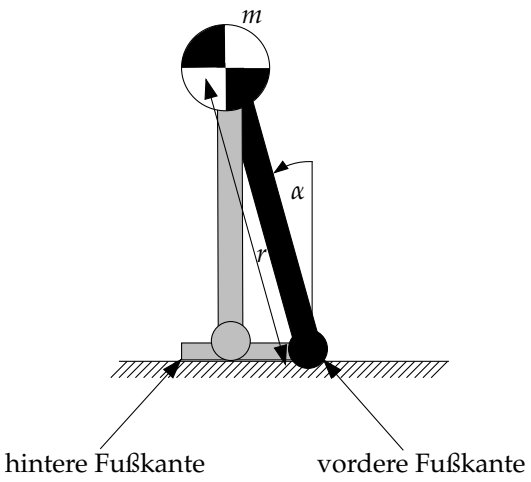
Während die beiden bisher vorgestellten Mustererkennungsverfahren mit Informationen aus vielen verschiedenen Sensoren angelernt werden, wird in diesem Kapitel ein Verfahren zur Stabilitätsuntersuchung mit einem analytischen Ausdruck erläutert. Dabei wird der Fußwinkel des Standfußes mit seiner Fußwinkelgeschwindigkeit in Beziehung gesetzt.

5.1 Grundlagen

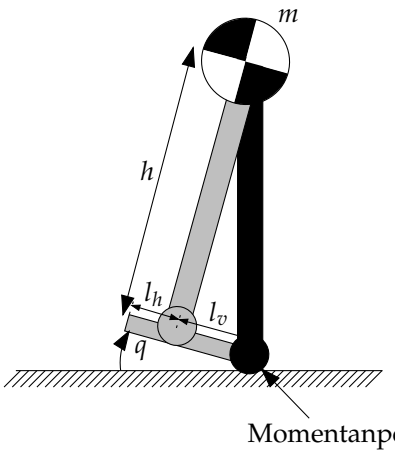
Der Roboter wird vereinfacht als inverses Pendel mit einer Punktmasse dargestellt. Bei einem inversen Pendel wird die Pendelmasse – die Masse des Robotertorsos – oberhalb des Pendelstabes angenommen. Dieses inverse Pendel hat nur einen rotatorischen Freiheitsgrad um den Winkel q. Dabei rotiert das Pendel um einen Momentanpol, der durch eine Fußkante des Roboters gegeben ist. In Abbildung 5.1 ist das inverse Pendel für die vordere Fußkante abgebildet. In der Ruhelage des Roboters (Abbildung 5.1(a)) hat der Winkel α den aus der Höhe h und der Länge des vorderen Fußteils l_v bestimmten Wert

$$\alpha = \arctan\left(\frac{l_v}{h}\right) \approx 11^\circ. \tag{5.1}$$

In Abbildung 5.1(b) ist der Roboter genau um diesen Winkel $q = \alpha$ gekippt. Wenn die Fußwinkelgeschwindigkeit $\dot{q} \leq 0$ ist, kippt der Roboter nicht über seine vordere Fußkante, sondern kippt in seine stabile Lage zurück. Nun ist ersichtlich, dass bei manchen Kombinationen von q und $\dot{q}$ der Roboter bzw. das inverse Pendel auf den Boden fällt: Wenn zum Beispiel $q \geq \alpha$ und $\dot{q} > 0$ ist, wird es nach vorne kippen und hinfallen.



(a) Ruhelage des Roboters



(b) Der Roboter ist nach vorne gekippt

Abbildung 5.1: Der Roboter ist als inverses Pendel vereinfacht. Er kippt über die vordere Fußkante des Standfußes.

Im Folgenden soll eine funktionelle Beziehung $\dot{q}(q)$ gesucht werden. Dafür ist eine Betrachtung der Energien sinnvoll [HSG04]. Die Energie des Pendels ist

$$E_{Pendel} = \frac{1}{2} J \dot{q}^2 + rmg \cos(q - \alpha) \tag{5.2}$$

und die maximale Energie, bei der das Pendel nicht nach vorne kippt, ist

$$E_{max} = rmg. \tag{5.3}$$

Durch Gleichsetzen von $E_{Pendel} = E_{max}$, Umstellen und Einsetzen des Massenträgheitsmomentes für die Punktmasse $J = mr^2$ ergibt sich die Beziehung

$$\dot{q}_{max}(q) = \sqrt{\frac{2g}{r}[1 - \cos(q - \alpha)]}. \tag{5.4}$$

Mit dieser Beziehung lässt sich eine Stabilitätsaussage machen: Ist $\dot{q}(q) < \dot{q}_{max}(q)$, so ist das System stabil.

In der Laufbewegung ändert sich die Schwerpunktsposition durch die verschiedenen Beinkonfigurationen. Deshalb ergibt sich die Zeitabhängigkeit des Trägheitsmomentes

$J(t)$, der Länge der Pendelstange $r(t)$ und des Winkels $\alpha(t)$. Somit muss die Berechnung der Gleichung 5.4 fortwährend durchgeführt werden.

5.2 Anwendung

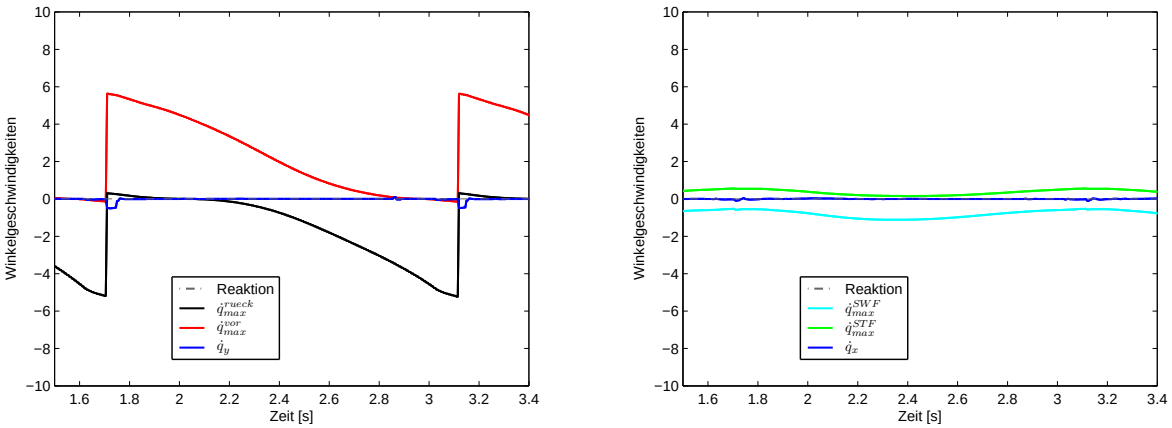
In der Roboter-Simulation wird die Berechnung der maximalen Kippwinkelgeschwindigkeit $\dot{q}_{max}(q)$ alle 10 ms durchgeführt.

In Abbildung 5.2 sind die Verläufe der maximalen Kippwinkelgeschwindigkeiten über alle vier Fußkanten abgebildet. In Abbildung 5.2(a) sind die maximalen Kippwinkelgeschwindigkeiten $\dot{q}_{max}^{vor}$ und $\dot{q}_{max}^{rueck}$ und die aktuelle Kippwinkelgeschwindigkeit in y-Richtung während eines normalen ungestörten Schrittes aufgetragen. Das Stabilitätsmaß lässt sich durch $\dot{q}_{max}^{rueck} < \dot{q}_y < \dot{q}_{max}^{vor}$ ausdrücken. Es fällt auf, dass $\dot{q}_{max}^{vor}$ während des Schrittes monoton sinkt, während $|\dot{q}_{max}^{rueck}|$ monoton steigt. Die Begründung hierfür ist die Verlagerung des Schwerpunktes während eines Schrittes. Beim Beginn des Schrittes ist der Schwingfuß hinten unter dem Torso; auch der Schwerpunkt liegt hier. In dieser Beinkonfiguration kann erst eine größere Drehrate $\dot{q}_y$ den Roboter nach vorne destabilisieren, während nach hinten bereits eine kleines $\dot{q}_y$ für eine Destabilisierung reicht. Bei $t = 1{,}75\,\text{s}$ müsste der Roboter eigentlich nach hinten fallen. Diese Schwerpunktbetrachtungen sind schon in ähnlicher Weise in Abschnitt 2.5 angestellt worden. Beim Vergleich der Bilder zur Stabilitätsanalyse in Abbildung 2.4(a) und Abbildung 5.2 zeigt sich eine Analogie:

- Die Eigenstabilität nach einem Stoß von hinten sinkt während eines Schrittes (Abbildung 2.4(a)) genauso wie die maximale zulässige Winkelgeschwindigkeit $\dot{q}_{max}^{vor}$ in Abbildung 5.2[1]. In späten Schrittphasen $t_p \geq 0{,}6$ ist die Eigenstabilität im Gegensatz zur $\dot{q}_{max}^{vor}$ größer; dies liegt an der stabilisierenden Wirkung des Schwingfußes, der kurz vor dem Aufsetzten vor dem Torso ist. Diese Tatsache ist aber in der Berechnung von $\dot{q}_{max}$ nicht berücksichtigt und deshalb ergibt sich der Unterschied.
- Beim Stoß von vorne steigt die Eigenstabilität in Abbildung 2.4(a) mit Fortschreiten der Schrittphase ebenso monoton an wie $|\dot{q}_{max}^{rueck}|$ in Abbildung 5.2(a) steigt.

1 In Abbildung 5.2 ist auf der Abszisse zwar die Zeit in Sekunden abgetragen, jedoch ist eine ganze Schrittperiode abgebildet. Deshalb ist ein Vergleich möglich.

In Abbildung 5.2(b) sind die aktuelle Kippwinkelgeschwindigkeit in die seilliche Richtung und die maximal erlaubten Kippwinkelgeschwindigkeiten über den Standfuß $\dot{q}_{max}^{STF}$ und über den Schwingfuß $\dot{q}_{max}^{SWF}$ abgebildet. Diese maximalen Kippgeschwindigkeiten in x-Richtung sind geringer als die in vorwärts-Richtung. Bei $t = 4\,\mathrm{s}$ hat $\dot{q}_{max}^{STF}$ ein lokales Minimum während $\dot{q}_{max}^{SWF}$ ein lokales Maximum hat. Auch dieses Verhalten ist in (Abbildung 2.4(a)) zur Stabilitätsanalyse zu finden , bei der der simulierte Roboter beim Stoß über den STF in mittleren Schrittphasen mehr Eigenstabilität besitzt. Das lässt sich mit der Position des Schwerpunktes zum Zeitpunkt $t_p \approx 0$ anhand Abbildung 2.3(a) erklären. Zu diesem Zeitpunkt ist der Roboter relativ stark über den Standfuß gelehnt, so dass nur geringe Impulse beziehungsweise Fußkippwinkelgeschwindigkeiten für ein Kippen und somit Fallen über die seitliche Fußkante genügen.



(a) Fußwinkelgeschwindigkeiten in vor-Richtung

(b) Fußwinkelgeschwindigkeiten in seitlicher Richtung

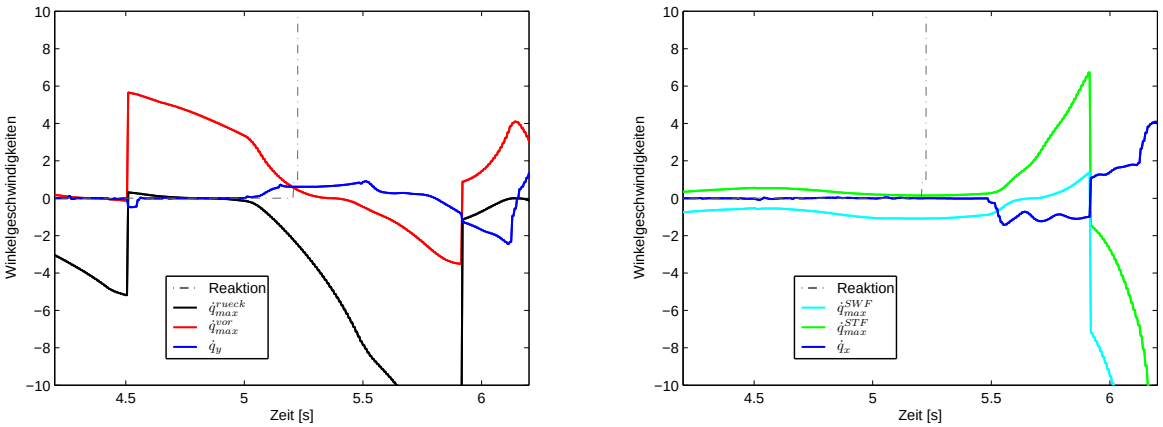
Abbildung 5.2: Winkelgeschwindigkeit des Standfußes während eines normalen Schrittes. Die maximalen Kippwinkelgeschwindigkeiten in alle vier Richtungen sind als Stabilitätsgrenzen anzusehen.

Bisher ist nur ein normaler Schritt betrachtet worden. In Abbildung 5.3 sind die Verläufe der Fußwinkelkippgeschwindigkeiten nach einem Stoß von hinten bei $t = 5\,\mathrm{s}$ zu finden. Aus Abbildung 5.3(a) ist ersichtlich, wie nach dem Stoß $\dot{q}_y$ durch die Kippbewegung des Roboters nach vorne ansteigt und gleichzeitig $\dot{q}_{max}^{vor}$ sinkt. Bei $t = 5{,}2\,\mathrm{s}$ ist $\dot{q}_y > \dot{q}_{max}^{vor}$ und somit die Stabilitätsgrenze überschritten. Zu diesem Zeitpunkt würde der Roboter die Standardreaktion – einen Schritt nach vorne – ausführen. Im späteren Verlauf, wenn der Abstand zwischen $\dot{q}_y$ und $\dot{q}_{max}^{vor}$ in der Kippbewegung größer wird, würde die Schutzreakti-

on ausgeführt werden. Abbildung 5.3(b) zeigt, dass sich nach dem Stoß die Verläufe der Kippwinkelgeschwindigkeiten in x-Richtung bis 500 ms nicht ändern, sondern erst, wenn der Roboter schon im späten Fallverlauf ist.Daraus folgt, dass für diese Fallrichtungen auch nur die Fußkippwinkelgeschwindigkeiten in y-Richtung aussagekräftig sind. Für seitliche Kippbewegungen sind dementsprechend die maximalen seitlichen Kippwinkelgeschwindigkeiten relevant.

Die Klassifizierung mit dieser Methode wird im Folgenden **FKbK**[2] abgekürzt.

Die FKbK lässt den Roboter eine Reaktion nach dem Kriterium ausführen, welche der vier maximalen Kippwinkelgeschwindigkeiten von der aktuellen Kippwinkelgeschwindigkeit überschritten wird. Bei einer geringen Differenz $|\dot{q} - \dot{q}_{max}|$ wird die Standardreaktion ausgeführt, zum Beispiel bei $\dot{q}_y < \dot{q}_{max}^{rueck}$ der Schritt nach hinten. Vergrößert sich die Differenz $|\dot{q} - \dot{q}_{max}|$ wird die Schutzreaktion Hocke ausgeführt.



(a) Fußwinkelgeschwindigkeiten in vor-Richtung

(b) Fußwinkelgeschwindigkeiten in seitlicher Richtung

Abbildung 5.3: Winkelgeschwindigkeit des Standfußes während eines Stoßes von hinten bei 5.0 s ($t_p = -0{,}3$) mit einem Impuls $p = 18$ Ns. Die maximalen Kippwinkelgeschwindigkeiten in alle vier Richtungen sind als Stabilitätsgrenzen abgebildet.

2 **F**uß-**K**ipp**b**ewegung-**K**lassifizierung.

6 Vergleich der Mustererkennungsverfahren

In den vorhergehenden Kapiteln sind die Verfahren zur Mustererkennung vorgestellt worden. In diesem Kapitel erfolgt zunächst eine Erläuterung von Bewertungskriterien und im Anschluss die Bewertung der einzelnen Verfahren hinsichtlich ihrer Brauchbarkeit zur Sturzdetektion und Klassifikation.

6.1 Benchmark-Umgebung

Für jedes Mustererkennungsverfahren wird der Roboter-Simulator mit dem `runRobo.pl` -Skript gesteuert, so dass alle Kombinationen der folgenden Parameter simuliert werden[1]:

- Stoßrichtung $\in \{$Stoß von hinten Stoß von vorne Stoß über STF Stoß über SWF$\}$
- Stoßimpuls $\in \{4\,\text{Ns} \quad 8\,\text{Ns} \quad 12\,\text{Ns} \quad 16\,\text{Ns} \quad 20\,\text{Ns} \quad 24\,\text{Ns} \quad 28\,\text{Ns} \quad 36\,\text{Ns}\}$
- Stoßzeitpunkt $t_{p_s} \in \{-0{,}8 \quad -0{,}6 \quad -0{,}4 \quad -0{,}2 \quad 0 \quad +0{,}2 \quad +0{,}4 \quad +0{,}6 \quad +0\,{,}8\}$

Dabei beginnt der simulierte Roboter für jede Kombination zu laufen und wird zu den Stoßzeitpunkten gestoßen. Dann führt der Roboter nach Maßgabe des zu untersuchenden Mustererkennungsverfahrens eine Reaktion aus, entweder eine der vier Standardreaktionen (Tabelle 2.1), die Schutzreaktion oder keine Reaktion. Nach einem Stoß kann es mehrere Sekunden dauern bis der Roboter einen Grenzzyklus an der Stabilitätsgrenze zwischen Fallen und nicht-Fallen verlassen hat und wirklich hinfällt. Deshalb wird die

1 Die Gesamtsimulationsdauer für alle diese 288 Kombinationen beträgt ungefähr eine Stunde auf einem Pentium IV 3 GHz.

Simulation noch 7 s nach dem Stoß weitergeführt. Die Ergebnisse wie Stoßimpuls, Stoßrichtung, Reaktion, Reaktionsdauer und die Information, ob der Roboter gefallen ist, werden in einer Textdatei gespeichert. Nach dem Ende aller Einzelsimulationen liest ein MATLAB-Skript diese Textdatei ein und wertet sie aus. Dabei bereitet es die Daten auf und berechnet die Vergleichskriterien (siehe Abschnitt 6.2). Hierbei werden die in diesem Kapitels zu sehenden Abbildungen erstellt.

6.2 Vergleichskriterien

Die Stabilitätsanalysen (Abschnitt 2.5) verdeutlichten, wie die Stabilität des Roboters sich nach den Stößen je nach Stoßzeitpunkt, Stoßstärke und Stoßrichtung verhält. Es sind zwei verschiedene Simulationen durchgeführt worden:

SmgR: In der *Simulation mit gesteuerter Reaktion* führt der Roboter 120 ms nach dem Beginn des Stoßes die jeweilige Standardreaktion (Tabelle 2.1) aus. Beim einem Stoß von vorne führt der Roboter also – unabhängig von Stoßimpuls und Stoßzeitpunkt – mit dem Schwingfuß immer einen Schritt nach hinten aus. Die Dauer des Stoßes beträgt 60 ms, also wird durch diese SmgR aufgezeigt, wie sich der Roboter verhält, wenn er die Standardreaktion 60 ms nach dem Ende des Stoßes ausführt.

SoR: Der Roboter führt in der *Simulation ohne Reaktion* niemals eine Reaktion aus. An dieser ist erkennbar, dass bei manchen Parametern eine Reaktion gar nicht nötig ist, weil der Roboter eine ausreichende Eigenstabilität besitzt.

Mit diesen beiden Simulationen wird die Simulation mit dem jeweiligen Mustererkennungsverfahren vergleichen:

SmR: In der *Simulation mit Reaktion* legt das jeweilige Mustererkennungsverfahren online fest, wann der Roboter welche Reaktion ausführen soll. Dabei kann die Reaktion auch gar keine Reaktion seien, da zum Beispiel geringe Stoßimpulse den Roboter selten zu Fall bringen. Die SmR muss sich dabei an der SmgR vergleichen lassen und möglichst besser sein, indem es zum Beispiel die Reaktion früher ausführen lässt. Im Vergleich mit der SoR darf die SmR auf keinen Fall schlechter sein, denn das würde bedeuten, dass das Mustererkennungsverfahren die Stabilität des Roboters erst verschlechtert hat.

Für die folgenden Untersuchungen werden noch zusätzliche Abkürzungen eingeführt:

- γ bedeutet, dass der Roboter am Ende der Simulation ***g**efallen* ist.
- $\bar{\gamma}$ bedeutet, dass der Roboter zum Ende der Simulation ***n**icht **g**efallen* ist.

Für den Vergleich der SmR mit den SmgR und SoR wird eine neue Darstellungsweise der Stabilitätsanalyse eingeführt. In Abbildung 6.1 ist solch eine Abbildung zu finden. Es werden wie in der normalen Stabilitätsanalyse die verschiedenen Stoßrichtungen, Stoßimpuls und Stoßzeiten unterschieden. Den acht Kombinationen, die sich aus den drei Simulationenen SmgR, SoR und SmR mit jeweils zwei Möglichkeiten, ob der Roboter gefallen ist oder nicht, werden Farben zugewiesen. Dabei bedeuten die Farben *schwarz*, *hellgrün*, *dunkelgrün* und *blau* in dieser Reihenfolge eine Verbesserung der SmR (Abbildung 6.1(b)):

$$\text{Farb}_{\text{kalt}} \in \{\underbrace{\text{schwarz} \quad \text{hellgrün} \quad \text{dunkelgrün} \quad \text{blau}}_{\overrightarrow{\text{Verbesserung}}}\}.$$

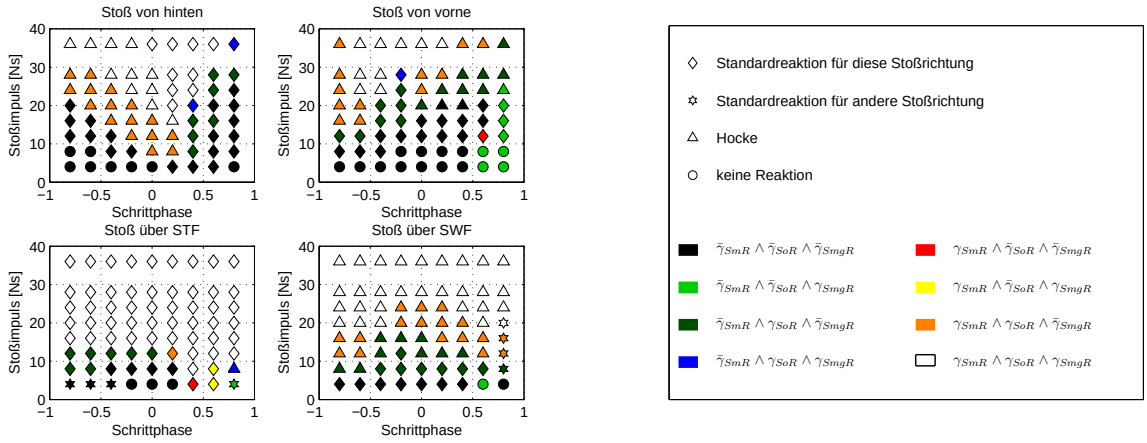
Mit den Farben *weiß*, *orange*, *gelb* und *rot* wird in dieser Reihenfolge eine Verschlechterung der Mustererkennungsverfahrens aufgezeigt[2]:

$$\text{Farb}_{\text{warm}} \in \{\underbrace{\text{weiß} \quad \text{orange} \quad \text{gelb} \quad \text{rot}}_{\overrightarrow{\text{Verschlechterung}}}\}.$$

Diese Farben sollen dem Betrachter dieser durchaus komplexen Abbildungen zuerst auffallen und anhand der Farben soll er mit einem Blick sehen können, wo das jeweilige Mustererkennungsverfahren sinnvollere Reaktionen bestimmt hat. Die Helligkeitsunterschiede *weiß* und *schwarz* sind dabei als neutral anzusehen und die Farben als Bewertungen. Finden sich mehr kalte Farben (grün und blau) ist das Mustererkennungsverfahren besser, finden sich mehr warme Farben (gelb, orange und rot) ist das Mustererkennungsverfahren schlechter im Vergleich zur SoR und SmgR.

Eine weitere Information wird über die Formen der Symbole ausgedrückt. Diese stellen die Reaktion dar, die das jeweilige Mustererkennungsverfahren ausgewählt hat. Dabei

2 Dabei ist nicht festgelegt, ob $\gamma_{SmR} \wedge \gamma_{SoR} \wedge \bar{\gamma}_{SmgR}$ (orange Farbe) oder $\gamma_{SmR} \wedge \bar{\gamma}_{SoR} \wedge \gamma_{SmgR}$ (gelbe Farbe) besser ist.



(a) Stabilitätsanalsyse PK^{WS} (b) Legende zur Stabilitätsanalsyse

Abbildung 6.1: Eine Beispielabbildung zur erweiterten Stabilitätsanalyse

wird *keine Reaktion* durch einen Kreis dargestellt, die *Standardreaktion für die Stoßstärke* hat die Raute und die *Schutzreaktion Hocke* wird durch ein Dreieck symbolisiert. Die Form des Pentagramms steht für die *Standardreaktion für eine andere Stoßrichtung*, die bei der jeweiligen Stoßrichtung ungewöhnlich ist und nicht zu finden sein sollte. Die Reihenfolge dieser Symbole sollte in einer Abbildung mit steigendem Stoßimpulsen Kreis, Raute, Dreieck sein.

Da der Vergleich mit diesen Abbildungen zwar viel Informationen bereitstellt, aber durch den Betrachter vorzunehmen ist, sind zusätzlich Vergleichgrößen entwickelt worden, die durch Gütefunktionen die Bewertung der Mustererkennungsverfahren mit skalaren Werten ermöglichen:

- Die **Sturzvermeidungsgröße** σ bewertet, wie viele Stürze durch die von dem Mustererkennungsverfahren ausgelöste Reaktion vermeiden werden können. Der Wertebereich ist mit $\sigma \in \{0 \dots 10\}$ festgesetzt, wobei 0 die bestmögliche Bewertung und 10 die schlechtmöglichste Bewertung sind.

$$\sigma = \begin{cases} 0 & \text{für } \bar{\gamma}_{SmR} \\ f_\sigma(\text{Stoßimpuls, Stoßrichtung, Reaktion}) \in \{0 \dots 9\} & \text{für } \gamma_{SmR} \end{cases} \tag{6.1}$$

Fällt der Roboter bei der SmR nicht, so wird die beste Bewertung $\sigma = 0$ erreicht. Fällt er hingegen, erhält σ einen Wert in Abhängigkeit von Stoßrichtung, Stoßimpuls und Reaktion: Ist keine Reaktion ausgeführt worden, wird σ je nach Stoßstärke sehr stark erhöht. Hat er die Standardreaktion (siehe Tabelle 2.1) ausgeführt, fällt σ milder aus. Hat der Roboter aber eine andere Reaktion ausgeführt, wird das Mustererkennungsverfahren mit einem größeren σ bestraft.

- Die **Robustheitsgröße ρ** drückt aus, dass nach leichten Stößen eine Reaktion nicht immer notwendig ist. Wenn nämlich der Roboter bei jeder noch so geringen Störung Reaktionen ausführt, hat er keine Zeit mehr, seine eigentlichen Aufgaben zu erfüllen. Der Roboter muss also auch robust gegenüber Störungen sein, die ihn nicht zu Fall bringen. Die Robustheitsgröße ρ hat den gleichen Wertebereich wie σ.

$$\rho = \begin{cases} 0 & \text{für } \bar{\gamma}_{SmR}, \bar{\gamma}_{SoR}, \text{Reaktion} = 0 \\ 0 & \text{für } \bar{\gamma}_{SmR}, \gamma_{SoR} \\ 0 & \text{für} \gamma_{SmR}, \gamma_{SoR} \\ f_\rho(\text{Stoßimpuls, Stoßrichtung, Reaktion}) \in \{0 \dots 7\} & \text{für } \bar{\gamma}_{SmR}, \bar{\gamma}_{SoR}, \text{Reaktion} \neq 0 \\ 10 & \text{für } \gamma_{SmR}, \bar{\gamma}_{SoR} \end{cases} \tag{6.2}$$

Steuert das Mustererkennungsverfahren die Reaktionen nur, wenn der Roboter auch wirklich fällt, erhält es eine gute Bewertung, also ein kleines ρ. Eine sehr gute Bewertung von $\rho = 0$ wird bei folgenden Kombinationen vergeben:

  - Fällt der Roboter sowohl bei der SoR als auch der SmR nicht, dann ist eine Reaktion nicht notwendig. Das Mustererkennungsverfahren hat den Stoß richtigerweise als nicht gefährlich eingestuft.
  - Fällt er hingegen nur bei der SoR, war die Reaktion notwendig und erfolgreich.
  - Fällt er bei beiden Simulationen, kann über die Robustheit keine Aussage gemacht werden.

Wenn der Roboter sowohl mit gesteuerter als auch ohne Reaktion nicht gefallen ist, aber trotzdem eine Reaktion ausgeführt hat, wird ein Wert zwischen 0 und 7 vergeben. Dieser Wert wird bei gleicher Stoßrichtung mit wachsendem Stoßimpuls geringer,

weil der Roboter bei starken Stößen reagieren soll, bei schwachen, die ihn nicht zu Fall bringen, hingegen nicht.

Der Höchstwert von $\rho = 10$ wird vergeben, wenn $\bar{\gamma}_{SmR}$ und γ_{SoR} gelten. Dann ist der Roboter erst durch die vom Mustererkennungsverfahren veranlasste Reaktion gefallen, was genau im Gegensatz zum Ziel der Sturzvermeidung durch Reflexreaktionen steht.

Die Reaktionen werden in ρ zusammenfasst. Deshalb ist ein Blick auf die unbewerteten Reaktionen durchaus sinnvoll. *Keine Reaktion* sollte bei Stößen mit geringen Impulsen ausgeführt werden. Beim manuellen Vergleichen mit den Abbildungen der Stabilitätsanalyse kann verglichen werden, ob der Roboter wirklich nicht gefallen ist. Die *Standardreaktion Reflexschritt* sollte bei mittleren Stößen ausgeführt werden, um den Sturz zu vermeiden. Bei stärkeren Stoßimpulsen sollte die *Reaktion Hocke* ausgeführt werden, wenn das Fallen nicht mehr vermeidbar ist. Somit sollte der Anteil der *Stürze ohne Hocke* möglichst gering sein. Sind bei gleichem Stoßimpuls sowohl dieser Anteil und der Anteil der *Standardreaktion Reflexschritt* hoch, dann ist das Mustererkennungsverfahren *zu optimistisch*, weil es die vermeidbaren Stürze nicht von den unvermeidbaren Stürze unterscheidet.

- Die **Erfolgsquote ϵ** beschreibt das Gleiche wie die Sturzvermeidungsgröße – nämlich inwiefern der Roboter Stürze durch von dem Mustererkennungsverfahren ausgelöste Reaktionen vermeiden kann. Allerdings enthält ϵ keine eine Bewertung, bei welchen Stößen das Fallen eher vermeidbar gewesen wäre. ϵ ist der Quotient aus den Simulationen, bei denen der Roboter nicht gefallen ist, und insgesamt allen Simulationen:

$$\epsilon = \frac{\sum \bar{\gamma}}{\sum \bar{\gamma} + \sum \gamma}. \tag{6.3}$$

- Die **Reaktionszeit τ** ist die Zeitdifferenz von dem Beginn des Stoßes bis zum Ausführen der Reaktion. Ist die Reaktionszeit zu lang, ist der Roboter seit dem Stoß schon zu sehr mit dem Torso in Richtung des Bodens gefallen, so dass er entweder keine Reaktion mehr ausführen oder die Reaktion den Sturz nicht mehr vermeiden kann.

- Die **Rechendauer** bezeichnet den Zeitraum, den ein Mustererkennungsverfahren benötigt, um die Berechnung für einen Messzeitpunkt durchzuführen. Dieser Zeit-

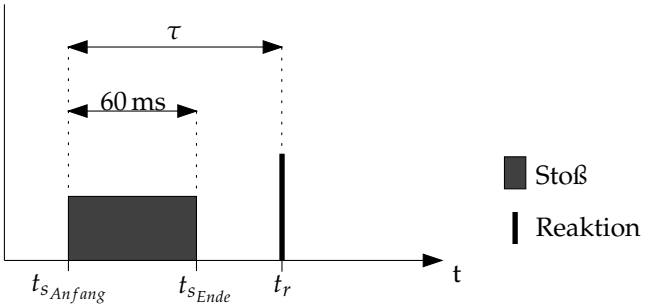


Abbildung 6.2: Zeitdiagramm mit Stoß und Reaktionszeitpunkt. Der Stoß beginnt bei $t_{s_{\text{Anfang}}}$ und endet bei $t_{s_{\text{Ende}}}$. Die Reaktion wird bei t_r ausgeführt.

raum sollte möglichst gering sein. In den hier durchgeführten Simulationen ist die Rechendauer nicht berücksichtigt worden, da die Simulation bei hoher Rechenlast dementsprechend langsamer läuft. In der Wirklichkeit ist die Rechendauer hingegen durchaus ein wichtiger Aspekt, weil das Mustererkennungsverfahren eventuell noch mit veralteten Daten rechnet, wenn der Roboter schon längst gefallen ist.

6.3 Vergleich der Prototypenklassifikationen

In Abschnitt 3.5 ist das Anlernen der Prototypen für die Prototypenklassifikation erläutert worden. Die Prototypen werden durch Mittelwertbildung aus Merkmalsvektoren erstellt, in denen sich verschiedene Sensordaten befinden. Es hat sich die Frage gestellt, zu welchem Zeitpunkt nach dem Stoß diese Merkmalsvektoren auszuwählen sind. Deshalb sind zwei Prototypenklassifikationen erstellt worden. Die Prototypenklassifikation, die mit Merkmalsvektoren angelernt worden ist, die noch während des Stoßes aufgenommen worden sind, heißt PK^{ws}. Die andere Prototypenklassifikation, die mit Merkmalsvektoren nach dem Stoß trainiert worden ist, heißt PK^{ns}. Die PK^{ws} könnte eher auf das Erkennen des Stoßes spezialisiert sein, während die PK^{ns} eher das Fallen generalisiert beschreiben könnte und zum Beispiel auch Fallzustände erkennen, die nicht durch einen Stoß, sondern durch eine Stufe verursacht werden.

In Abbildung 6.4 sind einige Merkmale dieser beiden Prototypen gegenübergestellt:

In den Gegenüberstellungen der Torsotranslationsgeschwindigkeiten in x- und y- Richtung v_{T_y} und v_{T_x} der Prototypen der PK^{ws} (Abbildung 6.4(a)) sind die Klassen in eine

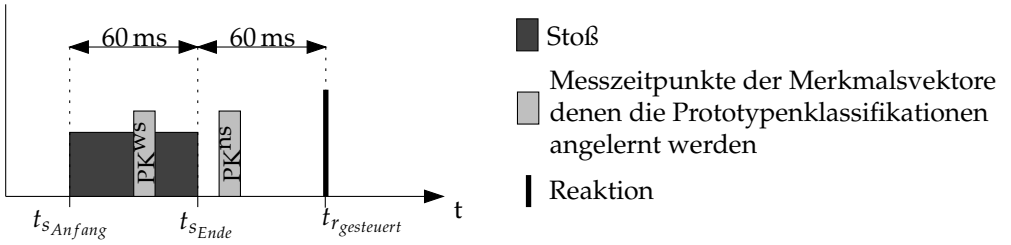


Abbildung 6.3: Zeitdiagramm mit dem Stoß und den Messzeitpunkten für Merkmalsvektoren, mit denen die Prototypenklassifikationen angelernt worden sind

Fallen-Richtung nicht sehr differenziert und teilweise identisch. Bei der Gegenüberstellung der Fußwinkel q_{F_y} und Fußwinkelgeschwindigkeit nach vorne dq_{F_y} haben sogar die Sturz-Klassen einen geringeren Abstand zu den normal-Klassen als die OK-Klassen; es müsste jedoch genau umgekehrt sein, weil beim Sturz die Fußwinkelgeschwindigkeit sich erhöht.

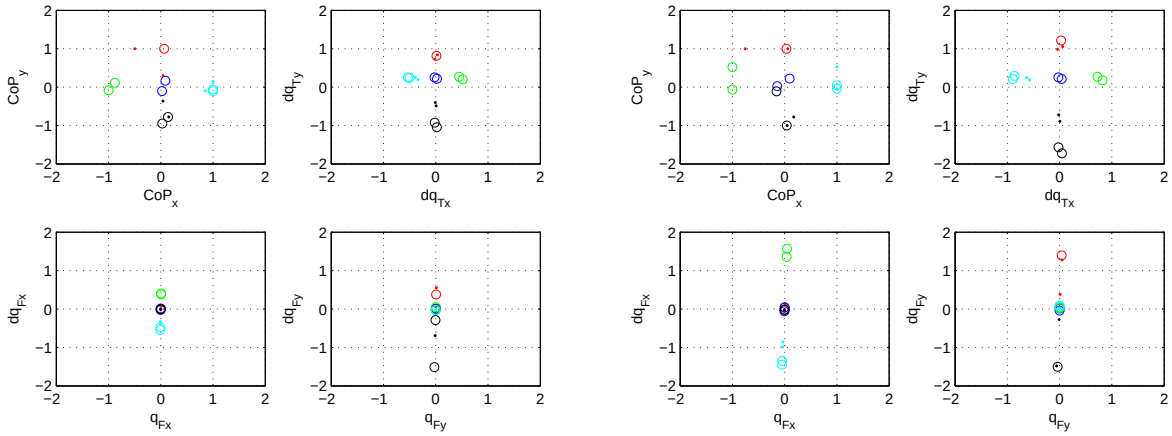
Die Prototypen der PK^{ns} (Abbildung 6.4(b)) stellen sich besser da: Bei den Torsotranslationsgeschwindigkeiten sind die Abstände der OK-Klassen geringer zur normal-Klasse als die der Sturz-Klassen. Die Fußwinkelgeschwindigkeiten der OK-Klassen sind hier nun richtigerweise geringer als die der Sturz-Klassen. Somit scheinen die Prototypen der PK^{ns} die bessere Wahl zu sein, was sich auch in den folgenden Benchmarks zeigen wird.

In Abbildung 6.5 die Bewertung der beiden Prototypenklassifikationen PK^{ws} und PK^{ns} zu finden. Anhand von Abbildung 6.5(a) und Abbildung 6.5(b) lässt sich erkennen, dass sowohl bei der Sturzvermeidungsgröße σ als auch bei der Robustheitsgröße ρ die Kurven der PK^{ns} bei allen Stoßimpulsen unterhalb der Kurven der PK^{ws} liegen. Somit ist die PK^{ns} besser als die PK^{ws}. Der starke Anstieg der Kurven in Abbildung 6.5(b) bei Impulsen $16\,\text{Ns} \leq p \leq 20\,\text{Ns}$ ist durch eine Unstetigkeit von f_ρ[3] zu erklären. σ und ρ sollten allerdings nur relativ verglichen werden.

Die Gründe für die bessere Performance von PK^{ns} lassen sich in den differenzierten Abbildungen zur Stabilitätsanalyse (Abbildung 6.5(c) und Abbildung 6.5(d)) erkennen. Im Folgenden werden die Stoßrichtungen einzeln betrachtet:

- Beim Stoß von hinten haben die PK^{ws} und PK^{ns} zwei Gemeinsamkeiten:

3 ρ erhält aus f_ρ die Werte, die nach verschiedenen Kriterien wie auch der Stoßstärke gestaffelt sind. Bei dem Wechseln von 16 Ns zu 20 Ns fällt der Unterschied von ρ besonders hoch aus.



(a) Prototypen der PK^{ws} (b) Prototypen der PK^{ns}

Abbildung 6.4: Darstellungen einiger Merkmale, aus denen die Prototypen der PK^{ws} und PK^{ns} erzeugt worden sind. Die Punkte entsprechen den OK-Klassen, die Kreise den Sturz-Klassen. Die Farben haben folgenden Klassen-Bedeutung: blau: normal; rot: Fallen nach vorn; schwarz: Fallen nach hinten: grün: Fallen über SWF und cyan: Fallen über STF.

  - Um die Schrittphase $t_p = 0{,}2$ haben beide Prototypenklassifikationen am wenigsten Erfolg. Das liegt – wie schon in Abschnitt 2.5 erwähnt – an der ungünstigen Beinkonfiguration.
  - In den späteren Schrittphasen $t_p \geq 0{,}4$ sind beide erfolgreicher und sogar besser als bei der SmgR.

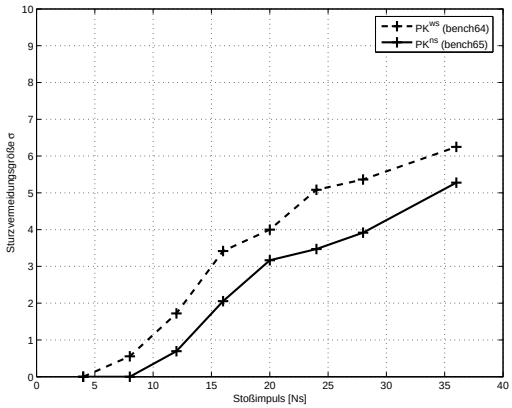
  In früheren Schrittphasen $t_p \leq 0$ unterscheiden sich PK^{ws} und PK^{ns}. Die PK^{ws} klassifiziert im Gegensatz zur PK^{ns} Sturz-Klassen bei geringen Stoßimpulsen und lässt die Schutzreaktion Hocke ausführen. Dadurch können weniger Stürze vermieden werden.

- Beim Stoß von vorne gibt es die Gemeinsamkeit, dass PK^{ws} und PK^{ns} in den späteren Schrittphasen $t_p \geq 0{,}6$ bei geringen Stoßimpulsen keine Reaktion ausführen lassen. Bei der SmgR ist hier die Reaktion ausgeführt worden, was den Roboter erst zu Fall gebracht hat. Ansonsten führt die PK^{ns} fast immer die Standardreaktion aus, kann somit viele Stürze bei starken Stößen vermeiden und ist sogar besser als die SmgR. Die PK^{ws} hingegen lässt den Roboter bei Stoßimpulsen $p \geq 20\,\text{Ns}$ häufiger die Schutzreaktion ausführen und ist somit weniger erfolgreich bei der Sturzvermeidung.

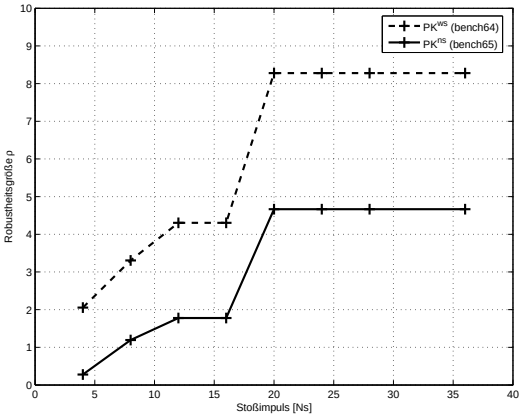
- Beim Stoß über den Standfuß können nicht so viele Stürze vermieden werden. Auffällig ist, dass die PK^{ns} die Eigenstabilität des Roboters berücksichtigt und keine Reaktion ausführt während die PK^{ws} schon die Standardreaktion ausführen lassen muss. Interessant ist auch die Simulation bei $t_p = 0{,}8$ bei einem Stoßimpuls $p = 8\,\text{Ns}$: Die PK^{ws} lässt die Schutzreaktion Hocke ausführen, während die PK^{ns} eine andere Standardreaktion bestimmt. Trotzdem fällt der Roboter bei beiden nicht, während er bei der SmgR mit der ausgeführten Standardreaktion fällt.
- Auch beim Stoß über den Schwingfuß ist die PK^{ns} erfolgreicher als die PK^{ws}, weil die PK^{ws} schon bei Stoßimpulsen $p \geq 12$ Ns die Hocke ausführt. Bei den sehr geringen Stößen $p = 4$ Ns berücksichtigt die PK^{ns} die Eigenstabilität wieder besser als die PK^{ws}.

Abbildung 6.5(g) zeigt, dass die PK^{ns} bei geringen Stoßimpulsen häufiger als die PK^{ws} keine unnötige Reaktion ausführen lässt. Abbildung 6.5(g) zeigt, dass die PK^{ws} bei geringen Stoßimpulsen $p \leq 12\,\text{Ns}$ häufiger die Standardreaktion ansteuert. Bei Stoßimpulsen $p \geq 12\,\text{Ns}$ wiederum führt die PK^{ns} häufiger die Standardreaktion aus. Und Abbildung 6.5(i) zeigt, dass die PK^{ws} zu allen Stoßimpulsen häufiger die Hocke ausführt. Die Hocke wird bei der PK^{ns} erst mit steigendem Stoßimpuls gewählt. Die Differenzierung zwischen normalen, vermeidbaren und nicht mehr vermeidbaren Fallsituationen ist bei der PK^{ws} im Vergleich zur PK^{ns} schlechter, weil die PK^{ws} vermeidbare Fallsituationen schon als nicht mehr vermeidbare einordnet.

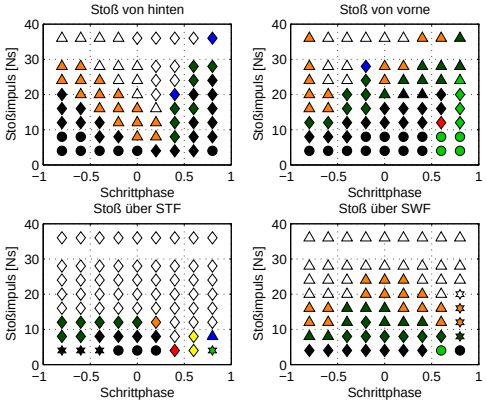
In Abbildung 6.5(e) sind die Reaktionszeiten τ der beiden Prototypenklassifikationen abgebildet. Bei beiden Prototypenklassifikationen ist τ bei Stoßimpulsen ab 12 Ns kleiner als die 120 ms bei der SmgR. Dadurch lassen sich die vereinzelten Verbesserungen beider Prototypenklassifikationen zur SmgR erklären.



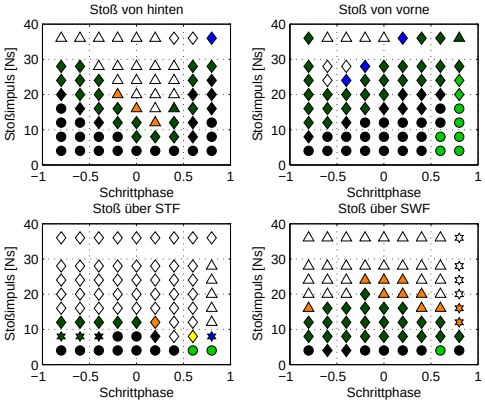
(a) Sturzvermeidungsgröße σ



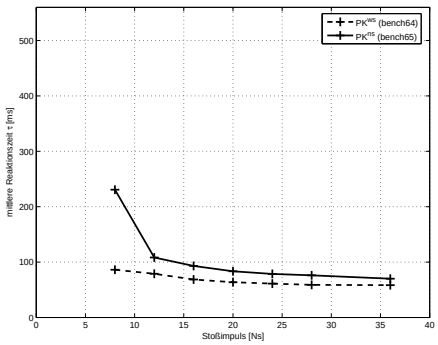
(b) Robustheitsgröße ρ



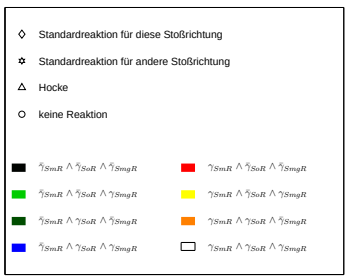
(c) Stabilitätsanalsyse PK^{ws}



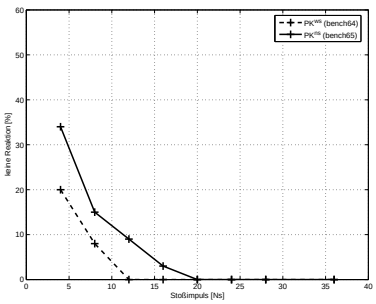
(d) Stabilitätsanalsyse PK^{ns}



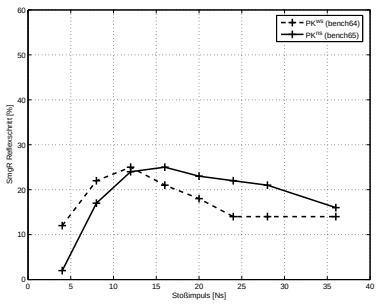
(e) mittlere Reaktionszeit τ



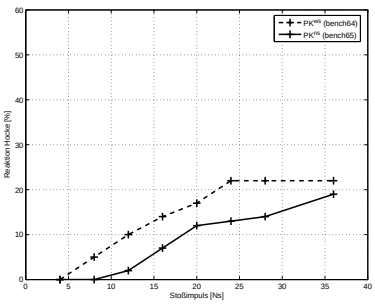
(f) Legende zur Stabilitätsanalyse



(g) keine Reaktion



(h) Standardreaktion Reflexschritt



(i) Reaktion Hocke

Abbildung 6.5: Bewertung der PK^{ws} und PK^{ns}

6.4 Vergleich der Klassifikation mit Hidden Markov Models

Die Performance der Hidden Markov Models hängt von folgenden Parametern ab:

- Anzahl der Trainingsschritte des Baum-Welch-Algorithmus (Unterabschnitt 4.2.4)
- Anzahl der verborgenen Zustände der HMM
- vorgeschaltete Prototypenklassifikation mit der PK^{ws} oder PK^{ns}
- Beobachtungsgenerierung, also die Klassifikation mit dem Parameter-Hidden Markov Model ${}_{p}$HMMK siehe (Unterabschnitt 4.3.1) oder Richtungs-Hidden Markov Model ${}_{r}$HMMK (siehe Unterabschnitt 4.3.2)

Es hat sich gezeigt, dass sich ab einer Anzahl von zehn Trainingsschritten (Abschnitt 4.4) keine Verbesserung der HMM mehr ergibt und die HMM-Parameter konvergiert sind. Da die Trainingsschritte offline durchgeführt werden, ist die benötigte Rechenzeit unbedeutend. Die HMM in dieser Arbeit sind daher immer mit zehn Trainingsschritten angelernt worden.

Die Anzahl der verborgenen Zustände der HMM wurde mit $N \in \{3 \quad 4 \quad 5\}$ gewählt. Eine geringere Anzahl ist mit der Komplexität der möglichen Beobachtungen und Zustandsübergänge ist nicht sinnvoll (Abbildung 4.5). Auf der anderen Seite würde ein HMM mit mehr als sechs Zuständen Eigenschaften aus der Gehen-Bewegung beschreiben.

Die Nomenklatur für die Abkürzung der HMM-Klassifikation ist ${}_{r,p}\text{HMMK}^{ws,ns}_{3,4,5}$. Die Buchstaben *r* und *p* bezeichnen die HMM-Klassifikationen mit der Beobachtungsgenerierung über die Prototypen oder die Richtungen. Die Buchstaben *ws* und *ns* bezeichnen die Prototypenklassifikation, die der HMMK vorgeschaltet ist. Die Ziffern *3*,*4* oder *5* bezeichnen die Anzahl der Zustände in den HMM.

Die beiden HMM-Klassifikationen ${}_{p}$HMMK und ${}_{r}$HMMK werden im Folgenden in Kombinationen mit unterschiedlicher Anzahl an Zuständen und den beiden Prototypenklassifikationen PK^{ws} und PK^{ns} als Beobachtungsgenerierung verglichen.

6.4.1 Klassifikation mit Prototyp-Hidden Markov Models

Die ${}_{\mathrm{p}}\mathrm{HMMK}$ wertet bei der Beobachtungsgenerierung nur die wahrscheinlichste Klasse aus der vorgeschalteten Prototypenklassifikation aus. In den Abbildungen auf Seite 71 werden verschieden ${}_{\mathrm{p}}\mathrm{HMMK}$ untereinander und mit den Prototypenklassifikationen $\mathrm{PK}^{\mathrm{ws}}$ und $\mathrm{PK}^{\mathrm{ns}}$ verglichen.

An Abbildung 6.6(a) und Abbildung 6.6(c) lässt sich erkennen, dass die ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ws}}$ sowohl eine bessere Robustheits- als auch in der Sturzvermeidungsgröße im Vergleich zu der $\mathrm{PK}^{\mathrm{ws}}$ hat. Dabei unterscheiden sich die Kurven der ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ws}}$ mit unterschiedlichen Zuständen geringfügig. Die ${}_{\mathrm{p}}\mathrm{HMMK}_3^{\mathrm{ws}}$ hat die besten Werte, wobei sich die Kurven der Wahrscheinlichkeiten der HMM der ${}_{\mathrm{p}}\mathrm{HMMK}_4^{\mathrm{ws}}$ und ${}_{\mathrm{p}}\mathrm{HMMK}_5^{\mathrm{ws}}$ nicht stark unterscheiden.

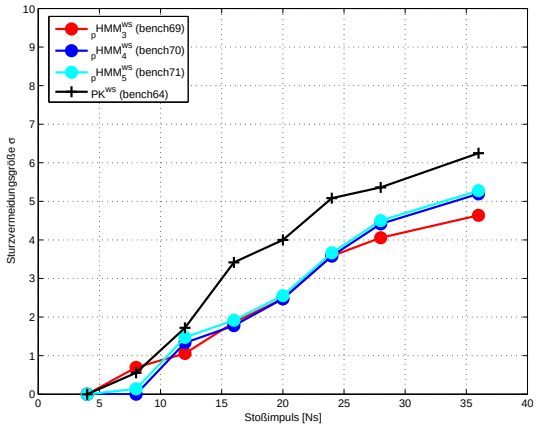
Auf der rechten Seite befinden sich Abbildung 6.6(b) und Abbildung 6.6(d). Hier sind die ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ns}}$ nicht immer besser als das der $\mathrm{PK}^{\mathrm{ns}}$. Die Sturzvermeidungsgröße σ der ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ns}}$ ist nur bei manchen Stoßimpulsen besser als die $\mathrm{PK}^{\mathrm{ns}}$. Die Robustheitsgröße ρ der ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ns}}$ ist sogar schlechter als das der $\mathrm{PK}^{\mathrm{ns}}$. Auch hier schneidet das ${}_{\mathrm{p}}\mathrm{HMMK}_3^{\mathrm{ns}}$ am besten ab.

Beim genaueren Vergleichen der ${}_{\mathrm{p}}\mathrm{HMMK}_i^{\mathrm{ws}}$ und ${}_{\mathrm{p}}\mathrm{HMMK}_i^{\mathrm{ns}}$ mit $i \in \{3 \quad 4 \quad 5\}$ ergibt sich, dass sich die ${}_{\mathrm{p}}\mathrm{HMMK}_i^{\mathrm{ws,ns}}$ gleich verhalten. Sie haben nämlich die gleichen Parameter, sind also identisch. Beim Zurückverfolgen der Trainings-Beobachtungen sind nur geringe Unterschiede zu erkennen[4]. Die Beobachtungsgenerierung ist also identisch.

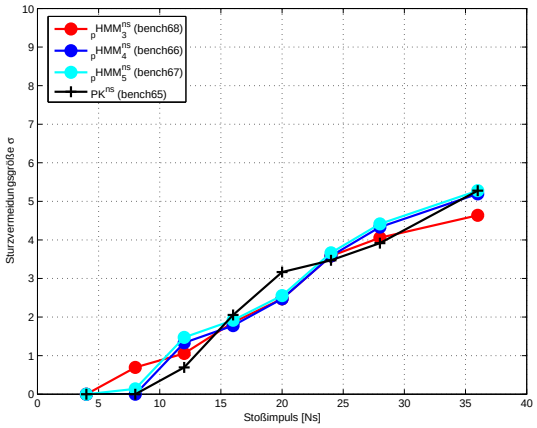
Nun wird im Folgenden nur noch die ${}_{\mathrm{p}}\mathrm{HMMK}_3$ berücksichtigt, weil diese die besten Ergebnisse aufweisen und mit der geringsten Zustandsanzahl auch die wenigsten online-Berechnungen benötigt. Die Stabilitätsanalyse findet sich in Abbildung 6.6(f). Je nach Stoßrichtung kann die ${}_{\mathrm{p}}\mathrm{HMMK}_3^{\mathrm{ns}}$ Stürze entweder nicht so gut oder besser als die SmgR vermeiden. Auffällig ist nur das schlechte Ergebnis beim Stoß über den Schwingfuß bei $t_p = 0$, bei dem die ${}_{\mathrm{p}}\mathrm{HMMK}_3^{\mathrm{ns}}$ einen Sturz erkannt und die Hocke ausgeführt hat.

Die ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ws}}$ kann also das Ergebnis der schlechteren $\mathrm{PK}^{\mathrm{ws}}$ sehr verbessern. Mit der $\mathrm{PK}^{\mathrm{ns}}$ ergibt sich für die ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ns}}$ nur noch teilweise eine Verbesserung.

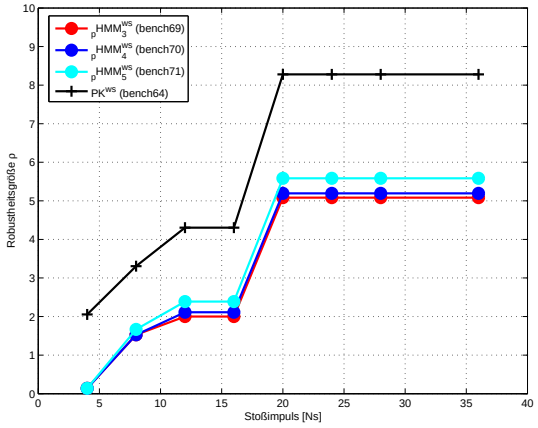
4 die Trainings-Beobachtungen sind bis auf die letzten Beobachtungen identisch. Allerdings finden sich teilweise in einer Beobachtungssequenz bei den frühen Prototypen nur 20 Wiederholungen der gefallen-Beobachtung und beim anderen HMM 21 gefallen-Beobachtungen.



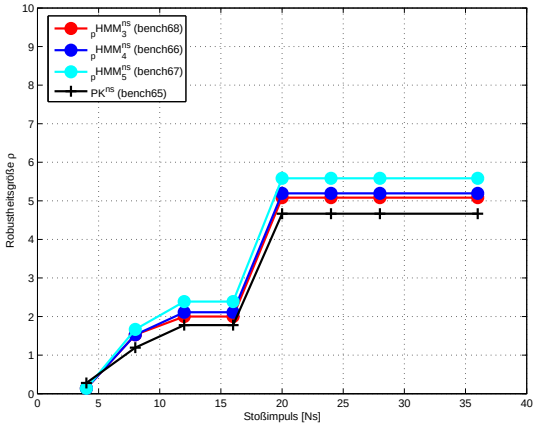
(a) Sturzvermeidungsgröße σ der ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ws}}$



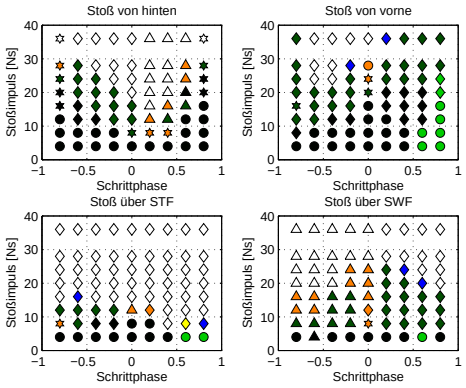
(b) Sturzvermeidungsgröße σ der ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ns}}$



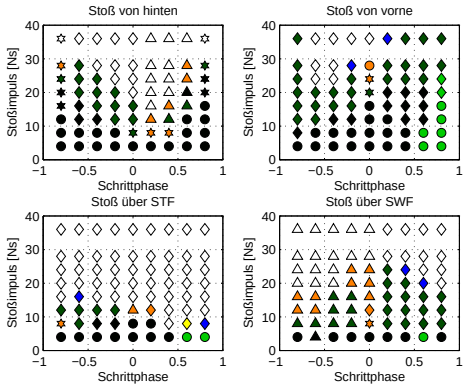
(c) Robustheitsgröße ρ der ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ws}}$



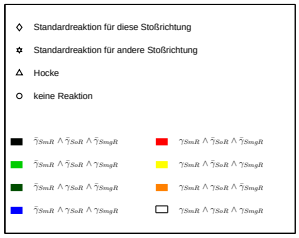
(d) Robustheitsgröße ρ der ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ns}}$



(e) Stabilitätsanalyse der ${}_{\mathrm{p}}\mathrm{HMMK}_3^{\mathrm{ws}}$



(f) Stabilitätsanalyse der ${}_{\mathrm{p}}\mathrm{HMMK}_3^{\mathrm{ns}}$



(g) Legende zur Stabilitätsanalyse

Abbildung 6.6: Abbildungen zur Bewertung der ${}_{\mathrm{p}}$HMMK, die mit verschiedenen Prototypenklassifikationen und verschiedener Anzahl an Zuständen angelernt worden

6.4.2 Klassifikation mit Richtungs-Hidden Markov Models

Die ${}_{\mathrm{r}}$HMMK werten als Beobachtungen die Abstände zu allen Klassen der vorgeschalteten Prototypenklassifikation aus (siehe Unterabschnitt 4.3.2), während die ${}_{\mathrm{p}}$HMMK nur die wahrscheinlichste Klasse auswertet. Die Klassen haben entsprechend ihrer Bedeutung Richtungs- und Betragseigenschaften, die für alle Klassen mit ihrem Klassenabstand gewichtet überlagert werden. Hieraus ergeben sich die Beobachtungen. Die Vergleichsgrößen zur ${}_{\mathrm{r}}$HMMK finden sich in Abbildung 6.7.

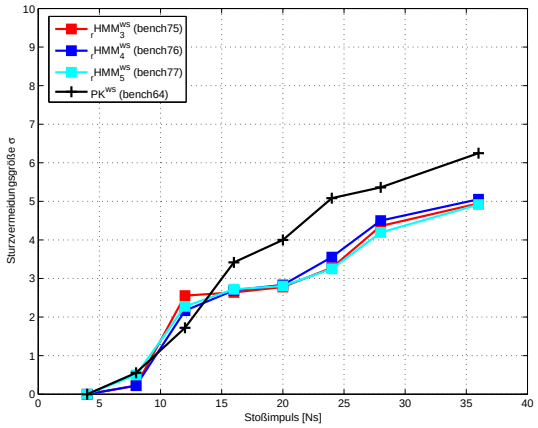
Anhand Abbildung 6.7(a) ist ersichtlich, dass die ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ws}}$ eine bessere Sturzvermeidungsgröße als die $\mathrm{PK}^{\mathrm{ws}}$ erreichen. Dabei ist diesmal die ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ws}}_{5}$ besser, wenn auch die Unterschiede zu den ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ws}}_{3,4}$ sehr gering sind. Abbildung 6.7(c) zeigt, dass die Robustheitsgröße der ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ws}}$ schlechter als die der $\mathrm{PK}^{\mathrm{ws}}$ sind, während sich die ${}_{\mathrm{p}}\mathrm{HMMK}^{\mathrm{ws}}$ hier verbessern konnten.

Abbildung 6.7(b) zeigt, dass die ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ns}}$ bei Stoßimpulsen $p \geq 6\,\mathrm{Ns}$ eine bessere Sturzvermeidungsgröße im Vergleich zu $\mathrm{PK}^{\mathrm{ns}}$ hat. Hierbei ist die Anzahl der Zustände der ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ns}}$ fast nicht relevant. Um Rechenzeit zu sparen, ist immer ein HMM mit wenigen Zuständen zu wählen, da die Anzahl der Zuständen bei der online-Anwendung des Viterbi-Algorithmus quadratisch in die Rechenzeit eingeht (siehe Unterabschnitt 4.2.3). Deshalb ist auch hier die zu bevorzugende Zustandsanzahl drei. In Abbildung 6.7(d) ist die Robustheitsgröße ρ der ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ns}}$ zu sehen. Auch diese ist schlechter als die der $\mathrm{PK}^{\mathrm{ws}}$ und die Kurven der ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ns}}_{3,4,5}$ sind identisch.

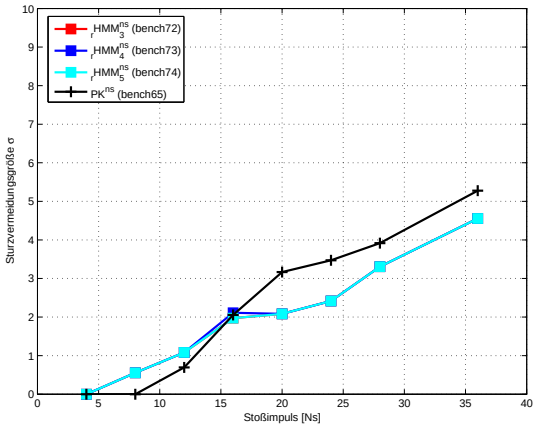
In Abbildung 6.7(f) ist die Stabilitätsanalyse der ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ns}}_{3}$ zu finden. Es fällt auf, dass diese Klassifikation der SmgR schon recht nahe kommt. Bei einigen Stoß-Kombinationen ist sie schlechter:

- Beim Stoß von hinten in der Schrittphase $0 \leq t_p \leq 0{,}2$ lässt die ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ns}}_{3}$ den Roboter die Reaktion Hocke oder keine Reaktion aus, weshalb diese den Sturz nicht vermeiden kann.
- Bei dem Stoß über den Standfuß bei der Schrittphase $0{,}4 \leq t_p \leq 0{,}8$ führt diese auch die Hocke aus und hat so keinen Erfolg.

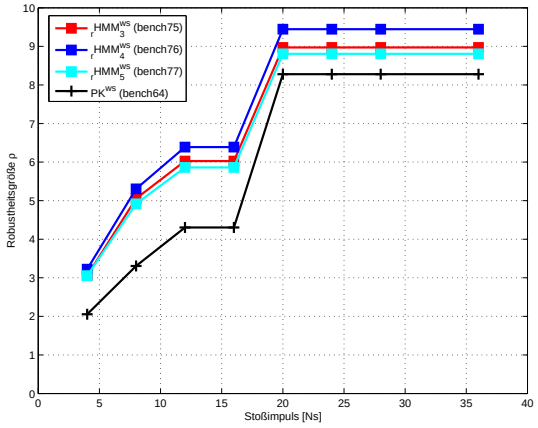
Ansonsten ist die ${}_{\mathrm{r}}\mathrm{HMMK}^{\mathrm{ns}}_{3}$ bei mehrere Kombinationen bei starken Stößen sogar besser als die SmgR.



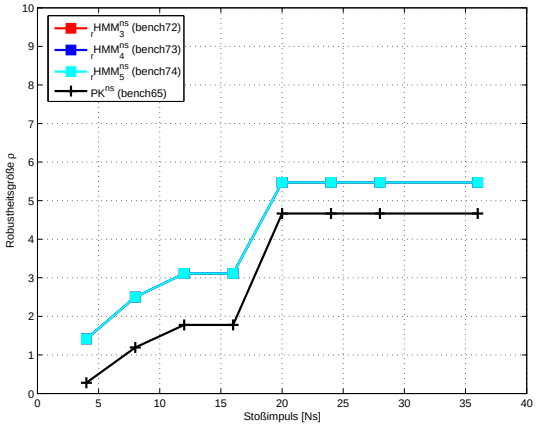
(a) Sturzvermeidungsgröße σ der $_{\mathrm{r}}$HMMK$^{\mathrm{ws}}$



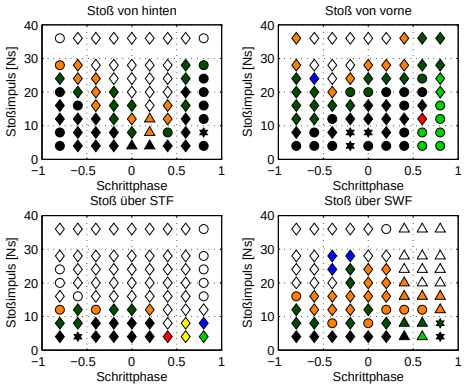
(b) Sturzvermeidungsgröße σ der $_{\mathrm{r}}$HMMK$^{\mathrm{ns}}$



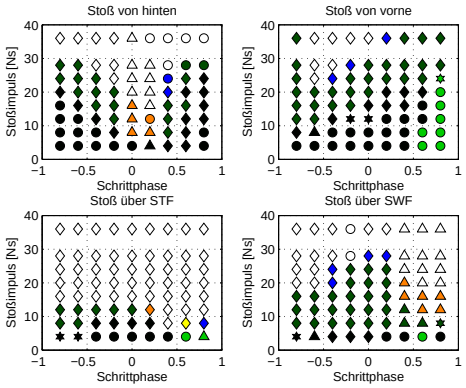
(c) Robustheitsgröße ρ der $_{\mathrm{r}}$HMMK$^{\mathrm{ws}}$



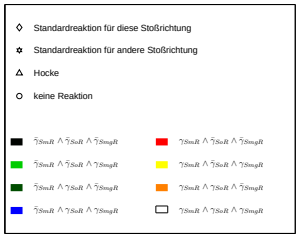
(d) Robustheitsgröße ρ der $_{\mathrm{r}}$HMMK$^{\mathrm{ns}}$



(e) Stabilitätsanalyse der $_{\mathrm{r}}$HMMK$_5^{\mathrm{ws}}$



(f) Stabilitätsanalyse der $_{\mathrm{r}}$HMMK$_3^{\mathrm{ns}}$



(g) Legende zur Stabilitätsanalyse

Abbildung 6.7: Abbildungen zur Bewertung der $_{\mathrm{r}}$HMMK, die mit verschiedenen Prototypenklassifikationen und verschiedener Anzahl an Zuständen angelernt worden

6.5 Vergleich aller Mustererkennungsverfahren

In Abbildung 6.8 und 6.9 sind die Graphen mit den Vergleichsgrößen und die Stabilitätsanalysen zu finden.

Abbildung 6.8(c) zeigt die **Erfolgsquote** der Mustererkennungsverfahren. Anhand des sinkenden Verlaufs der Quoten bei steigendem Stoßimpuls ist ersichtlich, dass nach stärken Stößen der Roboter häufiger fällt. Hierbei wird die Kurve mit der Erfolgsquote der SmgR von drei Mustererkennungsverfahren übertroffen. Das geschieht allerdings erst bei mittleren Stoßimpulsen von $p \approx 20\,\mathrm{Ns}$. Das Mustererkennungsverfahren mit der höchsten Erfolgsquote ist die ${}_{\mathrm{r}}\mathrm{HMMK}_3^{\mathrm{ns}}$, während je nach Stoßimpuls die $\mathrm{PK}^{\mathrm{ns}}$ oder die ${}_{\mathrm{p}}\mathrm{HMMK}_3^{\mathrm{ns}}$ die zweitbesten sind. Die FKbK kann mit diesen nicht mithalten und ist immer schlechter als die SmgR. Die schlechtesten Mustererkennungsverfahren sind bezügliche der Erfolgsquote diejenigen, die auf der $\mathrm{PK}^{\mathrm{ws}}$ – nämlich die $\mathrm{PK}^{\mathrm{ws}}$ und die ${}_{\mathrm{r}}\mathrm{HMMK}_3^{\mathrm{ws}}$ – basieren.

In Abbildung 6.8(e) und Abbildung 6.8(f) sind die Anteile bestimmter **Reaktionen** im Vergleich zu allen Reaktionen aufgeführt. Abbildung 6.8(e) stellt den Anteil der Reaktionen dar, bei denen der Roboter gefallen ist und *nicht* die Schutzreaktion Hocke ausgeführt hat. Es ist klar ersichtlich, dass die SmgR den höchsten Anteil hat, weil in dieser Simulation ja nur die Standardreaktionen und keine Hocke nach der Definition der SmgR ausgeführt worden ist. Hier ist sogar die ${}_{\mathrm{r}}\mathrm{HMMK}_3^{\mathrm{ws}}$ noch schlechter. Ansonsten sind die anderen ${}_{\mathrm{r,p}}\mathrm{HMMK}_3^{\mathrm{ws,ns}}$ und die FKbK besser. Am besten sind hier die beiden Prototypenklassifikationen, die ab mittleren Stoßimpulsen von $p \approx 20\,\mathrm{Ns}$ am häufigsten die Hocke auswählen. Hierzu ist schlüssig, dass die ${}_{\mathrm{r,p}}\mathrm{HMMK}_3^{\mathrm{ws,ns}}$ den Roboter häufiger die Standardreaktion ausführen lassen (Abbildung 6.8(f)) Die FKbK und Prototypenklassifikationen $\mathrm{PK}^{\mathrm{ws,ns}}$ tun dies nicht so häufig, weil sie eher die Hocke auswählen. Die HMMK ermitteln also häufiger die Standardreaktionen, haben damit auch teilweise eine höhere Erfolgsquote, fallen dafür aber häufiger. Sie sind also etwas optimistischer und haben damit auch partielle Erfolge.

Abbildung 6.8(d) lässt sich ein Abfallen der **Reaktionszeiten** erkennen. Bei geringen Stoßimpulsen $p \leq 12\mathrm{Ns}$ sind die Reaktionszeiten bei einigen Mustererkennungsverfahren besonders hoch, weil bei diesen Stoßimpulsen nur wenige Reaktionen überhaupt ausgeführt werden und der Mittelwert nur aus diesen wenigen Werten berechnet wird, bei denen die Mustererkennungsverfahren meist eine Fehlklassifikation haben. Die festgelegte

konstante Reaktionszeit der SmgR von $\tau = 120\,\text{ms}$ ist gut zu erkennen, die fast alle Mustererkennungsverfahren bei Stoßimpulsen $t \geq 12\,\text{Ns}$ unterschreiten. Es ist auch auffällig, dass die Mustererkennungsverfahren, die auf der PK^{ws} basieren auch kürzere Reaktionszeiten, während die auf der PK^{ns} basierenden etwas höhere Reaktionszeiten besitzen. Die Prototypen der PK^{ws} besitzen nämlich mehr Informationen über den Stoß selbst und die frühe Sturzstadien und können diese dementsprechend früher erkennen, weil diese auch früher auftreten. Die Reaktionszeit der FKbK ist die höchste und überschreitet sogar bei $p = 24\,\text{Ns}$ die der SmgR.

Allerdings sind in diesen Reaktionszeiten nicht die **Rechenzeiten** enthalten, die ein Mustererkennungsverfahren benötigt, um aus den Daten eine Klassifikation durchzuführen. Deshalb sind die Rechenzeiten der in MATLAB implementierten Mustererkennungsverfahren als Vergleich in Tabelle 6.1 herangezogen worden. Es ist leicht ersichtlich, dass die HMMK mindestens um den Faktor 200 langsamer als die PK ist. Diese Tatsache kann dadurch entschärft werden, dass nur bestimmte Gruppen von HMMK zum Beispiel in einer Fallrichtung berechnet werden können. Auch lässt sich erkennen, wie die Rechenzeit der HMMK bei steigender Zustandsanzahl analog zu N^2T wächst. Es sollten also HMMK mit möglichst wenig Zuständen benutzt werden.

	PK	HMMK_3	HMMK_4	HMMK_5	FKbK
Rechenzeit pro Messzeitpunkt [ms]	0,5336	104	166	239	0,119

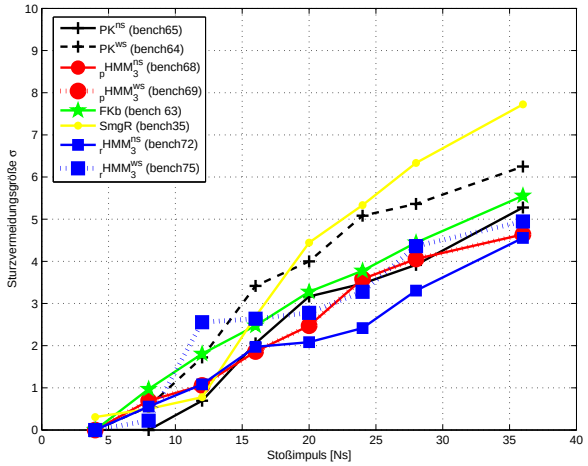
Tabelle 6.1: Rechenzeiten der Mustererkennungsverfahren. Diese Zeiten sind mit der in MATLAB implementierten Mustererkennungsverfahren bei 864 einzelnen Messzeitpunkten durchgeführt worden. Die Anzahl der Prototypenklassen sind 16 und die der Anzahl der HMM ist 30. Diese Berechnungen sind mit MATLAB 7 (R14) unter einem Intel Pentium IV mit 3 GHz durchgeführt worden.

Nachdem die Mustererkennungsverfahren mit diesen nicht bewerteten Vergleichsgrößen aneinander gemessen worden sind, sollen nun noch die Sturzvermeidungsgröße σ und die Robustheitsgröße ρ betrachtet werden. In Abbildung 6.8(a) ist die **Sturzvermeidungsgröße** σ zu finden. Wie bei der Erfolgsquote ist die ${}_{\text{r}}\text{HMMK}_3^{\text{ns}}$ das beste Mustererkennungsverfahren. Danach folgen im engen Abstand die ${}_{\text{p}}\text{HMMK}_3^{\text{ws}}$, die anderen HMMK und die FKbK. Die PK^{ws} ist diesmal das schlechteste Mustererkennungsverfahren. Trotzdem ist aber selbst die PK^{ws} besser als die SmgR.

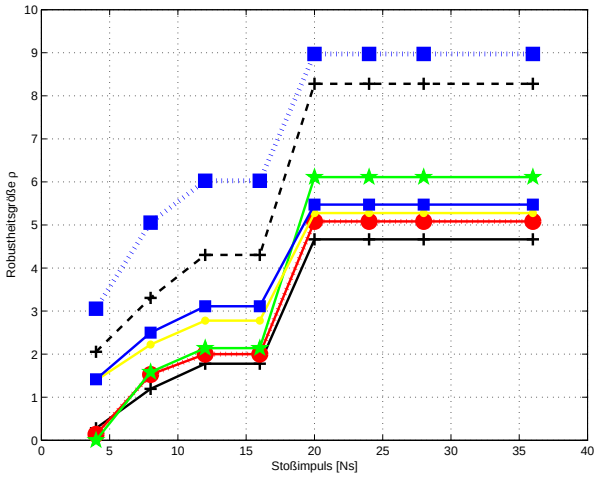
Bei der **Robustheitsgröße** (Abbildung 6.8(b)) ist die Reihenfolge etwas anders. Hier ist

die PK^{ws} das beste Verfahren. Es folgen die ${}_pHMMK_3^{ns}$, ${}_rHMMK_3^{ns}$ und in etwas größerem Abstand die FKbK . Weit abgeschlagen sind wieder die PK^{ws} und ${}_rHMMK_3^{ws}$.

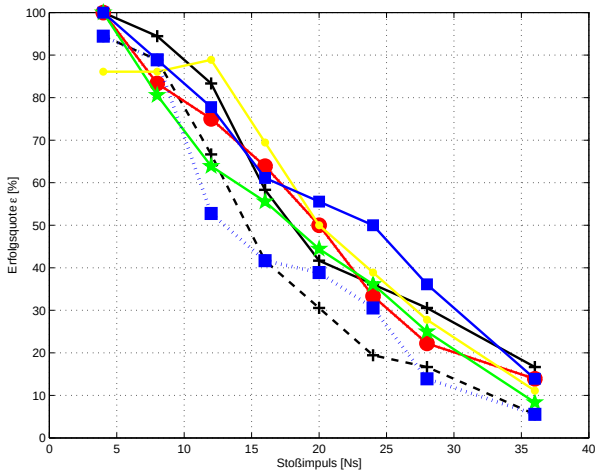
Die ${}_rHMMK_3^{ns}$ ist wegen seiner guten Sturzvermeidungsgröße und Erfolgsquote und einer relativ guten Robustheitsbewertung das ausgeglichenste. Die HMM können somit die Ergebnisse der Prototypenklassifikation verbessern, allerdings mit der Voraussetzung von genügend Rechenleistung. Ist diese nicht vorhanden, ist die PK^{ns} die bessere Wahl, die sogar noch robuster als die ${}_rHMMK_3^{ns}$ ist.



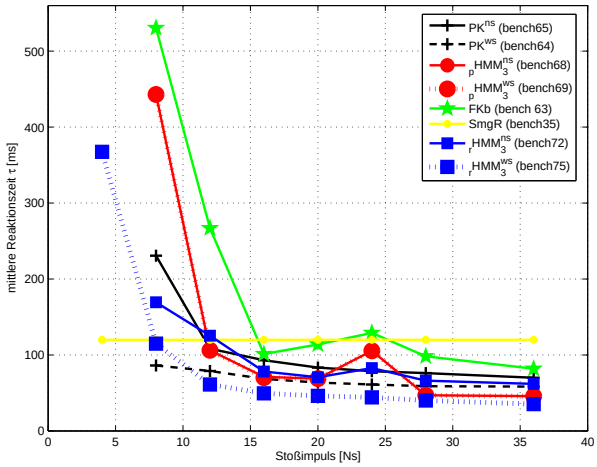
(a) Sturzvermeidungsgröße σ



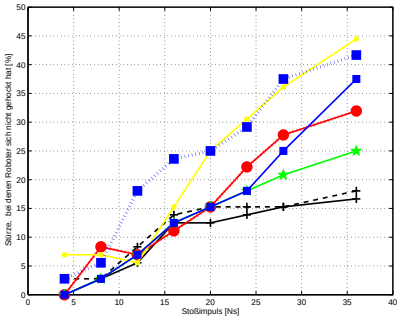
(b) Robustheitsgröße ρ



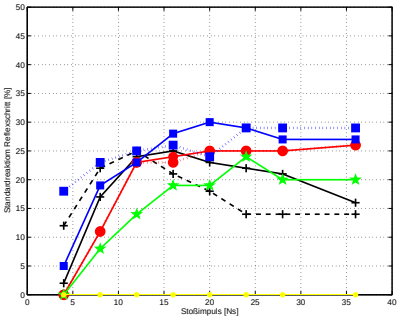
(c) Erfolgsquote ϵ



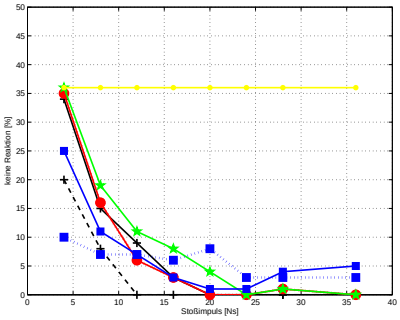
(d) mittlere Reaktionszeit τ



(e) Stürze ohne Hocke

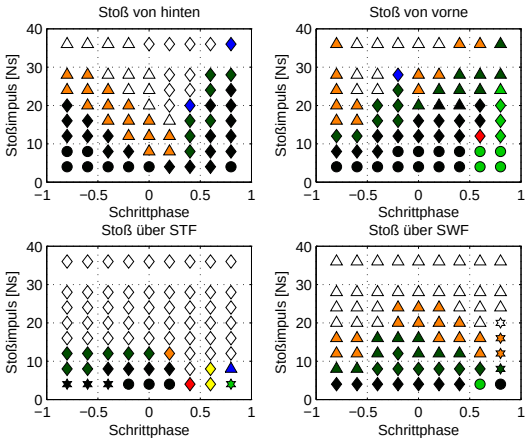


(f) Standardreaktion Reflexschritt

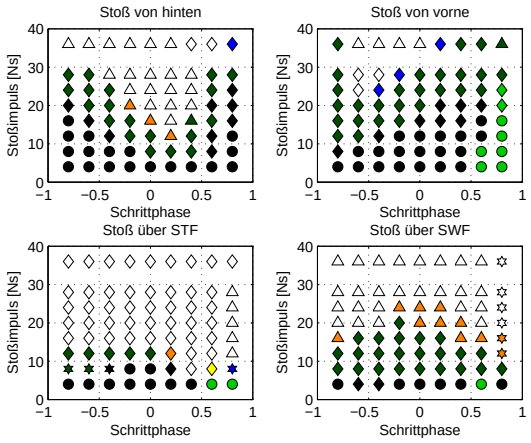


(g) keine Reaktion

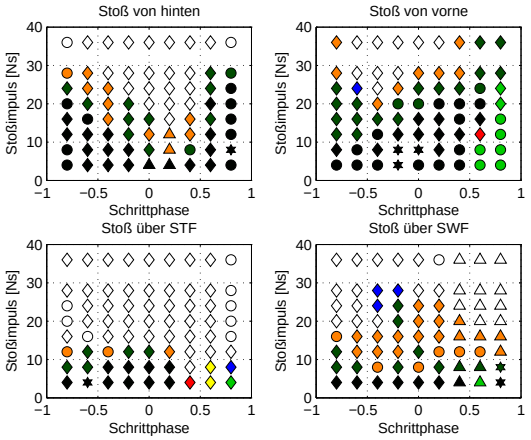
Abbildung 6.8: Abbildungen zur Bewertung der verschiedenen Mustererkennungsverfahren



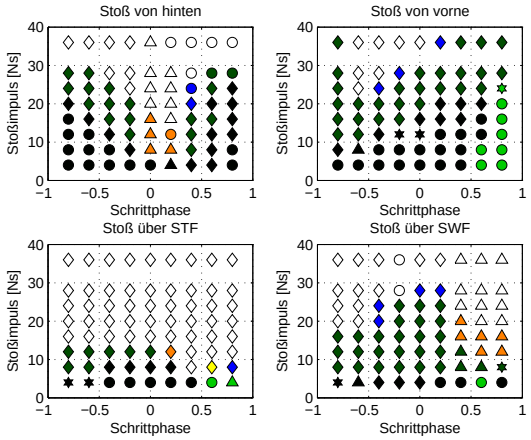
(a) Stabilitätsanalyse der PK^{ws}



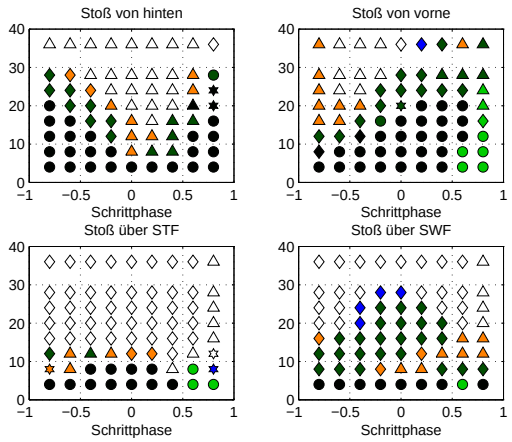
(b) Stabilitätsanalyse der PK^{ns}



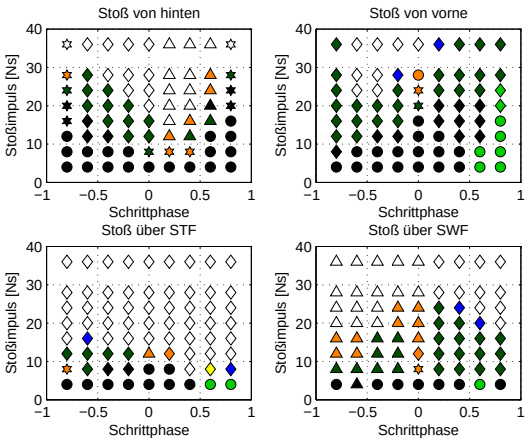
(c) Stabilitätsanalyse der ${}_{r}HMMK_{5}^{ws}$



(d) Stabilitätsanalyse der ${}_{r}HMMK_{3}^{ns}$



(e) Stabilitätsanalyse der FKbK



(f) Stabilitätsanalyse der ${}_{p}HMMK_{3}^{ns}$

Abbildung 6.9: Abbildungen zur Stabilitätsanalyse der verschiedenen Mustererkennungsverfahren bei verschiedenen Stoßimpulsen aus verschiedenen Richtungen.

7 Verifikation der Simulationsergebnisse an BARt-UH

In den bisherigen Kapiteln sind die Mustererkennungsverfahren mit Hilfe des Roboter-Simulator untersucht worden. In diesem Kapitel wird die Frage beantwortet, ob diese Verfahren auch mit dem Roboter BARt-UH funktionieren. Dabei wird besonders darauf geachtet, ob die Mustererkennungsverfahren, die mit dem Simulator angelernt wurden, auf BARt-UH übertragen werden können und auch dort sinnvolle Ergebnisse liefern.

7.1 Stoßversuche und Messungen an BARt-UH

BARt-UH ist mit den in Abschnitt 2.1 beschriebenen Sensoren ausgestattet:

- Zwei Gleichgewichtssensoren (IMU), die Winkel und Winkelgeschwindigkeit messen können, sind auf dem Torso und dem Standfuß angebracht.
- Kraftsensoren befinden sich unter dem Standfuß, mit denen der CoP bestimmt werden kann.
- Beschleunigungssensoren in x- und y-Richtung sind auf dem Torso befestigt.

BARt-UH sendet die Sensordaten alle 10 ms an einen PC, der die Daten für die spätere Auswertung aufzeichnet. Die Stromversorgung für BARt-UH erfolgt nicht über Akkus, sondern über ein Netzteil, dem zwei Kondensatoren nachgeschaltet sind, um die Stromstärke für kurze hohe Stromspitzen beim ersten Anschalten oder schnellen Beschleunigungen der Motoren bereitstellen zu können.

Damit BARt-UH nach den Stößen nicht zu Boden fällt, ist er mit zwei Seilen an der Decke gesichert. Die Stöße werden mit einem Stoßgewicht mit einer Masse $m_{Sg} = 2{,}7\,\text{kg}$ appliziert, das ebenfalls an zwei Seilen an der Decke befestigt ist (siehe Abbildung 7.1). Dabei wird das Stoßgewicht um die Höhen $\Delta h \in \{0{,}39\,\text{m} \quad 0{,}54\,\text{m} \quad 0{,}63\,\text{m}\}$ von seiner Ruhelage ausgelenkt. Nach dem Loslassen des Stoßgewichts wird die potentielle Energie des Gewichts E_{pot} bis zum tiefsten Punkt h_0 in kinetische Energie E_{kin} umgewandelt. An dieser Position befindet sich BARt-UH und wird von dem Gewicht getroffen. Bei dem Stoß überträgt sich ein Teil von E_{kin} auf den Roboter. Je nach Stoßstärke kann der Roboter so zum Sturz gebracht werden. Da der Stoß nicht ideal elastisch ist, geht ein weiterer Teil der Energie durch die Verformung des Stoßgewichtes verloren. Das nun zurückschwingende Stoßgewicht muss nun aufgefangen werden, damit es mit einem weiteren Schwingen BARt-UH nicht erneut trifft. Da mit BARt-UH nur Bewegungen in y-Richtung ausgeführt werden können, sind nur die Stoßrichtungen $\in \{\text{Stoß von vorne} \quad \text{Stoß von hinten}\}$ möglich gewesen. Um die Stoßversuche besser analysieren zu können, sind sie mit einer Videokamera aufgenommen worden.

Die Stoßversuche sind größtenteils während des Laufens von BARt-UH durchgeführt worden. Dabei wird BARt-UH mit dem rechten Fuß als Standfuß gestartet und führt einen Schritt aus. Wenn der rechte Fuß nach einem Schritt wieder der Standfuß ist, ist der Zeitraum für den Stoß, weil sich die μIMU und die Fußkraftsensoren am rechten Fuß befinden und die Sensordaten des Standfußes vom Interesse sind. Da das Loslassen des Stoßgewichtes manuell erfolgt, zeigen einige LED an BARt-UH den Zeitpunkt hierfür an. Somit können Stöße zu verschiedenen Zeitpunkten reproduzierbar durchgeführt werden. Bei mehren Stoßversuchen hintereinander werden die Motoren in den Gelenken schnell warm, so dass nicht viele Versuche hintereinander durchgeführt werden können, ohne sie zu beschädigen. Die Motoren sind für ein schnelles Fortführen der Versuche mit einem Lüfter gekühlt worden.

Damit BARt-UH beim Stehen auf einem Fuß nicht über den Schwingfuß fällt, sind die Füße in der Mitte unter dem Torso verschoben angebracht. Allerdings muss BARt-UH mit beiden gestreckten Beinen auch stehen können, bei denen die Füße sich nebeneinander befinden. Aus diesem Grund sind die Füße ineinander verschränkt (Abbildung 7.2(b)). Um die Stabilität nach vorne und hinten in y-Richtung zu erhöhen, ist noch eine Fußerweiterung angebracht worden. Bei den Stoßversuchen ist BARt-UH immer gestoßen worden, wenn er auf dem rechten Fuß stand. Wird BARt-UH nun von vorne gestoßen, fällt er nach hinten, kippt über eine Ecke der Fußerweiterung zur Seite und fällt. Ohne die Fußerwei-

terung würde BARt-UH sich normal nach hinten bewegen und könnte durch seine hohe Eigenstabilität in dieser Richtung das Fallen häufig auch ohne Reaktion vermeiden. Es hat sich gezeigt, dass die Fußerweitertung nicht die Stabilität erhöht sondern vermindert.

Außerdem ist nach einem Stoß von hinten mit einer einfachen Sturzdetektion mittels der Torsobeschleunigung in y-Richtung ein Schritt nach vorne ausgeführt worden. Daran hat sich gezeigt, dass BARt-UH sich mit einem Schritt stabilisieren kann.

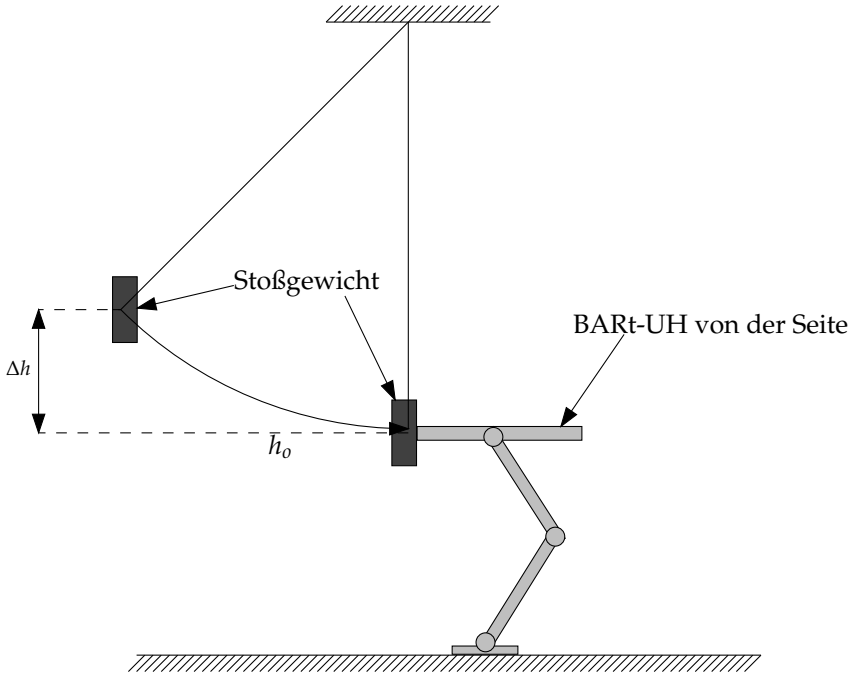


Abbildung 7.1: Schematischer und nicht maßstabsgetreuer Versuchaufbau mit dem Stoßgewicht, das an BARt-UH stößt

Für die weitere Berechnung ist die Stoßziffer e nötig. Dabei wird die Höhe des Stoßgewichts vor und nach dem Stoß in Verbindung gebracht:

$$e = \frac{v_{Sg_{vorStoss}}}{v_{Sg_{nachStoss}}} = \frac{\sqrt{2gh_{Sg_{vorStoss}}}}{\sqrt{2gh_{Sg_{nachStoss}}}} = \sqrt{\frac{h_{Sg_{vorStoss}}}{h_{Sg_{nachStoss}}}} = \sqrt{\frac{0{,}80\,\mathrm{m}}{1{,}18\,\mathrm{m}}} \approx 0{,}82 \qquad \text{mit} \quad v_{BARt} = 0. \tag{7.1}$$

Damit die Stoßversuche mit den Simulationen vergleichbar seien können, muss der Stoßimpuls der Versuche für jede Auslenkung Δh bestimmt werden. Dies ist auf zwei Wegen durchgeführt worden:

1. Die Geschwindigkeit von BARt beim zentralen Stoß ist nach [BHP01]

$$v_{BARt} = \frac{(m_{Sg} - e \cdot m_{BARt}) \cdot v_{BARt_{vorStoss}} + (1+e) \cdot m_{BARt} \cdot v_{Sg}}{m_{Sg} + m_{BARt}} \text{ mit } v_{Sg} = m_{Sg} \cdot g \cdot \Delta h. \tag{7.2}$$

   Der Impuls, den BARt erhalten hat, ist

$$p_v = m_{BARt} \cdot v_{BARt}. \tag{7.3}$$

2. Der Impuls lässt sich auch über die Torsowinkelgeschwindigkeit berechnen, die der Roboter kurz nach dem Stoß hat. Mit der Vereinfachung, dass die translatorschen Geschwindigkeit, die der Roboter durch den Stoßimpuls hat, mit der rotatorischen Geschwindigkeit für den sehr kurzen Zeitraum gleichgesetzt werden kann, ergibt sich aus

$$p_\omega = m_{BARt} \cdot v_{BARt} \qquad \text{mit} \qquad v_{BARt} = l_{Bein} \cdot \omega \tag{7.4}$$

   mit der Länge l_{Bein} des Standbeines

$$p_\omega = m_{BARt} \cdot l_{Bein} \cdot \omega. \tag{7.5}$$

In Tabelle 7.1 sind die bestimmten Impulse für die verschiedenen Auslenkungen aufgelistet. Über beide Rechenwege ergeben sich ähnliche Werte.

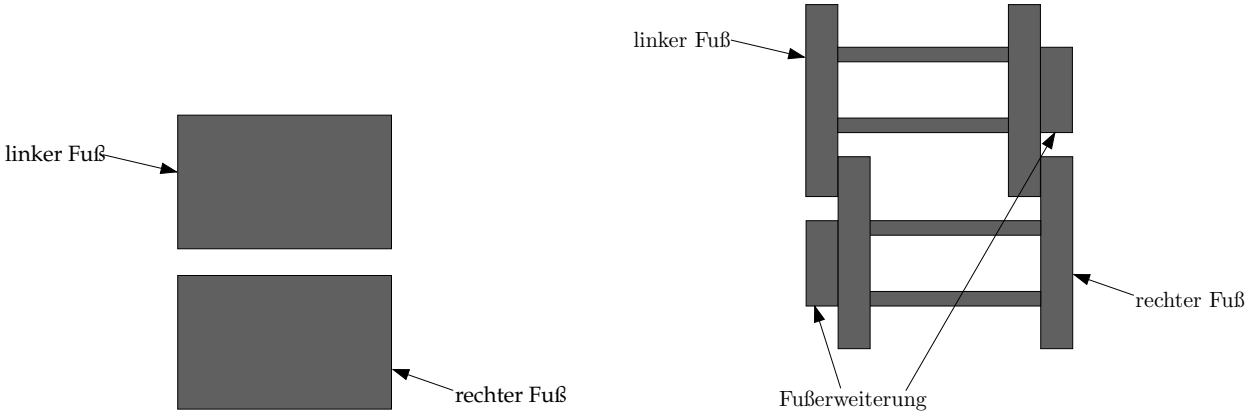
	Auslenkung Δh		
	0,39 m	0,54 m	0,63 m
p_v	12 Ns	14,1 Ns	15,2 Ns
p_ω	11,2 Ns	14 Ns	16,8 Ns

Tabelle 7.1: berechnete Impulse der Stoßversuche mit BARt-UH mit den Parametern $e = 0{,}82$, $m_{Sg} = 2{,}7\,\text{kg}$, $m_{BARt} = 20\,\text{kg}$, ${}_0v_{BARt} = 0$, $g = 9{,}82\frac{\text{N}}{\text{kg}}$, $l_{Bein} = 0{,}7\,\text{m}$, $\omega \in \{0{,}8\,\frac{\text{rad}}{\text{s}} \quad 1{,}0\,\frac{\text{rad}}{\text{s}} \quad 1{,}2\,\frac{\text{rad}}{\text{s}}\}$.

7.2 Die an BARt-UH angepasste Simulation

Das zuvor untersuchte simulierte Robotermodell ist modifiziert worden, so dass er BARt-UH mehr ähnelt. Es hat nun nur noch sechs Freiheitsgrade, größeren Füße mit den Fußerweiterungen und bewegt sich mit der gleichen Schritttrajektorie wie der reale BARt-UH.

Die Füße des Simulators in der unmodifizierten Version sind rechteckig und kleiner (Abbildung 7.2(a)), um dem simulierten Roboter nicht zu viel Stabilität durch die Füße zu geben. Durch die zusätzlichen Freiheitsgrade können die Füße auch beliebig seitlich auf dem Boden aufgesetzt werden, so dass keine besondere Fußform gewählt werden musste.



(a) Füße des unmodifizierten simulierten BARt-UH

(b) Füße des modifizierten simulierten BARt-UH

Abbildung 7.2: verschiedene Füße des simulierten Roboters

Mit diesem Simulator mit dem modifizierten Robotermodell ist auch eine Stabilitätsanalyse durchgeführt worden, um diese mit der des unmodifizierten Robotermodell (siehe Abschnitt 2.5) vergleichen zu können. Allerdings musste diese Stabilitätsanalyse wegen der Fußerweiterung an den Füßen zweimal durchgeführt werden, so dass der simulierte Roboter gestoßen wird, wenn sich die Fußerweiterung sowohl vorne als auch hinten am Fuß befindet. In Abbildung 7.3 sind hierzu die Stabilitätsanalysen zu finden.

Nach einem **Stoß von vorne** ist die SmgR mit dem rechten Fuß als Standfuß (Abbildung 7.3(d)) besonders in späteren Schrittphasen $t_p \geq 0{,}4$ instabiler als die SmgR mit dem

linken Fuß als Standfuß (Abbildung 7.3(b)). Bei den SoR ist dieser Unterschiede nicht so ausgeprägt. Einerseits ist der Roboter in Schrittphasen $t_p \geq -0{,}2$ in der SoR mit dem rechten Fuß als Standfuß (Abbildung 7.3(c)) zwar stabiler ist als in der SoR mit dem linken Fuß als Standfuß (Abbildung 7.3(a)). Andererseits scheint die Fußerweiterung in früheren Schrittphasen $t_p < -0{,}2$ stabilisierend zu wirken, da in diesen Kombinationen der Schwingfuß nämlich noch eine geringe Entfernung zum Boden hat.

Nach einem **Stoß von hinten** ist der Roboter bei der SoR mit dem linken Fuß als Standfuß (Abbildung 7.3(a)) in den Schrittphasen $t_p \leq 0,6$ instabiler als bei der SoR mit dem rechten Fuß als Standfuß (Abbildung 7.3(c)). Bei der SmgR gibt es fast keine Unterschiede.

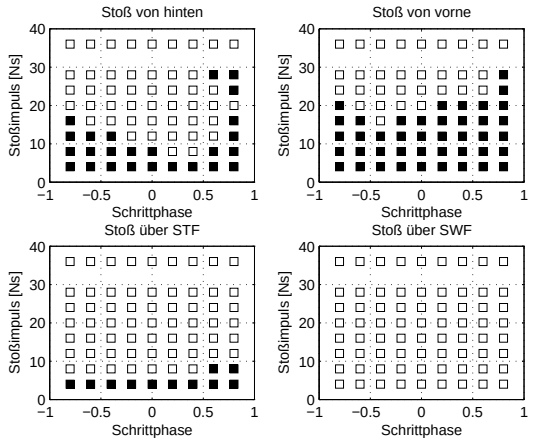
Die Fußerweiterung ist also – wie schon im vorherigen Kapitel festgestellt – der Stabilität in der Regel nicht förderlich.

7.3 Vergleich der Messwerte

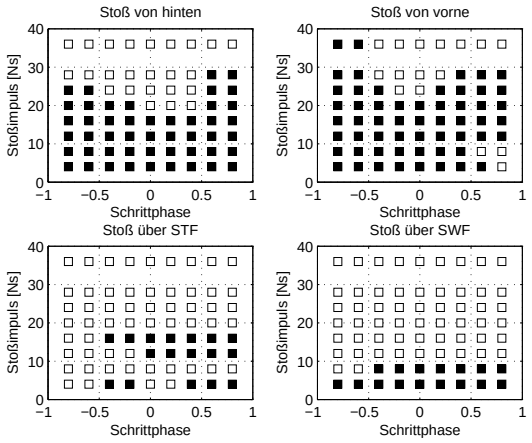
Ähnliche Verläufe der Merkmale sind die Voraussetzung für eine Übertragung der mit dem Simulator mit dem modifizierten Robotermodell angelernten Mustererkennungsverfahren auf BARt-UH. Deshalb sind einige Stoßversuche an BARt-UH mit dem Simulator mit dem modifizierten Robotermodell nachgestellt worden. Dabei können die Parameter Stoßzeitpunkt, Stoßimpuls und Stoßdauer für die Simulation aus den Messdaten des Stoßversuches an BARt-UH gewonnen werden.

Im Folgenden werden nur Messwerte untersucht, die Eigenschaften der Bewegung nach vorne und hinten aufzeigen, da nur in dieser Richtung gestoßen worden ist. Es hat sich gezeigt, dass vor allem vier Merkmale einen sehr ähnlichen Verlauf haben: der CoP in y-Richtung, der Fußwinkel in y-Richtung, die Fußwinkelgeschwindigkeit in y-Richtung und die Torsorotationsgeschwindigkeit in y-Richtung. In Abbildung 7.4 sind die Kurven dieser Messwerte nach einem Stoß von hinten an BARt-UH und dem Simulator übereinander aufgetragen:

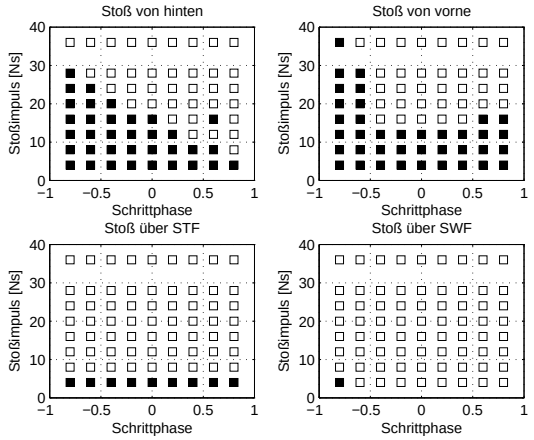
- Der Verlauf des **CoP in y-Richtung** steigt beim Simulator nach dem Stoß sehr schnell auf den Wert 1 und verbleibt dort. Der CoP in y-Richtung steigt beim BARt-UH nicht so schnell und verbleibt bei einem Wert unterhalb von 1, obwohl die beiden hinteren



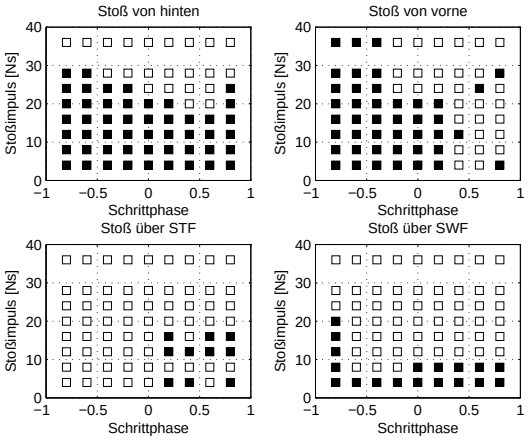
(a) SoR, linker Fuß ist STF (Fußerweiterung vorne)



(b) SmgR, linker Fuß ist STF (Fußerweiterung vorne)



(c) SoR, rechter Fuß ist STF (Fußerweiterung hinten)



(d) SmgR, rechter Fuß ist STF (Fußerweiterung hinten)

Abbildung 7.3: Abbildungen zur Stabilitätsanalyse mit dem Simulator mit dem modifizierten Robotermodell. Kasten mit schwarzem Inhalt heißt *Roboter ist nicht gefallen*, Kasten mit weißer Fläche heißt *Roboter ist gefallen*

Fußkanten den Boden nicht mehr berühren; danach sinkt er wieder, weil BARt-UH über die Fußkante nach vorne fällt, an der sich keine Fußkraftsensoren befinden. Die Tatsache, dass der CoP in y-Richtung nicht den Wert 1 erreicht, kann an einem falschen Offset bei der Berechnung des CoP liegen.

- Die **Torsorotationsgeschwindigkeiten** steigen nach dem Stoß sehr ähnlich auf ein gleiches Maximum bei $1{,}5\frac{rad}{s}$ an und fallen danach ähnlich ab.

- Die **Fußwinkel** haben einen ähnlichen qualitativen Verlauf mit einem lokalen Maximum bei $t = 5{,}9\,\text{s}$; der Fußwinkel von BARt-UH ist allerdings höher. Auch die **Fußwinkelgeschwindigkeiten** haben einen ähnlichen Verlauf mit höheren Werten bei den Messungen an BARt-UH.

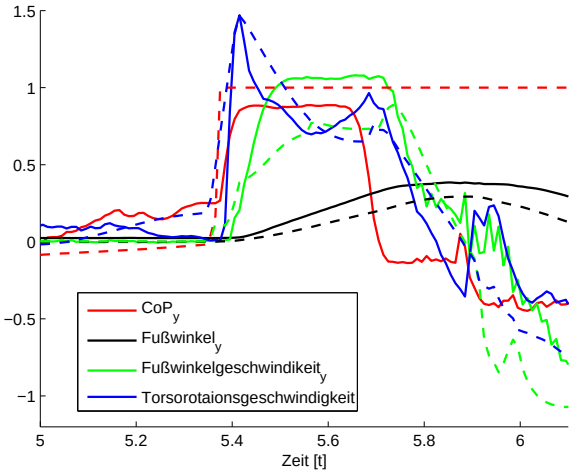


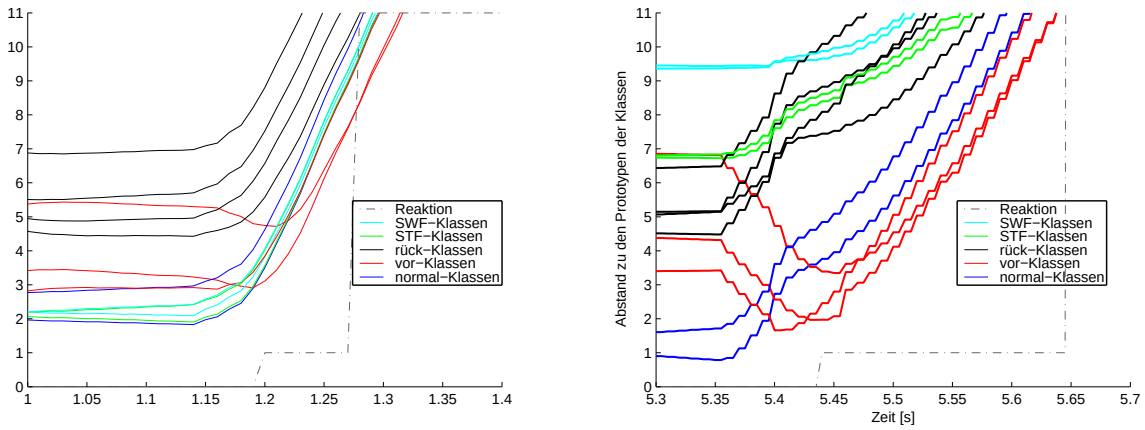
Abbildung 7.4: Diese Abbildung zeigt die Verläufe einiger Merkmale des Simulators und BARt bei einem ähnlichen Stoß von hinten bei 5,35 s ($t_p = 0{,}35$). Der errechnete Stoßimpuls im Stoßversuch ist $p_{BARt} = 14\,\text{Ns}$ und der im Simulator angebrachte Stoßimpuls ist $p_{Simulator} = 18\,\text{Ns}$ gewesen. Die gestrichelten Kurven sind die in der Simulation erzeugten Merkmale, die durchgezogenen Linien sind die gemessenen Merkmale an BARt. Die Einheiten sind für die Winkelgeschwindigkeiten $\frac{\text{rad}}{s}$ und für den Winkel rad.

7.4 Transfer der Prototypen auf BARt-UH

Die Prototypen für den Simulator mit dem modifizierten Robotermodell sind nach dem Verfahren, das in Abschnitt 3.4 beschrieben wird, angelernt worden. Anhand der Abbildungen der Stabilitätsanalyse dieses Simulators mit dem modifizierten Robotermodell (Abbildung 7.3) werden die Stoßimpulse für jede Stoßrichtung ausgewählt. Da die Stoßversuche an BARt-UH nur mit dem rechten Fuß als Standfuß gemacht worden sind, werden auch nur die entsprechenden Abbildungen der Stabilitätsanalyse betrachtet. Die Merkmalsvektoren bei der SoR, bei denen der Simulator auch ohne Reaktion nicht fällt (Abbildung 7.3(c)), werden der normal-Klassen zugeordnet. Dann wird die SmgR in Abbildung 7.3(d) betrachtet und es werden die Stoßimpulse bestimmt, bei denen ein Fallen durch die Standardreaktion

vermieden werden kann; die Merkmalsvektoren dieser Messzeitpunkte werden den OK-Klassen zugewiesen. Die Merkmalsvektoren nach Stoßimpulsen, die den Roboter immer zu Fall bringen, werden den Sturz-Klassen zugeordnet. Die Prototypen werden angelernt, dazu werden allerdings nur die in Abschnitt 7.3 genannten Merkmale und die Schrittphase verwendet.

Mit Hilfe eines MATLAB-Skripts wird die Prototypenklassifikation mit dem Prototypen-Set anhand der an BARt-UH gemessenen Merkmale offline durchgeführt. Als Beispiel ist wieder der schon im vorherigen Abschnitt erwähnte Stoßversuch geprüft worden (die Stoßparameter finden sich in der Bildunterschrift von Abbildung 7.4). Die Abbildungen, die die Klassenprototypenabstände zum aktuellen Merkmalsvektor visualisieren, sind in Abbildung 7.5 zu finden. Hier ist zu sehen, dass nach dem Stoß die Abstände der Prototypen derjenigen Klassen, die nicht die Fallrichtung beschreiben, größer werden. Nach 50 ms hat in beiden Abbildungen eine vor_OK-Klasse den geringsten Abstand und es würde die Standardreaktion ausgeführt werden. Nach einem weiteren Zeitraum, der nun bei beiden unterschiedlich ist, hat eine vor_Sturz-Klasse den geringsten Abstand, es wird der Sturz nach vorne erkannt und die Schutzreaktion würde ausgeführt werden.



(a) Prototypenklassifikation auf Basis der Merkmalsvektoren der Messwerte von BARt-UH. Der Stoß ist bei 1,16 s erfolgt.

(b) Prototypenklassifikation auf Basis der Merkmalsvektoren der Messwerte des Simulators. Der Stoß ist bei 5,35 s erfolgt.

Abbildung 7.5: Abstand der Prototypen, die mit dem Simulator angelernt worden sind, zu dem jeweiligen Merkmalsvektor. Die Simulation ist dem Fallen von BARt-UH nachgestellt worden.

Auch die normalen Laufbewegungen werden richtig mit den normal-Klassen erkannt. Die Prototypenklassifikation ist also mit dem im Simulator angelernten Prototypen-Set auf den BARt-UH übertragbar.

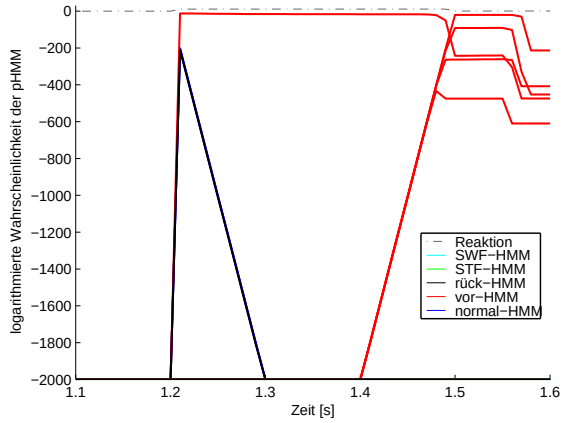
7.5 Transfer der Hidden Markov Models auf BARt-UH

In diesem Abschnitt wird untersucht, ob – wie in Abschnitt 7.4 – die Hidden Markov Models auf dem Simulator angelernt und auf die Messwerte des BARt-UH angewendet werden können.

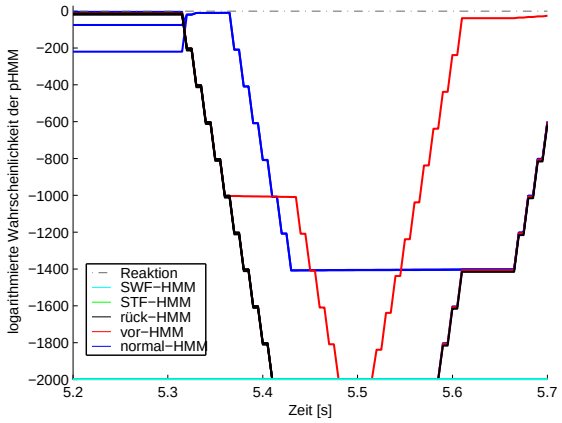
7.5.1 Klassifikation mit Prototyp-Hidden Markov Models

Die $_{\text{p}}$HMMK ist wie in Abschnitt 4.4 angelernt worden. Das in Abschnitt 7.4 erstellte Prototypen-Set ist als Grundlage für die Beobachtungsgenerierung benutzt worden. In Abbildung 7.6 sind die berechneten Wahrscheinlichkeiten der $_{\text{p}}$HMMK auf Basis der Messwerte von BARt-UH und der Messungen des Simulators aufgetragen. Die Verläufe der Wahrscheinlichkeiten der HMM der $_{\text{p}}$HMMK sind zwar durchaus unterschiedlich, aber 50 ms nach dem Stoß fallen in beiden Abbildungen die Wahrscheinlichkeiten derjenigen HMM der $_{\text{p}}$HMMK, die die Fallrichtung nicht beschreiben, rapide ab. In Abbildung 7.6(a) ist zu sehen, dass sofort nach dem Stoß ein vor-HMM der $_{\text{p}}$HMMK das wahrscheinlichste ist und auch für den weiteren Verlauf bleibt. In Abbildung 7.6(b) hingegen ist ein vor-HMM der $_{\text{p}}$HMMK zwar kurz nach dem Stoß das wahrscheinlichste, wird dann aber wegen einer nicht so angelernten Beobachtungssequenz 10 ms nach dem Stoß von einem normal-HMM der $_{\text{p}}$HMMK abgelöst wird, auch wenn es 20 ms nach dem Stoß wieder das wahrscheinlichste HMM der $_{\text{p}}$HMMK wird.

Die Klassifikation $_{\text{p}}$HMMK funktioniert in diesem Beispiel mit den Messwerten von Bart-UH, hat aber in der Simulation in einem kurzen Zeitraum nach dem Stoß eine Fehlklassifikation. Bei beiden $_{\text{p}}$HMMK ist ersichtlich, dass ein vor-HMM der $_{\text{p}}$HMMK 50 ms nach dem Stoß das wahrscheinlichste HMM der $_{\text{p}}$HMMK ist und auch die entsprechende Standardreaktion ausgeführt werden würde. Bei einigen anderen Messungen kommt es jedoch zu Fehlklassifikationen.



(a) Berechnung der Wahrscheinlichkeiten der HMM der ${}_{\mathrm{p}}$HMMK auf Basis der Messwerte von BARt-UH. Der Stoß ist bei 1,16 s erfolgt.



(b) Berechnung der Wahrscheinlichkeiten der HMM der ${}_{\mathrm{p}}$HMMK auf Basis der Messwerte des Simulators. Der Stoß ist bei 5,35 s erfolgt.

Abbildung 7.6: Logarithmierte Wahrscheinlichkeiten der HMM der ${}_{\mathrm{p}}\mathrm{HMMK}_3^{\mathrm{ns}}$, die mit dem Simulator angelernt worden sind. Die Simulation ist dem Stoßversuch mit BARt-UH nachgestellt worden.

7.5.2 Klassifikation mit Richtungs-Hidden Markov Models

Die ${}_{\mathrm{r}}$HMMK ist auf die gleiche Weise wie die ${}_{\mathrm{p}}$HMMK mit dem simulierten Roboter und den daraus gewonnenen Beobachtungen angelernt worden. Beim Transfer auf die Messwerte des BARt-UH haben sich allerdings viele Fehlklassifikationen ergeben. So sind beispielsweise selbst während des normalen Laufens Fall-Bewegungen klassifiziert worden. Der Grund hierfür liegt in der Beobachtungsgenerierung (Unterabschnitt 4.3.2), die bei ähnlichen Sturzszenarien von BARt-UH und dem simulierten Roboter unterschiedliche Beobachtungen generiert. Zunächst mussten Schwellwerte angepasst werden, die zwischen Beobachtungen für normales Gehen und Beobachtungen für das Fallen unterscheiden. Die Polardiagramme ergaben verschobene Winkel und Beträge für die Stürze (Abbildung 7.7). Die liegt daran, dass die Beobachtungsgenerierung mit den entsprechenden Schwellwerten zu stark auf den simulierten Roboter spezialisiert ist.

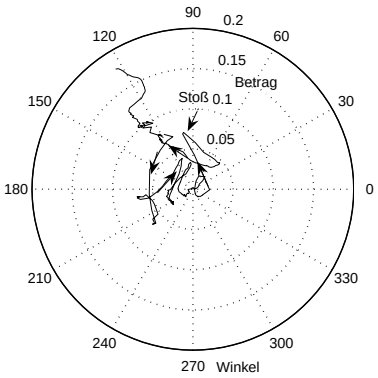
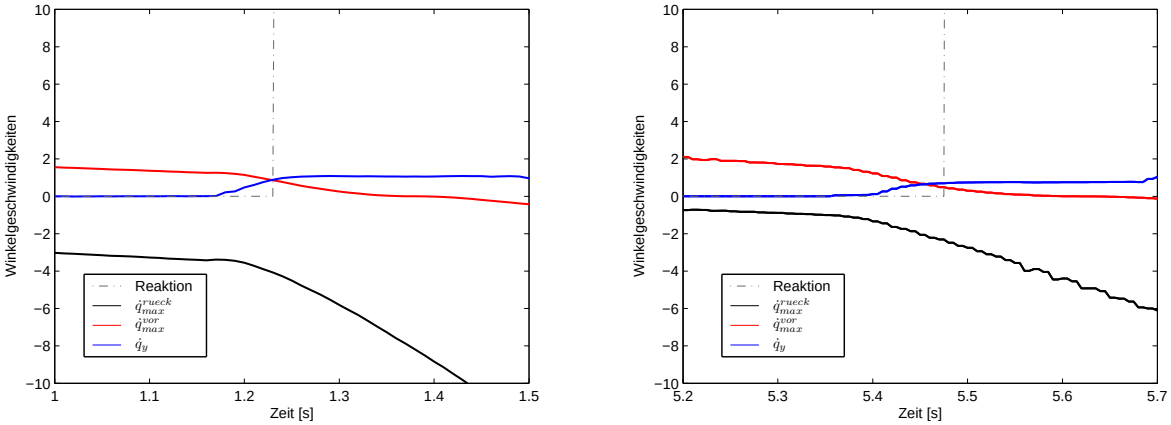


Abbildung 7.7: Polardiagramm mit Winkel und Betrag der für die $_{\text{r}}$HMMK erzeugten Beobachtungen. Die $_{\text{r}}$HMMK sind mit dem simulierten Roboter angelernt worden. Hier zu sehen sind die Beobachtungen auf Basis der Messwerte von BARt-UH nach einem Stoß von hinten. Der Winkel nach dem Stoß sollte ungefährt 0° sein.

7.6 Transfer der Stabilitätsaussagen anhand der Fuß-Kippbewegung auf BARt-UH

In Kapitel 5 ist ein Verfahren zur Stabilitätsaussage mit den Fußkippwinkelgeschwindigkeiten $\dot{q}_{x,y}$ und den maximal erlaubten Fußkippwinkelgeschwindigkeiten $\dot{q}_{max}^{vor,rueck,STF,SWF}$ vorgestellt worden. Dabei wird die Fallrichtung dadurch bestimmt, welche $\dot{q}_{max}^{vor,rueck,STF,SWF}$ von $\dot{q}_{x,y}$ übertroffen wird. Die Unterscheidung zwischen einer Fallbewegung, die mit der Standardreaktion noch vermeidbar wäre, und einer nicht mehr vermeidbaren Fallbewegung wird mit der Größe des Abstandes von $\dot{q}_{max}^{vor,rueck,STF,SWF}$ und $\dot{q}_{x,y}$ durchgeführt.

Es haben sich beim Vergleichen der Klassifikation mit dem unmodifizierten und dem modifizierten simulieren Robotermodells sowie den Messwerten von BARt-UH gute und sehr ähnliche Verläufe der Fußkippwikelgeschwindigkeiten ergeben. In Abbildung 7.8 sind die zwei sehr ähnlichen Verläufe der Fußkippwinkelgeschwindigkeiten nach einem Stoß von hinten an den Roboter dargestellt. Abbildung 7.8(a) zeigt $\dot{q}$ anhand der Messungen von BARt-UH, dass sofort 10 ms nach dem Stoß $\dot{q}_y$ ansteigt und 70 ms nach dem Stoß $\dot{q}_{max}^{vor}$ übersteigt. In Abbildung 7.8(b) sind die Messungen des Simulators mit dem modifizierten Robotermodell zu finden. Hier ist nach 100 ms $\dot{q}_y > \dot{q}_{max}^{vor}$.



(a) Winkelgeschwindigkeiten auf Basis der Merkmalsvektoren der Messwerte von BARt-UH. Der Stoß ist bei 1,16 s erfolgt.

(b) Winkelgeschwindigkeiten auf Basis der Merkmalsvektoren der Messwerte des Simulators. Der Stoß ist bei 5,35 s erfolgt.

Abbildung 7.8: Diese Abbildungen zeigen die maximal berechneten Winkelgeschwindigkeiten, die mit dem Simulator angelernt worden sind. Die Simulation ist dem Fallen des BARt-UH nachgestellt worden.

8 Zusammenfassung

Zweibeinige Roboter können durch Störungen – wie zum Beispiel durch Stöße – leicht zu Fall gebracht werden. Eine schnelle und zuverlässige Klassifikation des Zustands eines Roboters ermöglicht es, eine Anzahl von Stürzen durch eine rechtzeitig ausgeführte Reaktion zu vermeiden. In dieser Arbeit ist untersucht worden, inwiefern sich verschiedene Mustererkennungsverfahren zur rechtzeitigen Erkennung und Beurteilung von Stürzen zweibeiniger Roboter eigenen.

Mit Hilfe eines Roboter-Simulators sind unterschiedliche Sturzszenarien analysiert worden, die durch Stöße gegen den Roboter verursacht wurden. Dabei hat sich gezeigt, dass Stoßrichtung, Stoßzeitpunkt und Stoßstärke einen großen Einfluss sowohl auf den Sturzverlauf als auch eine mögliche Vermeidung durch eine Reaktion haben. Für jede Sturzrichtung ist eine zuvor vorgegebene Reaktion festgesetzt worden. Zuerst ist mit Reaktionen, die ohne Mustererkennungsverfahren von dem simulierten Roboter ausgeführt worden sind, festgestellt worden, welche Stürze überhaupt vermeidbar sind. In weiteren Simulationen werden Mustererkennungsverfahren eingesetzt, die entscheiden sollen, wann und welche Reaktion auszuführen ist. Es sind im Verlauf der Arbeit drei Mustererkennungsverfahren implementiert worden:

- Die **Prototypenklassifikation** (Kapitel 3) berechnet den Abstand des aktuellen Roboterzustands zu vorher angelernten Klassen, die eine bestimmte Bedeutung wie beispielsweise *Sturz nach vorne* haben. Die Prototypen bestehen aus Sensorinformationen wie zum Beispiel Winkeln und Geschwindigkeiten des Torsos und des Standfußes.
- Die **Hidden Markov Models** (Kapitel 4) entstammen dem Gebiet der Spracherkennung. Sie beschreiben stochastische Modelle, in denen Beobachtungssequenzen Zuständen mit einer bestimmten Wahrscheinlichkeit zugeordnet werden können. Durch das Wechseln von einem Zustand in einen anderen können in der Spracherkennung Worte beschrieben werden. In dieser Arbeit sind einzelne Fallsituationen als

Zustände modelliert worden. Eine vorgeschaltete Prototypenklassifikation erzeugt die Beobachtungen.

- Bei der **Stabilitätsaussage anhand der Fuß-Kippbewegung** (Kapitel 5) wird eine maximal für die Stabilität erlaubte Fußkippwinkelgeschwindigkeit über die Fußkanten in Abhängigkeit des aktuellen Fußkippwinkels und der Lage des Schwerpunkts errechnet. Die Kippbewegung über eine der Fußkanten ist mit dem Modell eines inversen Pendels angenähert worden.

Für den Vergleich der Mustererkennungsverfahren (Kapitel 6) sind verschiedene Vergleichsgrößen wie eine Sturzvermeidungs- und eine Robustheitsgröße definiert worden (Abschnitt 6.2). Die Sturzvermeidungsgröße bewertet, wie viele Stürze vermieden werden können und welche Reaktionen der Roboter dabei ausgeführt hat. In der Robustheitsgröße wird das Augenmerk darauf gerichtet, dass der Roboter in bestimmten Situationen eine höhere Eigenstabilität besitzt und nicht immer eine Reaktion notwendig ist, um einen Sturz zu vermeiden.

Es hat sich ergeben, dass alle Mustererkennungsverfahren Stürze erkennen können (Abschnitt 6.5). Dabei haben die Hidden Markov Models eine sehr gute Performance bei der Sturzvermeidung gezeigt. Allerdings ist ihre Robustheit etwas schlechter als die der Prototypenklassifikation. Die Stabilitätsaussage anhand der Fuß-Kippbewegung sticht in keiner Bewertung positiv oder negativ hervor.

Die Berechnungen der Hidden Markov Models sind allerdings komplexer und benötigen ein Vielfaches der Rechenzeit gegenüber der Prototypenklassifikation und der Stabilitätsaussage anhand der Fuß-Kippbewegung (Tabelle 6.1). Weiterhin hat es sich ergeben, dass die Hidden Markov Models anfälliger gegenüber leicht veränderten Situationen sind.

Das Übertragen der mit dem simulierten Roboter angelernten Mustererkennungsverfahren auf den realen Roboter funktioniert prinzipiell gut. Sowohl die Prototypenklassifikation (Abschnitt 7.4) als auch die Stabilitätsaussage anhand der Fuß-Kippbewegung (Abschnitt 7.6) liefern sehr gute Werte mit wenigen Fehlklassifikationen. Die Klassifikation mit den Prototyp-Hidden Markov Models funktioniert bei mehreren Stürzen (Unterabschnitt 7.5.1). Bei den Richtungs-Hidden Markov Models hat sich eine Vielzahl von

Fehlklassifikationen ergeben (Unterabschnitt 7.5.2). Die Ursache ist die Beobachtungsgenerierung, die zu sehr auf den simulierten Roboter spezialisiert ist und beim realen Roboter BARt-UH andere Beobachtungen erzeugt.

Zusammenfassend ist zu bemerken, dass das Anwenden der Klassifikation mit den Hidden Markov Models recht paramterabhängig und rechenintensiv ist. Die Anwendung der anderen Mustererkennungsverfahren erscheint wesentlich robuster und benötigt weniger Rechenzeit.

8.1 Ausblick

Der Transfer der mit dem simulierten Roboter angelernten Klassifikation mit Hilfe der Richtungs-Hidden Markov Models ließ sich nicht auf die Messungen von BARt-UH übertragen (Unterabschnitt 7.5.2). Die Beobachtungsgenerierung müsste so modifiziert werden, dass in der vorgeschalteteten Prototypenklassifikation nicht alle Klassen zur Beobachtungsgenerierung beitragen, sondern nur die mit den geringsten Abständen zum aktuellen Merkmalsvektor. Außerdem muss eine differenziertere Kombination der Schwellwerte erstellt werden, die die Winkel und Betrags-Informationen in skalare Beobachtungen umwandelt.

Die Vermeidung von Stürzen besteht aus zwei Aufgabengebieten. Einerseits die in dieser Arbeit untersuchten Mustererkennungsverfahren und andererseits die Reaktionen, die der Roboter ausführt. In dieser Arbeit stand für jede Fallrichtung nur eine Reaktion zur Verfügung. Mit Reaktionen, die während des Fallens online an die Situation angepasst werden können, wären wahrscheinlich eine größere Anzahl von Stürzen vermeidbar.

Die Klassifikation mit den Hidden Markov Models liefert bei dem simulierten Roboter größtenteils bessere Ergebnisse als die Prototypenklassifikation und die Stabilitätsaussage anhand der Fuß-Kippbewegung. Allerdings ist eine Sturzklassifikation mit den Hidden Markov Models sehr anfällig gegenüber Beobachtungen, die nicht angelernt worden sind. Die Ersatzwahrscheinlichkeiten (Abschnitt 4.4) hierfür waren vermutlich bei den Untersuchungen zu niedrig angesetzt. Durch ein Vergrößern der Ersatzwahrscheinlichkeiten

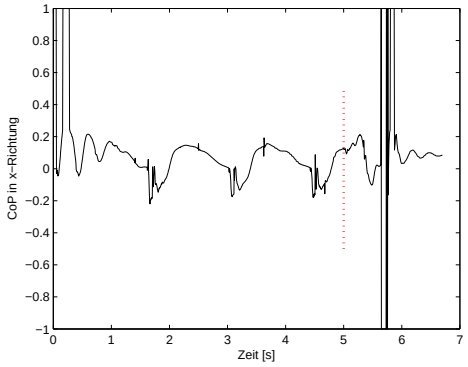
könnte diese Anfälligkeit verringert werden, so dass auch nicht antrainierte Beobachtungen die Gesamtwahrscheinlichkeit eines Hidden Markov Models nicht zu sehr erniedrigt.

Die Hidden Markov Models haben die Eigenschaft, zeitliche Zusammenhänge zu beschreiben. Mehrere zeitlich versetzte Prototypenklassifikationen könnten diese Eigenschaft nachahmen. Eine Prototypenklassifikation könnte während des Stoßes durchgeführt werden, eine kurz nach dem Stoß und eine weitere nach einem etwas längeren Zeitraum. Hierfür müsste allerdings für jeden Zeitpunkt ein unterschiedliches Prototypen-Set angelernt werden. Mit der Kombination dieser Ergebnisse ist es denkbar, eine bessere Gesamtklassifikation zu erreichen. Eventuell könnte hierbei auch die Stabilitätsaussage anhand der Fuß-Kippbewegung berücksichtigt werden.

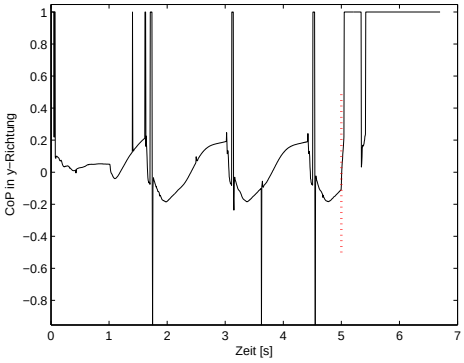
BARt-UH und der simulierte Roboter haben nur zwei Beine, die an dem Torso angebracht sind. Ein Roboter, der einem Menschen mit einem Oberkörper und zwei Armen ähnlicher ist, könnte mit dem entsprechenden Zuwachs an Freiheitsgraden eine Vielzahl von Reaktionen ausführen. Mit dem Oberkörper und den Armen wäre es ihm möglich, bei nicht stark ausgeprägten Fallbewegungen mit der Verlagerung seines Schwerpunktes ein Fallen zu verhindern. Ist der Stoß zu stark und der Roboter stürzt, könnte er eine Abrollbewegung durchzuführen, um damit die Stärke des Aufpralls auf den Boden zu minimieren.

A Anhang

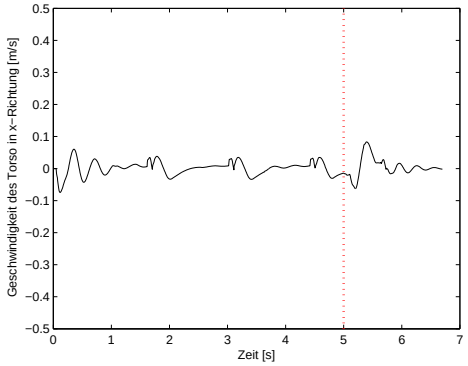
A.1 Plots der Merkmale



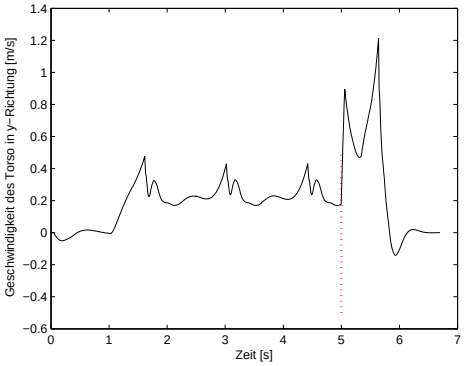
(a) Center of Pressure in x-Richtung



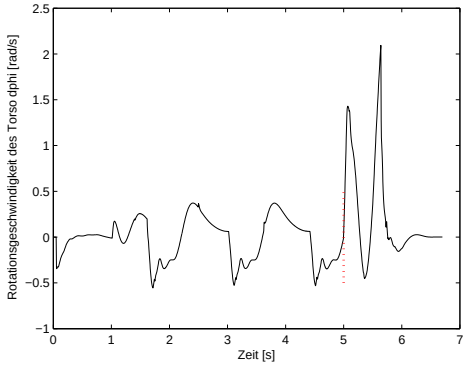
(b) Center of Pressure in y-Richtung



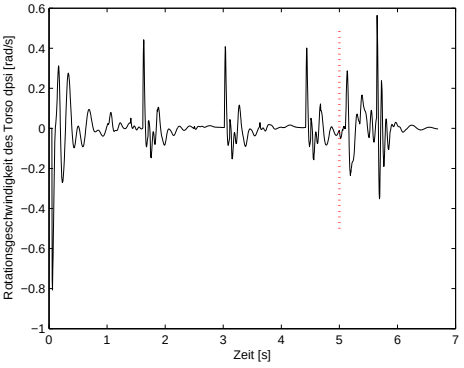
(c) Torsogeschwindigleit in x-Richtung



(d) Torsogeschwindigleit in y-Richtung

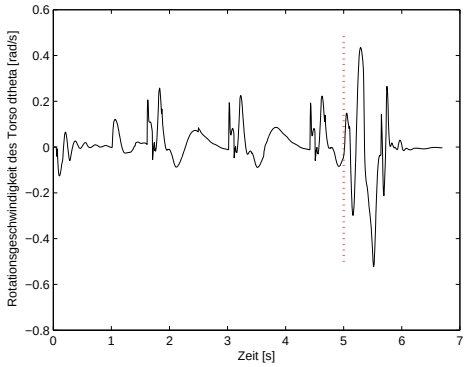


(e) Torsorotationsgeschwindigkeit $d\phi$

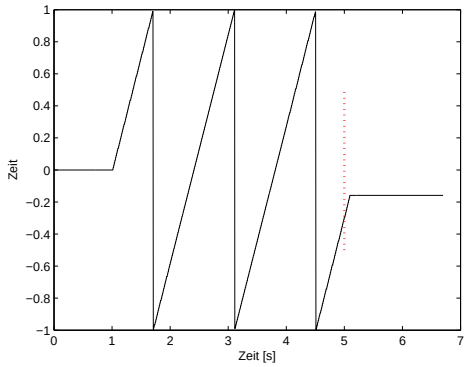


(f) Torsorotationsgeschwindigkeit $d\psi$

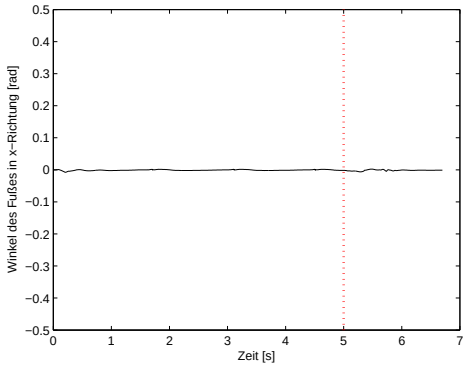
Abbildung A.1: Plots der Merkmale nach einem Stoß (rote gepunktete Linie) an den simulierten Roboter mit einem Impuls von 9 Ns von hinten zum Zeitpunkt 5 s; da der Roboter reagiert, bewegt sich der Schwingfuß nach vorne und berührt bei 5,65 s wieder den Boden. Eine normale Schrittperiode geht von $t = 3{,}1\,s$ ($t_p = -1$) bis $t = 4{,}5\,s$ ($t_p = 1$).



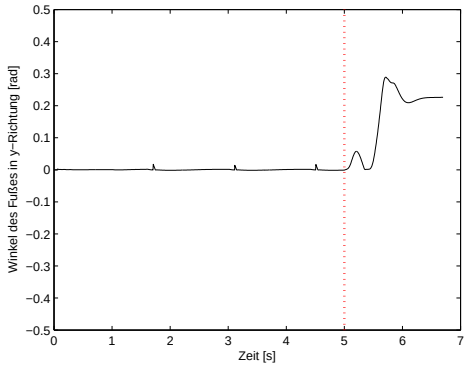
(a) Torsorotationsgeschwindigkeit $d\eta$



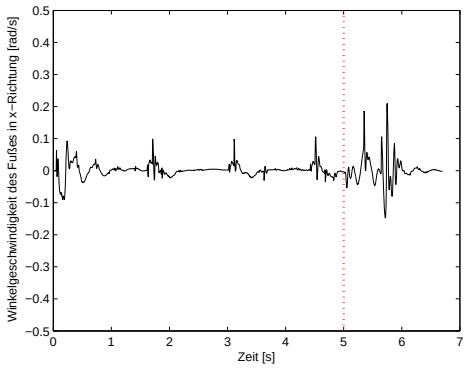
(b) Schrittphase



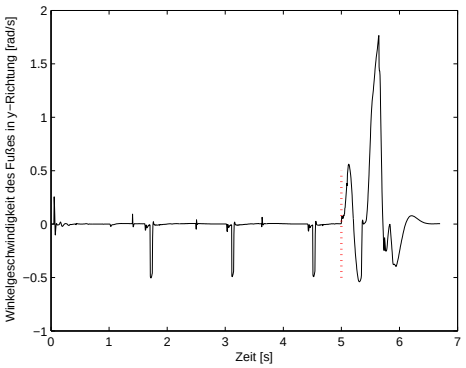
(c) Winkel des Fußes in x-Richtung



(d) Winkel des Fußes in y-Richtung



(e) Winkelgeschwindigkeit des Fußes in x-Richtung



(f) Winkelgeschwindigkeit des Fußes in y-Richtung

Abbildung A.2: Plots der Merkmale nach einem Stoß (rote gepunktete Linie) an den simulierten Roboter mit einem Impuls von 9 Ns von hinten zum Zeitpunkt 5 s; da der Roboter reagiert, bewegt sich der Schwingfuß nach vorne und berührt bei 5,65 s wieder den Boden. Eine normale Schrittperiode geht von $t = 3{,}1\,s$ ($t_p = -1$) bis $t = 4{,}5\,s$ ($t_p = 1$).

Literaturverzeichnis

[Alb02] Albert, Amos: *Intelligente Bahnplanung und Regelung für einen autonomen zweibeinigen Roboter*, Universität Hannover, Diss., 2002

[BHP01] Besdo, Dieter; Heimann, Bodo; Popp, Karl: *Formelsammlung zur Technischen Mechanik I - IV*. 2001

[BPSW70] Baum, L. E.; Petrie, T.; Soules, G.; Weiss, N.: A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. In: *Ann. Math. Stat.* 41 (1970), S. 164–171

[HGG05] Höhn, Oliver; Gačnik, Jan; Gerth, Wilfried: Detection and Classification of Posture Instabilities of Bipedal Robots. In: *Proceedings of the 8th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2005)*, 2005

[HS99] Hofschulte, Jens; Schermeier, Olaf: *Aufbau und Inbetriebnahme eines zweibeinigen Roboters*, Universität Hannover, Diplomarbeit, 1999. – Diplomarbeit (Gruppenarbeit) unveröffentlicht

[HSG04] Höhn, Oliver; Schollmeyer, Michael; Gerth, Wilfried: Sturzvermeidung von zweibeinigen Robotern durch reflexartige Reaktionen. In: *Eingebettete Systeme*, 2004, S. 60–69

[LE89] Liedtke, Claus-E.; Ender, Manfred: *Wissensbasierte Bildverarbeitung*. Springer Verlag, 1989

[MS01] Manning, Christopher D.; Schütze, Hinrich: *Foundations of Statistical Natural Language Processing*. The MIT Press, 2001

[Rab89] Rabiner, Lawrence R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Proceedings of the IEEE* (1989), Februar, S. 257–286

[See03] Seebode, Mario: *Entwicklung, Aufbau und Inbetriebnahme eines künstlichen Gleichgewichtsorgans für den Einsatz auf autonomen Servicerobotern*, Universität Hannover, Diplomarbeit, 2003. – Diplomarbeit unveröffentlicht