Diplomarbeit
Inertialstabilisierung für externe Sensoren

Bearbeitet von: Daniel Dönigus
Leitung: Prof. Dr. Oskar von Stryk
Betreuer: Martin Friedmann

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 29. Februar 2008

Vorname Nachname

Kurzzusammenfassung

Bei der Verwendung von Digitalkameras auf mobilen und insbesondere bei laufenden Robotern kommt es durch die Bewegungen des Roboters oft zu Verwacklungen. Hieraus können verschwommene Bilder resultieren, die eine weitere Auswertung erschweren. Es ist teilweise auch wünschenswert, dass der Blick während der Bewegung des Körpers in eine Richtung gehalten wird.

In dieser Arbeit wird ein System entwickelt, mit dem man den Endeffektor einer kinematischen Kette hinsichtlich seiner Orientierung stabilisieren kann. Diese Stabilisierung geschieht unabhängig von etwaigen externen Sensoren, die am Endeffektor angebracht sind. Es kommen Inertialsensoren zum Einsatz, die eine besonders günstige Berechnung der Stabilisierung ermöglichen. Es ist möglich, diese Berechnungen auf einem Mikrocontroller durchzuführen. Gleichzeitig ist das System modular aufgebaut und kann einfach an verschiedene kinematische Modelle und Hardware angepasst werden. Nach der Entwicklung wurde das System evaluiert. Die Ergebnisse dieser Evaluierung werden ebenfalls in dieser Arbeit vorgestellt.

Abstract

While using digital cameras on mobile and especially with walking robots there is often shaking because of the robot's motion. The result of this could be blurred images which makes it hard to evaluate them. Partially it is desirable to hold the gaze, while the body is moving.

In this work a system was developed which allows to stabilize an endeffector of a kinematic chain with respect to its orientation was developed. This stabilization is done independently of an external sensor that is possibly installed on the endeffector. The use of intertial sensors allows a lightweight calculation of the stabilization. It is possible to do these calculations on a microcontroller. At the same time the system is of a modular design so that it can be easily adapted to different kinematic models or hardware. The System was evaluated after the development. The results of this evaluation are also presented in this thesis.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Hintergrund und Motivation 1
1.2 Aufgabenstellung 2
1.3 Übersicht 4

2 Stand der Forschung **7**

2.1 Forschung am Fachbereich 7
2.2 Stabilisierte Sensoren 8
2.3 Animate Vision 8
2.4 Inertialsensoren als Gravitationsreferenz 9
2.5 Vestibulookulärer Reflex 10
2.6 Stand der Technik 10
2.6.1 Bildstabilisierung 11
2.6.2 Inertialsensoren 13

3 Grundlagen (Theorie) **15**

3.1 Transformationen 15
3.1.1 Rotationen 18
3.1.2 Elementare Rotationen 18
3.1.3 Eigenschaften von Rotationsmatrizen 18
3.1.4 Verkettungen von Rotationen 19
3.1.5 Allgemeine Rotationsmatrizen 20
3.1.6 Drehachse und Drehwinkel bestimmen 20
3.2 Winkelgeschwindigkeiten 20
3.2.1 Transformation in andere Koordinatensysteme 21
3.2.2 Addition von Winkelgeschwindigkeiten 22
3.2.3 Abhängigkeit von Winkelgeschwindigkeiten 22
3.3 Jacobi-Matrix 23
3.4 Jacobi-Modell 24
3.4.1 Inverses Jacobi-Modell 24

4 Grundlagen (Hardware- und Softwaretechnik) 27
4.1 Sensoren . . . 27
4.1.1 Gyroskope . . . 27
4.1.2 Beschleunigungssensoren . . . 29
4.2 Mikrocontroller . . . 30
4.2.1 Interrupts . . . 30
4.2.2 Timer . . . 30
4.2.3 A/D-Wandler . . . 31
4.2.4 UART . . . 32
4.2.5 Half Duplex UART . . . 32

5 Konzept 35
5.1 Datenfluss zwischen den Komponenten . . . 35
5.2 Stabilisierungsprogramm . . . 36
5.2.1 Initialisierungsphase . . . 38
5.2.2 Stabilisierungsphase . . . 38
5.2.3 Transformation der Sensordaten . . . 39
5.2.4 Redundanzen entfernen . . . 40
5.2.5 Transformation der Drehachsen . . . 41
5.2.6 Inverses Jacobimodell . . . 41
5.2.7 Driftkompensation und Richtungsänderung . . . 42
5.3 Konfiguration des kinematischen Modells . . . 43
5.4 Modularität . . . 44
5.5 Testsystem und Evaluierung . . . 44

6 Realisierung 47
6.1 Testsystem . . . 47
6.1.1 Verwendete Hardware . . . 48
6.1.2 Schnittstellen . . . 51
6.1.3 Kommunikation zu Host-System . . . 53
6.2 Stabilisierungsprogramm . . . 53
6.2.1 Programmablauf . . . 54
6.2.2 Initialisierung . . . 54
6.2.3 Stabilisierungsschleife . . . 58
6.2.4 Datenstrukturen . . . 59
6.3 Softwaremodularität . . . 61
6.4 Zusammenspiel und Abhängigkeiten der einzelnen Module . . . 62
6.5 Prozeduraler Aufbau des Stabilisierungsprogramms . . . 63
6.6 Integration in ein bestehendes Gesamtsystem . . . 64
6.7 Konfigurationswerkzeug . . . 65
6.7.1 Bedienung . . . 66

6.7.2 Elementtypen . . . 66
6.7.3 Dateiformat . . . 67
6.7.4 Codegenerierung . . . 67
6.7.5 Kommandozeile . . . 70

7 Experimentelle Erprobung **71**
7.1 Erweitertes Testsytem . . . 71
7.2 Messung des Sensordrifts . . . 72
7.3 Rückstellgenauigkeit . . . 75
7.3.1 Störungen durch laufende Servomotoren . . . 76
7.3.2 Nichtlinearität der Eingangssignale . . . 79
7.4 Messung der Rückstellgeschwindigkeit . . . 81
7.5 Aufnahmen eines stabilisierten externen Sensors . . . 84

8 Zusammenfassung und Ausblick **89**
8.1 Beschreibung der Entwicklung . . . 89
8.2 Evaluierung . . . 90
8.3 Ausblick . . . 91
8.3.1 Langzeitstabilisierung . . . 91
8.3.2 Lagestabilisierung . . . 91

A Installation Konfigurationstool (Windows) **93**

B Driftmessungen **95**

C Messungen der Rückstellgenauigkeit **101**

D Messungen während der Stabilisierung **105**

E Nötige Funktionsdeklarationen für die Schnittstellen **111**

F Skizzen und Fotos des Stabilisierungssystems **115**

Abbildungsverzeichnis

3.1 Transformationen zwischen Koordinatensystemen 15
3.2 Schaubild zur Veranschaulichung der Notation 16
3.3 Elementare Rotationen 18
3.4 Radius und Winkelgeschwindigkeit 21
3.5 Gelenkstellungsbedingte Abhängigkeit von Sensoren 22

4.1 Aufbau eines mechanischen Gyroskops 28

5.1 Blockschaltbild des Gesamtsystems 36
5.2 Zustandsübergangsgraph für das Stabilisierungsprogramm 37
5.3 Im Programm hinterlegte Modelle / Module 45

6.1 Skizze des Testsystems 48
6.2 Schnittstellen der einzelnen Hardwarekomponenten 51
6.3 Programmablaufplan - Hauptschleife 55
6.4 Programmablaufplan - Initialisierung 57
6.5 Programmablaufplan - Stabilisierung 60
6.6 Abhängigkeiten zwischen den Modulen 62
6.7 Abhängigkeiten zwischen den Funktionen 63
6.8 Von der Funktion stabilize() benötigte Funktionen 64
6.9 Konfigurationswerkzeug zur Anpassung des kinematischen Modells 65

7.1 Skizze des erweiterten Testsystems 72
7.2 Fehler durch Drift bei unterschiedlichen Initialisierungseinstellungen 74
7.3 Gemessener Driftfehler (Gyroskop 1) 75
7.4 Gemessener Driftfehler (Gyroskop 2) 76
7.5 Gemessener Driftfehler (Gyroskop 3) 77
7.6 Driftfehler bei laufenden Servos 78
7.7 Fehler des A/D-Wandlers 80
7.8 Servobewegungen während der Stabilisierung, Weg: 0/0/0 → 30/0/0 → 0/0/0 82
7.9 Servobewegungen während der Stabilisierung, Weg: 0/0/0 → 30/0/0 84
7.10 Servobewegungen während der Stabilisierung, Weg: 0/0/0 → 5/0/5 → 0/0/0 → 85

7.11 Servobewegungen während der Stabilisierung, Weg: 0/0/0 → 30/0/30 → 0/0/0 85
7.12 Bilderreihe eines externen Sensors ohne Stabilisierung 86
7.13 Bilderreihe eines externen Sensors ohne Stabilisierung 87

A.1 Dialog “Umgebungsvariablen” 94
A.2 Dialog zur Änderung einer Umgebungsvariablen 94

F.1 Foto des Testsystems 115
F.2 Frontansicht des Testsystems 115
F.3 Seitenansicht des Testsystems 116
F.4 Draufsicht des Testsystems 116
F.5 Foto des erweiterten Testsystems 117
F.6 Seitenansicht des erweiterten Testsystems 117
F.7 Seitenansicht des erweiterten Testsystems 118
F.8 Draufsicht des erweiterten Testsystems 118

Tabellenverzeichnis

6.1 Code-Dateien für alle benutzten Hardwarekomponenten 48
6.2 Anschlusseinstellungen Mikrocontroller-Host 52
6.3 Anschlusseinstellungen Mikrocontroller-Servos 52

E.1 Funktionen des Mikrocontroller-Moduls 111
E.2 Funktionen des Servo-Moduls 112
E.3 Funktionen des Gyroskop-Moduls 113

1 Einleitung

1.1 Hintergrund und Motivation

Das Fachgebiet Fachgebiet Simulation, Systemoptimierug und Robotik (SIM) des Fachbereichs Informatik der TU Darmstadt beschäftigt sich unter anderem mit der Entwicklung verschiedener autonomer mobiler Roboterplattformen. Beispiele solcher Entwicklungen sind Roboter für die Teilnahme in den verschiedenen Ligen des Roboterfußballs oder der kooperative Einsatz verschiedener autonomer Fahrzeuge.

Bei Robotern und autonomen Systemen sind die Sensoren während der Bewegung des Systems sehr oft Drehbewegungen um verschiedene Achsen ausgesetzt. Diese Bewegungen führen zu einer Veränderung der Orientierung des Sensorkoordinatensystems bezüglich der Orientierung des Weltkoordinatensystems. Bei der Nutzung der Sensordaten muss die veränderte Orientierung durch eine Transformation der Sensordaten stets wieder rückgerechnet werden.

Wie Dana H. Ballard in seinen Veröffentlichungen zu Animate Vision [1, 2] schreibt, ist es im Bereich des Computersehens günstiger und natürlicher, ein System zur automatischen Blickkontrolle zu entwickeln. Die Vorteile eines solchen Systems sind nach Ballard der größere Sichtbereich durch eine bewegliche Kamera, die Fixierung von Objekten, während sich die Umwelt eventuell in Bewegung befindet oder programmierte Kamerafahrten, die mit einem fixen Kamerasystem nicht möglich wären. In den folgenden Jahren wurden mehrere Systeme zur Blickrichtungssteuerung entwickelt, die basierend auf den Bildern von Digitalkameras und neuronalen Netzen selbständig die Blickrichtung stabilisieren.

Diese Form der Stabilierung setzt ein häufiges Auswerten der Kamerabilder voraus, was wiederum zu einer Belastung der Rechenkapazität führt. Im Fall mobiler Roboter ist aber gerade diese Rechenkapazität sehr begrenzt. Desweiteren kann die Stabilisierung mittels Kamerabildern nicht auf alle anderen Sensortypen, wie zum Beispiel Laserscanner oder Ultraschallsensoren übertragen werden, so dass hier ein anderer Ansatz gewählt werden muss. Im Idealfall sollte die komplette Stabilisierung auf einem Mi-

krocontroller (μC) durchgeführt werden können. Diese Controller sind bereits für Steuerungsaufgaben der Aktoren in einigen Robotersystemen im Einsatz.

Das Auslesen und Auswerten digitaler und analoger Signale kann auf einem Mikrocontroller mit vergleichsweise sehr geringem Aufwand gegenüber dem Auswerten von Kamerabildern geschehen. Inertialsensoren, die solche Signale ausgeben, sind darüber hinaus in den letzten Jahren sehr preisgünstig geworden. Somit liegt es nahe, solche Systeme zu verwenden, um eine Stabilisierung von externen Sensoren zu ermöglichen. Gerade beim Einsatz von Kamerasystemen sollte eine solche Stabilisierung Vorteile gegenüber einem unstabilisierten System haben. Die Bilder einer verwackelten Kamera können entweder stark unscharf sein, oder im Falle des Zeilensprungverfahrens Verschiebungen zwischen jeweils zwei Zeilen aufweisen. Eine genaue Objekterkennung würde dadurch erschwert. In jedem Fall müsste eine Vorverarbeitung stattfinden, um die Kamerabilder verwenden zu können.

Ein weiterer Vorteil der Stabilisierung mittels Inertialsensoren ist die Tatsache, dass mit derartigen Systemen jede Art von externem Sensor stabilisiert werden kann. Man ist nicht auf die Stabilisierung von Digitalkameras festgelegt. Es können auch Laserscanner oder Ultraschallsensoren mit einem solchen System stabilisiert werden.

Um eine Stabilisierung mit Hilfe von Inertialsensoren durchzuführen, müssen zum einen die Sensorwerte aus den Sensoren ausgelesen werden. Danach müssen diese Werte für die weiteren Verarbeitungen in einen gängigen Maßstab umgerechnet werden. Im Anschluß daran wird die eigentliche Stabilisierung in Form von einigen geometrischen Berechnungen durchgeführt und zum Schluß werden die nötigen Sollpositionen und -geschwindigkeiten an die Aktoren gesendet. Diese leichtgewichtigen Rechenaufgaben sollten durch einen Mikrocontroller durchgeführt werden können. Es wird kein Personal Computer (PC) nötig, um diese Aufgaben durchzuführen. Demnach ist es möglich, diese Art der Stabilisierung auch auf recht kleinen, mobilen Systemen mit wenig Rechenleistung und Energiekapazität einzusetzen.

1.2 Aufgabenstellung

Am Fachgebiet SIM der TU Darmsadt existieren einige mobile Robotersysteme, die teilweise mit verschiedenen Gyroskopsystemen ausgestattet sind. Die Gyroskope werden unter anderem dazu genutzt, ein Kippen des Roboters festzustellen und eine Lagestabilisierung durchzuführen. Hierbei erfolgt keine genaue Berechnung der Lage des Roboterkörpers. Wenn sich der Roboter zu weit nach vorne oder nach hinten neigt, wird er er mit Hilfe von Armbewegungen wieder stabilisiert. Ein weiteres Aufgabengebiet ist die Nutzung von Gyroskopen für die autonome Navigation von radgetriebenen Robotern.

Sowohl die momentan eingesetzten humanoiden und vierbeinigen Roboter als auch Systeme auf der Pioneer II-Plattform bieten die Möglichkeit, den Kopf mit eingebauter Kamera in mindestens einem

Freiheitsgrad zu bewegen. Somit ist es naheliegend, Sensordaten von Gyroskopen zu verwenden, um eine Blickrichtungsstabilisierung zu konzipieren, implementieren und zu evaluieren.

Im Rahmen dieser Diplomarbeit soll ein System entwickelt werden, das die automatische Inertialstabilisierung einer Kamera oder eines anderen externen Sensors am Ende einer kinematischen Kette durchführt. Da verschiedene Robotersysteme des Fachbereichs mit unterschiedlichen Gyroskop- und Servoanordnungen im Einsatz sind, muss bei einem solchen Stabilisierungssystem die Möglichkeit bestehen, diese Anordnungen anpassen zu können. Das Modell der kinematischen Kette sollte mit einem möglichst kleinen Aufwand geändert werden können. Ein Benutzer sollte einfach und intuitiv ein anderes Modell programmieren können. Im Idealfall sollte der Benutzer nicht in den bestehenden Sourcecode des Projektes eingreifen müssen, um diese Änderung des kinematischen Modells durchzuführen.

Bei den verwendeten Sensoren handelt es sich meist um einachsige Gyroskope. Man kann also nicht davon ausgehen, dass die Daten aller Gyroskope bezüglich eines gemeinsamen Koordinatensystems vorhanden sind. Desweiteren müssen die gemessenen Drehraten zweier Gyroskope auch nicht zwingend paarweise rechtwinklig zueinander stehen, was zu redundanten Daten führen kann. Dies muss beim Zusammenfassen der Sensordaten berücksichtigt werden. Es muss eine Vorverarbeitung stattfinden, die alle Daten in ein gemeinsames Bezugskoordinatensystem transformiert und die redundanten Daten herausfiltert.

Mit Hilfe von Gyroskopen kann nur eine kurzfristige Stabilisierung der Daten erfolgen, da sie immer einen Drift über die Zeit aufweisen, falls sie nicht absolut exakt initialisiert wurden oder es eine Verschiebung des Nullpunkts zur Laufzeit gibt. Eine Langzeitstabilisierung allein über Gyroskope ist somit nicht möglich. Außerdem sollen die Bewegungen des Roboterkopfes durch die Stabilisierung nicht beeinträchtigt werden. Es soll zum Beispiel immer noch möglich sein, mit der Kamera im Kopf des Roboters ein sich bewegendes Objekt im Blickfeld zu behalten. Es muss also neben der Stabilisierung auch die Möglichkeit geschaffen werden, von einem externen Host-Rechner eine Sollposition der Servomotoren vorzugeben. Dadurch kann man zum einen mittels Bildverarbeitung die Stabilisierung auf lange Zeit gewährleisten und zum anderen eine Bewegung des Kopfes ermöglichen.

Neben den unterschiedlichen kinematischen Modellen kommen auch komplett unterschiedliche Hardwarekomponenten bei den verwendeten Robotern zum Einsatz. Diese Komponenten müssen dann in der Regel auch durch andere Softwarebefehle angesprochen werden. Um die Arbeit mit minimalem Aufwand auch für andere Hardwarekomponenten verwenden zu können, sollte die komplette Software modular aufgebaut sein. Für jede verwendete Hardwarekomponente sollte ein Modul geschrieben werden, das unter bestimmten Vorgaben für einen bestimmten Hardwaretyp verwendet wird. Sollte später eine Komponente ausgetauscht werden, muss lediglich dieses eine Modul neu programmiert werden.

Im Anschluss an die Entwicklung des Stabilisierungssystems soll dieses evaluiert werden. Mit Hilfe einer Kamera sollen während der Stabilisierung kontinuierlich Bilder aufgenommen werden, die dann

zur Lagebestimmung des Endeffektors in jedem einzelnen Bild herangezogen werden können. Es soll hierbei evaluiert werden, welcher Orientierungsfehler beim Nachstellen auftritt und wie genau das System die ursprüngliche Orientierung halten kann, wenn es sich wieder in Ruhe befindet.

Bei der Evaluierung des Systems muss berücksichtigt werden, dass die Gyroskope einen Drift aufweisen und somit die Güte des Systems nicht direkt gemessen werden kann. Vielmehr muss auch der mittlere Sensordrift gemessen werden und in die Daten einfließen, die bei der Evaluierung gemacht werden.

1.3 Übersicht

Im folgenden Kapitel wird zunächst auf den aktuellen Stand in Forschung und Technik eingegangen. Beginnend mit Forschungen im Gebiet der Inertialstabilisierung am Fachbereich SIM, über verschiedene Veröffentlichungen aus Deutschland in diesem Bereich, wird auf sonstige weltweite Veröffentlichungen eingegangen. Darunter befinden sich auch die Anfänge der Sensorstabilisierung mittels der Kamerabilder selbst. Ein Teilkapitel umschließt zudem das biologische Vorbild eines solchen Stabilisierungssystems und beschäftigt sich dann mit verschiedenen anderen Systemen, die bereits im Alltag benutzt werden.

Im darauf folgenden Kapitel 3 werden die für diese Arbeit nötigen theoretischen Grundlagen beschrieben. Zunächst werden die geometrischen Zusammenhänge in Form von Transformationsmatrizen und deren Eigenschaften dargestellt. Danach folgt ein Teil, der sich mit den kinematischen Grundlagen beschäftigt. Hier wird auf Drehwinkelgeschwindigkeiten und deren Transformation eingegangen. Es wird auch auf Probleme, die bei der Verwendung der Drehwinkelgeschwindigkeiten als Sensordaten auftreten können, Bezug genommen. Danach wird das Jacobi-, beziehungsweise das inverse Jacobi-Modell angesprochen. Als letzter Teil werden die nötigen Grundlagen im Bereich der Bildverarbeitung erläutert, die bei der Evaluierung des Systems verwendet werden.

Ein weiteres Kapitel beschäftigt sich mit den Grundlagen der verwendeten Hardware. Hier werden zunächst die verwendeten Sensortypen beschrieben, wie deren Daten verarbeitet werden, und es wird das Problem des Sensordrifts bei internen Sensoren erläutert. Später werden verschiedene Systeme der verwendeten Mikrocontroller, wie Interrupts, Timer, Analog-Digital-Wandler (A/D-Wandler) und die verwendeten seriellen Schnittstellen zum Host-System und den Aktuatoren dargestellt.

Im Anschluss daran folgt ein Kapitel, das die einzelnen Konzepte der vorliegenden Arbeit beschreibt. Zum einen wird hier genau die zugrundeliegende Theorie hinter den Stabilisierungsalgorithmen erläutert. Es wird auf die mögliche Realisierung der gewünschten Teilaufgaben eingegangen. Diese Aufgaben sind die Adaptierung auf andere kinematische Modelle, die Modularität des Programms im Hinblick auf die verwendeten Hardwarekomponenten und die Evaluierung des Systems nach der Entwicklung.

Das folgende Kapitel 6 befasst sich mit der Realisierung eines Systems, mit dem die gestellten Aufgaben gelöst wurden. Es wird genauer auf den Aufbau des Stabilisierungs- und Testsystems selbst eingegangen. Dies beinhaltet zum einen den Programmablauf, aber auch die verwendeten Hardwarekomponenten. Es werden auch aufgetretene Probleme und deren Lösungen beschrieben. Danach werden die genauen Softwareschnittstellen der einzelnen Softwaremodule beschrieben, die jeweils eine Hardwarekomponente repräsentieren. Der letzte Teil dieses Kapitels beschreibt ein Konfigurationswerkzeug, mit dem es möglich ist, die Stabilisierungssoftware auf verschiedene kinematische Modelle anzupassen.

Der experimentellen Erprobung des Stabilisierungssystems ist im Folgenden ein separates Kapitel gewidmet. Am Anfang werden Experimente mit den verwendeten Hardwarekomponenten und im speziellen mit den Gyroskopen dargestellt. Danach werden die Experimente beschrieben, die mit der Evaluierung des Systems zusammenhängen. Bei diesen Experimenten wird die Rückstellgenauigkeit und die Geschwindigkeit des Stabilisierungssystems überprüft. Nach der Darstellung der Experimente werden die Messergebnisse präsentiert und ausgewertet.

Abschließend wird die Arbeit in Kapitel 8 zusammengefasst und ein Ausblick auf mögliche Verbesserungen gegeben. Zum einen wird hier auf die Verwendung von externen Sensoren zur Langzeitstabilisierung eingegangen und zum anderen wird eine mögliche Erweiterung des Stabilisierungssystems erläutert. Hierbei würde dann nicht nur die Orientierung des Endeffektors stabilisiert werden, sondern zusätzlich die Position. Die für die Arbeit durchgeführte Evaluierung wurde teilweise manuell durchgeführt. Ein Abschnitt befasst sich damit, wie auch dieser Teil rechnergestützt ausgewertet werden könnte.

Im Anhang der Arbeit wird die Installation beziehungsweise Einrichtung von Javaanwendungen auf Windowsrechnern erklärt, da ein entwickeltes Konfigurationswerkzeug unter Java entwickelt wurde. Danach folgen mehrere Unterkapitel, in denen die bei der experimentellen Erprobung aufgenommenen Messwerte aufgeführt sind. Im Anschluss werden die Softwareschnittstellen beschrieben, falls für das Stabilisierungssystem eine andere Hardware verwendet werden soll und somit auch die Software angepasst werden muss. Danach werden in einem Kapitel zwei Fotoreihen einer stabilisierten beziehungsweise unstabilisierten Kamera gezeigt. Als letztes Kapitel wurden im Anhang einige Skizzen und Fotos des Stabilisierungssystems angefügt.

2 Stand der Forschung

2.1 Forschung am Fachbereich

Am Fachgebiet SIM der TU Darmstadt werden gerade im Bereich des Roboterfußballs einige Roboter eingesetzt, die Gyroskope integriert haben. Diese Gyroskope werden zum aktuellen Zeitpunkt dazu benutzt, um eine Lagestabilisierung zu erreichen und somit ein Kippen der Roboter zu verhindern. Sobald der Roboter droht, in eine Richtung zu kippen, kommt es zu einer Gewichtsverlagerung durch eine Armbewegung. Dadurch kann dem Kippen des Roboters entgegengewirkt werden. Ein weiteres Projekt beschäftigte sich mit Inertialnavigation. Hierbei wurde ein Pioneer II-Roboter mit Gyroskopen ausgestattet, um autonom navigieren zu können.

Momentan sind am Fachbereich SIM und dem Graduiertenkolleg Cooperative , Adaptive and Responsive Monitoring in Mixed Mode Environments (GKmM) mehrere autonome Fahrzeuge in Entwicklung. Zum einen ist hier die Diplomarbeit von Christian Ecke zu nennen, die die Entwicklung eines autonomen Schiffes zum Ziel hat. Hier wird unter anderem ein Gyroskop zur Lagebestimmung verwendet. Desweiteren war am Institut für Flugsysteme und Regelungstechnik des Fachbereichs Machinenbau ein Quadrocopter in Planung, dessen Lageregelung auch mit Hilfe von drei orthogonal angeordneten Gyroskopen durchgeführt werden soll. Ein weiteres Projekt im Rahmen des GKmM wird von Anguelina Vatcheva ein radgetriebenes, autonomes Modellfahrzeug entwickelt. Auch hier sollen Gyroskope für die Lagebestimmung und Navigation zum Einsatz kommen. In allen Fahrzeugen, in denen Inertialsensoren verwendet werden, könnten diese gleichzeitig genutzt werden, um externe Sensoren zu stabilisieren.

Diese Projekte beschäftigen sich zwar jeweils mit der Verwendung von Gyroskopen in autonomen Systemen, jedoch werden hier die Sensoren für die Lagestabilisierung und -regelung des Fahrzeugs selbst verwendet. In der vorliegenden Arbeit wird gezielt mit Hilfe von Gyroskopen und Aktoren ein Sensor, beziehungsweise der Endeffektor einer kinematischen Kette, in seiner Lage stabilisiert. Hierbei ist unter Umständen die Lage der einzelnen Gyroskope und Aktoren schon vom vorliegenden Roboter- und Fahrzeugmodell vorgegeben. Die Software auf dem eingesetzten Mikrocontroller wird

vorher mittels des Konfigurationswerkzeuges sowohl an das gegebene kinematische Modell als auch an das Sensormodell angepasst.

Eine weitere Arbeit ist die Diplomarbeit von Gerhard Rohe. In dieser Arbeit wird ebenfalls eine Lagebestimmung durchgeführt. Hierbei kommen jedoch keine Inerialsensoren zum Einsatz sondern eine Kamera. Aufgrund einer Featureerkennung in verschiedenen aufgenommenen Bildern kann eine Orientierungsänderung und eine Translationsrichtung bestimmt werden. Die Lagebestimmung benötigt jedoch auf einem herkömmlichen PC einige Sekunden Rechenzeit für die Featureerkennung und Zuordnung von markanten Punkten auf beiden Bildern, so dass dieses Sysem momentan nicht in Echtzeit eingesetzt werden kann. Es wäre denkbar, diese Arbeit mit der vorliegenden Arbeit zu kombinieren, um neben der sehr schnellen Inertialstabilisierung auf eine zusätzliche Stabilisierung zurückgreifen zu können.

2.2 Stabilisierte Sensoren

An der TU München wurde ein System [3] für eine stabilisierte Kamera entwickelt, die in Fahrzeugen zum Einsatz kommen soll. Bei dem System handelt es sich um eine Kamera, die von zwei Aktuatoren um zwei Achsen gedreht werden kann. Als Sensoren kommt hierbei eine inertiale Messeinheit (IMU) in Form von drei rechtwinklig zueinander angebrachten Gyroskopen zum Einsatz.

Es wurde ein in sich geschlossenes und abgeschirmtes System geschaffen, das alle nötigen Teile wie Kamera, IMU und Aktuatoren beinhaltet. In diesem System ist die IMU fest an der Kamera angebracht und die Aktuatoren inklusive Harmonic Drive Getrieben so installiert, dass sich das Sytem um seinen Masseschwerpunkt dreht. Die komplette Einheit wurde am Dach eines Fahrzeugs fest installiert.

In diesem Projekt wurde eine Digitalkamera mit Hilfe von Inertialsensoren stabilisiert. Allerdings handelt es sich hierbei um ein in sich geschlossenes System, in dem hochwertige Einzelkomponenten zum Einsatz kommen. Ein Ziel der vorliegenden Arbeit war es hingegen, ein System zu entwickeln, das im Hinblick auf die verwendete Hardware und das Modell der kinematischen Kette leicht portierbar sein sollte.

2.3 Animate Vision

Mit seiner ersten Veröffentlichung im Bereich Animate Vision [1] hat Dana H. Ballard den Bereich des Maschinensehens um einen Schritt erweitert. Bei der Animate Vision geht es nicht mehr nur darum, einzelne Bilder und die darauf gezeigten Merkmale auszuwerten, sondern die Kameras in mehreren

Freiheitsgraden zu bewegen, um den vestibulookulären Reflex (siehe Kapitel 2.5) des natürlichen Sehens zu imitieren.

Ballard hat in seiner Arbeit gezeigt, dass es für einen Computer in der Rechenzeit günstiger ist, wenn die Kameras separat über eine gezielte Blickrichtungsregelung verfügen. Aufgrund der Arbeiten von Ballard gab es einige weitere Arbeiten, die sich mit dem Thema Blickrichtungsregelung und Animate Vision beschäftigten. Da die Auswahl der Veröffentlichungen mit dem Thema Animate Vision sehr umfangreich ist, werden hier nur einige Beispiele genannt. Es gibt zum Beispiel Arbeiten, die sich mit der Objektverfolgung [2], verschiedenen Steuerungslayern und deren parallelisierter Verarbeitung [4], rekonfigurierbaren Bildverarbeitungsarchitekturen [5] oder der Verwendung von Animate Vision-Systemen im sozialen Umfeld [6] befassen.

Für die Lageerkennung wurden in der Arbeit von Ballard und in den Nachfolgearbeiten keine Inertialsensoren verwendet, sondern vielmehr die Kamerabilder genutzt, um die Kamera selbst zu stabilisieren. Als Algorithmen wurde auf neuronale Netze zurückgegriffen, um die Stabilisierung zu gewährleisten.

Für eine Implementierung auf einem Mikrocontroller, wie sie momentan in den meisten kleineren Robotern zum Einsatz kommen, ist die Rechenleistung nicht ausreichend. Eine Möglichkeit wäre, um diese Arbeiten dennoch auf Robotern mit beschränkter Rechenkapazität nutzen zu können, die Daten auf einem externen Hostrechner zu verarbeiten. Dazu wäre ein schnelles drahtloses Netzwerk (WLAN) nötig. Es sind jedoch einige Anwendungsszenarien denkbar, in denen kein Hostrechner oder WLAN zur Verfügung steht.

Ein weiteres Problem bei der Blickrichtungsregelung mittels Kameras ist die Tatsache, dass stets eine Kamera als Sensor zur Verfügung stehen muss. Wenn zum Beispiel als externer Sensor ein Laser- oder Ultraschallsensor stabilisiert werden soll, so müsste zusätzlich eine Kamera verfügbar sein, um die Lagebestimmung durchzuführen.

2.4 Inertialsensoren als Gravitationsreferenz

Eine weitere Arbeit im Bereich des Computersehens ist eine Veröffentlichung von Jorge Lobo und Jorge Dias [7]. Hierbei werden Beschleunigungssensoren verwendet, um während der Bildaufnahme eine Referenz bezüglich der Erdanziehungskraft zu besitzen. Wenn ein System sich in Ruhe befindet oder sich mit gleichbleibender Geschwindigkeit bewegt, so wirkt auf der Erde trotzdem die Erdschwerebeschleunigung von ca. $g = 9,81 m/s^2$. Diese Tatsache kann ausgenutzt werden, um mit den drei Beschleunigungssensoren einen senkrechten Vektor zu bestimmen, falls die Tatsache gegeben ist und das System still steht oder sich mit konstanter Geschwindigkeit bewegt. Sobald dieser Vektor bestimmt ist, kann auch der Horizont bestimmt werden. Diese Information wird dann in diesem Paper [7] genutzt, um weitere 3D-Informationen aus Kamerabildern zu gewinnen.

2.5 Vestibulookulärer Reflex

Ein Stabilisierungssystem, wie es in dieser Diplomarbeit vorgestellt wird, existiert bereits in der Natur. Hierbei handelt es sich um den sogenannten vestibulookulären Reflex (VOR). Dieser Reflex gleicht bei Menschen und Tieren die Bewegungen des Kopfes mit Hilfe des okulomotorischen System aus. Die Bewegungen werden vom Gleichgewichtsorgan im Innenohr aufgenommen und direkt an die Augenmuskeln weitergeleitet.

Der vestibulookuläre Reflex gleicht beim Menschen ständig selbst die kleinsten Kopfbewegungen aus. Ausserdem hilft er dem Menschen, selbst bei schnellen Bewegungen des Kopfes, wie sie zum Beispiel beim Rennen auftreten, das Sichtfeld relativ stabil zu halten.

In der Biologie wurde der vestibulookuläre Reflex bereits einschlägig untersucht und wurde zum Beispiel in Büchern der Neurobiologie [8] beschrieben. Auf Grundlage der biologischen Untersuchungen des verstibulookulären Reflexes wurden einige biologisch inspirierte Arbeiten veröffentlicht. Diese Arbeiten beziehen sich teilweise auf den vestibulookulären Reflex selbst, wie er mit Hilfe neuronaler Netze [9] oder spezieller Regelschleifen [10] modelliert werden kann.

Es wurden jedoch auch biologisch inspirierte Roboterköpfe [11] entwickelt, die mit Inertialsensoren (Gyroskopen und Beschleunigungssensoren) ausgestattet sind und über mehrere Aktuatoren den Kopf bewegen können. Darüber hinaus gibt es sogar Arbeiten, die sich schon mit der Interaktion eines Roboters mit Menschen auseinandersetzen. Hierbei ist als Beispiel die Arbeit von Jiang Peng, Alan Peters, Xinyu Ao und Arthit Srikaew [12] zu nennen, bei der ein Roboter über einen beweglichen Roboterkopf ein winkendes Objekt erkennt und danach greift.

Diese Arbeiten kommen dem gewünschten Stabilisierungssystem teilweise schon sehr nahe. Es handelt sich jedoch stets um in sich geschlossene Systeme, bei denen das kinematische Modell fest vorgegeben ist. Die Systeme sind so nicht ohne weiteres auf andere kinematische Anordnungen übertragbar.

2.6 Stand der Technik

Im Folgenden sind einige Beispiele genannt, bei denen die Sensorstabilisierung und die Nutzung von Inertialsensoren bereits Einzug in die Technik gefunden haben. Einen Überblick, hauptsächlich über die Stabilisierungssysteme, die von der Firma Canon angeboten werden, bietet der Artikel “Image stabilization shows diversity of engineering approaches” von Bill Schweber [13].

2.6.1 Bildstabilisierung

In der Technik kommen Stabilisierungen für Sensoren hauptsächlich bei verschiedenen Foto- und Videokameras zum Einsatz. Das Einsatzgebiet ist hier jedoch nicht die komplette Stabilisierung der Kameraorientierung im Raum, sondern vielmehr das Ausgleichen von Vibrationen und Wackeln der Kamera, um ein möglichst ruhiges Bild aufzunehmen. Für diesen Zweck haben verschiedene Hersteller recht unterschiedliche Verfahren entwickelt, die im folgenden kurz vorgestellt werden. Bei diesen Systemen wird, im Gegensatz zur vorliegenden Arbeit, nur eine relativ kleine Orientierungsänderung stabilisiert. In den folgenden drei Unterkapiteln werden verschiedene Methoden der Bildstabilisierung vorgestellt, die in der Technik Anwendung finden.

Optische Stabilisierung

Die optische Stabilisierung stellt sicher, dass der einfallende Lichtstrahl durch die Linse auch trotz Verwacklung des Kamerakörpers vibrationsfrei auf den Aufnahmesensor fällt. Hierbei können zwei verschiedene Methoden zum Einsatz kommen.

Durch bewegliche Linsen und Prismen im Objektiv der Kamera kann der einfallende Lichtstrahl gezielt so verändert werden, dass er trotz Verwackelung der Kamera an die gleiche Stelle des Aufnahmesensors fällt. Im Objektiv der Kamera sind sowohl Inertialsensoren als auch ein Mikrocontroller eingebaut, mit denen die Optik gesteuert wird. Verwendung findet diese Art der Stabilisierung zum Beispiel in den Technologien Canon - Image Stabilizer [14], Nikon - Vibration Correction [15], Sigma - Optical Stabilizer [16], Tamron - Vibration Compensation [17] und Panasonic - Optical Image Stabilizer [18]. Die optische Stabilisierung mittels Objektiv kann sowohl bei digitalen als auch bei analogen Kamerassystemen zum Einsatz kommen.

Eine weitere Möglichkeit der optischen Stabilisierung besteht darin, den Bildaufnahmesensor innerhalb der Kamera selbst beweglich zu integrieren. Anstatt des Lichtstrahles wird nun der Sensor so bewegt, dass äußere Vibrationen ausgeglichen werden. Die komplette Technik für diese Methode muss in der Kamera selbst untergebracht sein. Einen beweglichen Bildsensor besitzen zum Beispiel Kameras mit den Techniken Sony - Super Steady Shot [19], Pentax Shake Reduction System [20], Ricoh - Vibration Correction und Olympus Image Stabilization [21]. Diese Art der optischen Stabilisierung kann ausschließlich bei digitalen Kamerasystemen verwendet werden. Da bei der Vibrationskompensation die Aufnahmeeinheit sehr schnell bewegt werden muss, kann dies nur mit dem relativ kleinen Charge-Coupled-Device-Sensor (CCD-Sensor)einer Digitalkamera realisiert werden und nicht mit dem Film und der dazugehörigen Transporteinheit einer Analogkamera.

Eine Veröffentlichung, die sich mit optischer Stabilisierung befasst, ist zum Beispiel ein Patent von Juan J. de la Cierva [22]. Hierin wird die Stabilisierung des Kamerabildes mit Hilfe von beweglichen

Spiegeln durchgeführt. Diese Spiegel werden auf Grund der Messdaten eines Gyroskops gesteuert. Damit ist es möglich, den einfallenden Lichtstrahl so zu verändern, dass beim Kamerabild kleine Vibrationen ausgeglichen werden können.

Mechanische Stabilisierung

Bei der mechanischen Stabilisierung wird der Kamerakörper selbst stabilisiert. Im einfachsten Fall geschieht dies durch ein extern angeschraubtes Stativ. Mit einem solchen Stativ ist die Kamera jedoch nur statisch einsetzbar. Sobald ein mobiles System benötigt wird, kann das Stativ alleine nicht eingesetzt werden.

Von verschiedenen Firmen wie zum Beispiel Ken-Labs, Kenyon oder Glidecam werden aber auch sogenannte Gyro-Stabilizer angeboten. Hierbei handelt es sich um mechanische Gyroskope, die fest an der Kamera angebracht werden. Die Gyroskope selbst werden aber hier nicht als Sensoren eingesetzt, sondern stabilisieren mit ihrer Schwungmasse die Kamera selbst um jeweils eine Achse.

Diese Systeme kommen zum Beispiel bei Film und Fernsehen überall dort zum Einsatz, wo die Kamera von Hand bewegt wird. In der Regel werden solche Systeme eingesetzt, wenn aus einem Fahrzeug oder Hubschrauber eine Szene aufgenommen werden soll. Außerdem wird die mechanische Stabilisierung auch von Paparazzi an normalen Fotokameras angebracht, um möglichst schnell verwacklungsfreie Fotos zu schießen.

Elektronische Stabilisierung

Neben der mechanischen Stabilisierung von Kamerabildern gibt es auch die Möglichkeit einer elektronischen Stabilisierung. Hierbei handelt es sich um eine digitale Nachbearbeitung, die schon direkt in die Kamera integriert ist. Es können unterschiedliche Verfahren zum Einsatz kommen. Einfaches Beispiel wäre ein Filter zum nachträglichen Schärfen des Bildes oder die Kombination zweier Bilder mit unterschiedlichen Belichtungszeiten. Bilder mit kurzen Belichtungszeiten besitzen in der Regel eine höhere Schärfe, aber die Farbinformationen können so nicht richtig wiedergegeben werden. Systeme mit elektronischer Stabilisierung kommen in Videosystemen mit der Technik Nikon - Electronic Vibration Reduction, Casio - Anti-Shake-Digital Signal Processor (DSP) oder Samsung - Anti-Shake-Reduction zum Einsatz. Diese Art der Bildstabilisierung muss direkt in der Kamera integriert sein und kann nicht nachträglich durch ein Objektiv aufgerüstet werden. Von den Herstellern gibt es keine genaueren Informationen über diese Art der Bildstabilisierung. Wissenschaftliche Arbeiten, die sich mit digitaler Stabilisierung von Bildern beschäftigen, sind unter anderem die Arbeit von Carlos Morimoto und Rama Chellappa [23], die einen Algorithmus für die Bildstabilisierung vorstellen oder eine Arbeit von Alan C. Brooks [23], in der verschiedene Stabilisierungalgorithmen verglichen werden.

2.6.2 Inertialsensoren

Inertialsensoren werden heutzutage hauptsächlich im Bereich der Navigation von Fahrzeugen eingesetzt. Zusätzlich kommen sie für Steuerungs- und Stabilisierungsfunktionen bei unbemannten Fahrzeugen und mobilen Robotern zum Einsatz. Die Stabilisierung wird hier jedoch in erster Linie für das Fahrzeug selbst durchgeführt und nicht für ein spezielles Sensorsystem.

Teilweise kommen Inertialsensoren auch in fest installierten Navigationsgeräten in Fahrzeugen zum Einsatz. Hierbei unterstützen sie das Global Positioning System (GPS), falls kein oder ein nicht ausreichendes Signal zur Verfügung steht. Bei der Navigation von Großraumflugzeugen mit Hilfe des Autopiloten werden neben dem GPS stets Inertialsensoren verwendet, um eine genaue Positionsbestimmung zu gewährleisten. Dort werden jedoch weitaus größere, schwerere und vor allen Dingen teurere Sensoren verwendet, als sie in dieser Arbeit betrachtet werden. Die Größe von ca. $10 \times 10 \times 10cm$ und ein ungefähres Gewicht von $0,5kg$ wäre in größeren Robotersystemen noch vertretbar, aber ein Preis von über 100.000$ würde wohl bei weitem den finanziellen Rahmen einer Diplomarbeit sprengen.

Ein weiteres, jedoch auch sehr ausgefallenes Einsatzgebiet von Inertialsensoren sind Trackingsysteme. Hierbei wird die Position oder Gelenkstellung eines Lebewesens gemessen. Da die Gelenke von Menschen und Tiere hauptsächlich durch Rotationsgelenke modelliert werden, liegt es nahe, dass deren Rotationsgeschwindigkeiten durch Gyroskope gemessen werden können.

3 Grundlagen (Theorie)

In diesem Kapitel werden alle geometrischen, mechanischen und kinematischen Grundlagen erläutert, auf die in späteren Kapiteln der Arbeit Bezug genommen wird und die für die Entwicklung der Arbeit von Bedeutung sind. Sofern keine anderen Quellen angegeben sind, wurde für diesen Teil der Arbeit das Skriptum zur Vorlesung "Robotik 1" [24] von Professor von Stryk als Quelle verwendet.

3.1 Transformationen

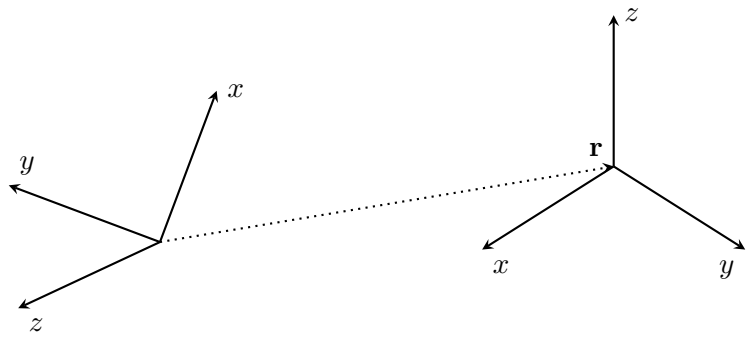

Abb. 3.1: Transformationen zwischen Koordinatensystemen

In der Robotik und anderen Disziplinen, wie zum Beispiel der grafischen Datenverarbeitung, sind für die Beschreibung von Objekten im Raum dreidimensionale Koordinaten nötig. Hierzu gibt es ein festes Weltkoordinatensystem und in der Regel weitere lokale Koordinatensysteme, die sich vom Weltkoordinatensystem in Position und Orientierung unterscheiden können. Ein Koordinatensystem ist jeweils definiert durch einen dreidimensionalen Ortsvektor und drei dreidimensionale Einheitsvektoren. Der Ortsvektor definiert die Position und die Einheitsvektoren $\mathbf{e}_1$, $\mathbf{e}_2$ und $\mathbf{e}_3$ bestimmen die Orientierung des Koordinatensystems bezüglich eines anderen Bezugskoordinatensystems.

Ist nun ein Vektor $\mathbf{v}_0$ in einem Koordinatensystem S_0 bekannt, so kann man ihn auch bezüglich eines anderen Koordinatensystems S_1 transformieren. Um diese Transformation durchzuführen, müssen der

Positionsvektor $\mathbf{r}$ und die drei Einheitsvektoren von S_1 bezüglich S_0 bekannt sein. Die Transformation berechnet sich zu

$$\mathbf{v}_0 = {}^0R_1 \cdot \mathbf{v}_1 + \mathbf{r}, \tag{3.1}$$

wobei

$${}^0R_1 = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{pmatrix} \tag{3.2}$$

die 3×3-Rotationsmatrix ist, die sich durch spaltenweises Aneinanderhängen der Vektoren $\mathbf{e}_1$, $\mathbf{e}_2$ und $\mathbf{e}_3$ ergibt.

Bei den Transformationen handelt es sich also um affin lineare Funktionen im $\mathbb{R}^3$. Wenn viele Vektoren transformiert werden müssen oder mehrere Transformationen nacheinander stattfinden, so ist es einfacher, homogene Koordinaten [25, 26] für die Rechnungen zu verwenden. Hierbei handelt es sich um Koordinaten im $\mathbb{R}^4$. Die Rechnungen benötigen in oben genannten Fällen jedoch trotzdem weniger Rechenzeit, da es sich stets um lineare Funktionen handelt.

In dieser Arbeit wird nicht weiter auf homogene Koordinaten eingegangen. Für die Blickrichtungssteuerung sind nur die Orientierungen zwischen den Koordinatensystemen relevant. Rotationen sind bereits im dreidimensionalen lineare Abbildungen und benötigen daher keine homogenen Koordinaten für eine Linearisierung der Berechnungen.

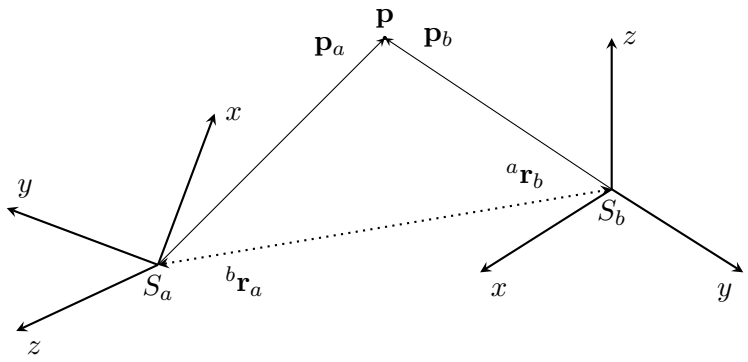

Abb. 3.2: Schaubild zur Veranschaulichung der Notation

Notation 1 *In den folgenden Unterkapiteln werden häufig Rotationsmatrizen und Vektoren verwendet, die bezüglich unterschiedlicher Koordinatensysteme vorliegen. Um Verwirrungen zu vermeiden, bekommen die Matrizen und Vektoren jeweils Indizes, anhand derer sofort festgestellt werden kann, in welches Koordinatensystem transformiert wird oder bezüglich welches Koordinatensystems ein Vektor vorliegt.*
Gegeben seien die beiden Koordinatensysteme S_a und S_b und ein Punkt $\mathbf{p}$ im Raum. Die Orientierung von Vektoren der Koordinatensysteme zueinander kann durch Rotationsmatrizen ausgedrückt werden. Hierbei ist im Folgenden ${}^aR_b$ die Darstellung der Orientierung eines Vektors im Koordinatensystem S_b ins System S_a. In der Rückrichtung ist ${}^bR_a$ entsprechend die Orientierungsdarstellung eines Vektors von System S_a nach S_b. Wenn ein Vektor in ein anderes Bezugssystem und dann wieder zurück transformiert wird, so muss er wieder dem ursprünglichen Vektor entsprechen, somit muss gelten

$$E = {}^aR_b \cdot {}^bR_a = {}^bR_a \cdot {}^aR_b, \tag{3.3}$$

wobei E die Einheitsmatrix ist.
Neben der Orientierung der Koordinatensysteme kann auch die Position ineinander überführt werden. Hierzu müssen die Positionsvektoren ${}^a\mathbf{r}_b$ von S_b bezüglich S_a und in der Gegenrichtung ${}^b\mathbf{r}_a$ von S_a bezüglich S_b bekannt sein. Der Zusammenhang zwischen den beiden Positionsvektoren ist

$${}^a\mathbf{r}_b = {}^aR_b \cdot (-{}^b\mathbf{r}_a) \tag{3.4}$$

beziehungsweise

$${}^b\mathbf{r}_a = {}^bR_a \cdot (-{}^a\mathbf{r}_b). \tag{3.5}$$

Der Punkt $\mathbf{p}$ kann bezüglich der beiden Koordinatensysteme dargestellt werden, so dass man in der Regel zwei verschiedene Darstellungen $\mathbf{p}_a$ und $\mathbf{p}_b$ für die Bezugssysteme S_a und S_b erhält. Die beiden Vektoren können durch

$$\mathbf{p}_a = {}^aR_b \cdot \mathbf{p}_b + {}^a\mathbf{r}_b \tag{3.6}$$

und

$$\mathbf{p}_b = {}^bR_a \cdot \mathbf{p}_a + {}^b\mathbf{r}_a \tag{3.7}$$

ineinander überführt werden.

Anmerkung 1 *Die Position der Koordinatensysteme wird in der vorliegenden Arbeit nicht berücksichtigt. Die Transformation der Koordinatensysteme hinsichtlich ihrer Position ist hier nur der Vollständigkeit halber aufgeführt. Sie käme erst bei einer kompletten Stabilisierung des Endeffektors einer kinematischen Kette zum Tragen, wie sie im Ausblick erwähnt ist. Hierbei sollten dann auch eher homogene Koordinaten (4×4-Matrizen und Vektoren im $\mathbb{R}^4$) verwendet werden.*

3.1.1 Rotationen

Die Rotation eines Vektors $\mathbf{p}$ um eine Drehachse $\mathbf{a}$ und einen Drehwinkel θ können durch die Multiplikation einer Rotationsmatrix R mit dem Vektor $\mathbf{p}$ erreicht werden. Das Ergebnis $\mathbf{p}_{rot} = R \cdot \mathbf{p}$ ist der rotierte Vektor.

3.1.2 Elementare Rotationen

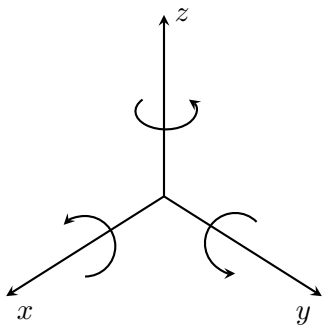

Abb. 3.3: Elementare Rotationen

Bei diesen Transformationen können elementare Rotationsmatrizen verwendet werden, bei denen die Drehachse mit einer der Achsen des Koordinatensystems zusammenfällt.

Im $\mathbb{R}^3$ haben die elementaren Rotationsmatrizen um die $x-$, $y-$ und $z-$Achse die Gestalt:

$$R(x;\theta_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} \tag{3.8}$$

$$R(y;\theta_y) = \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix} \tag{3.9}$$

$$R(z;\theta_z) = \begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

3.1.3 Eigenschaften von Rotationsmatrizen

Folgende Eigenschaften gelten für Rotationsmatrizen:

- Die Rotationsmatrix für eine Rotation um eine beliebige Achse und den Winkel 0° ist die Einheitsmatrix.
- Rotationsmatrizen sind nur dann kommutativ, wenn die Rotation beider Matrizen um die gleiche Achse geschieht oder mindestens eine der beiden Matrizen die Einheitsmatrix ist.
- Die Spaltenvektoren einer Rotationsmatrix besitzen die euklidische Länge 1 und stehen jeweils paarweise orthogonal aufeinander. Somit handelt es sich bei allen Rotationsmatrizen um orthonormale Matrizen.
- Durch die Orthonormalität der Rotationsmatrizen erhält man die Inverse einer solchen Matrix einfach, indem man die Matrix transponiert ($R^{-1} = R^T$).

3.1.4 Verkettungen von Rotationen

Transformationsmatrizen für verkettete Rotationen können aufgestellt werden, indem man die Rotationsmatrizen der Einzelrotationen miteinander multipliziert. Da die Multiplikation von Rotationsmatrizen im allgemeinen nicht kommutativ ist, muss auf die Reihenfolge bei der Multiplikation geachtet werden.

Ausgehend vom gewünschten Bezugskoordinatensystem steht die erste Rotationsmatrix ganz rechts, und alle nachfolgenden Rotationsmatrizen werden jeweils nacheinander von links multipliziert. Hierbei kann nachfolgend unter anderem zeitlich aufeinander folgend bedeuten. Dann würde die zeitlich älteste Rotationsmatrix ganz rechts und die zeitlich jüngste Rotation ganz links stehen.

Im Falle von kinematischen Ketten hat man verschiedene Bezugskoordinatensysteme $(S_0, S_1, ..., S_n)$, die hinsichtlich ihrer Orientierung mit Hilfe von Rotationsmatrizen $({}^0R_1, {}^1R_2, ..., {}^{n-1}R_n)$ jeweils ineinander überführt werden. Wenn die Orientierung des Koordinatensystems S_n bezüglich des Basiskoordinatensystems S_0 beschrieben werden soll, so geschieht dies durch die Rotationsmatrix

$$ {}^0R_n = {}^0R_1 \cdot {}^1R_2 \cdot ... \cdot {}^{n-1}R_n. \tag{3.11} $$

Anmerkung 2 *Da die Inverse einer Rotationsmatrix gleich der Transponierten ist, kann man auch die Gesamtrotation in die andere Richtung aufstellen. Das heisst, dass man die Orientierung des Basiskoordinatensystems S_0 bezüglich des Koordinatensystems S_n kennt. Die Reihenfolge der Rotationsmatrizen ist hierbei umgekehrt und für jede Rotationsmatrix muss die Inverse genommen werden, also die Transformation genau in die andere Richtung. Für die Transformation in die entgegengesetzte Richtung ergibt sich die Rotationmatrix*

$$ \begin{aligned} {}^nR_0 &= {}^nR_{n-1} \cdot {}^{n-1}R_{n-2} \cdot ... \cdot {}^1R_0 \\ &= {}^{n-1}R_n^{-1} \cdot {}^{n-2}R_{n-1}^{-1} \cdot ... \cdot {}^0R_1^{-1} \\ &= {}^{n-1}R_n^T \cdot {}^{n-2}R_{n-1}^T \cdot ... \cdot {}^0R_1^T. \end{aligned} \tag{3.12} $$

3.1.5 Allgemeine Rotationsmatrizen

Neben den elementaren Rotationen lässt sich auch eine allgemeine Rotationsmatrix aufstellen, bei der um eine beliebige Ache $\mathbf{v} = (v_1, v_2, v_3)^T$ der Länge 1 ($|\mathbf{v}| = 1$) und einen Drehwinkel α rotiert wird.

Diese Matrix besitzt folgende Gestalt:

$$\begin{pmatrix} \cos\alpha + v_1^2(1-\cos\alpha) & v_1v_2(1-\cos\alpha) - v_3\sin\alpha & v_1v_3(1-\cos\alpha) + v_2\sin\alpha \\ v_2v_1(1-\cos\alpha) + v_3\sin\alpha & \cos\alpha + v_2^2(1-\cos\alpha) & v_2v_3(1-\cos\alpha) - v_1\sin\alpha \\ v_3v_1(1-\cos\alpha) - v_2\sin\alpha & v_3v_2(1-\cos\alpha) + v_1\sin\alpha & \cos\alpha + v_3^2(1-\cos\alpha) \end{pmatrix} \tag{3.13}$$

3.1.6 Drehachse und Drehwinkel bestimmen

Aus einer beliebigen Rotationsmatrix lässt sich auch die Drehachse und der Drehwinkel bestimmen. Gegeben sei die Drehmatrix

$$D = \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix}, \tag{3.14}$$

so ergibt sich der Drehwinkel zu

$$\phi = \arccos\left(\frac{1}{2}(Spur(D) - 1)\right). \tag{3.15}$$

Die Drehachse wird berechnet zu

$$\mathbf{a} = \frac{1}{2\sin\phi}\begin{pmatrix} d_{32} - d_{23} \\ d_{13} - d_{31} \\ d_{21} - d_{12} \end{pmatrix} \text{ für } \phi \neq \pi. \tag{3.16}$$

3.2 Winkelgeschwindigkeiten

Genau wie es lineare Geschwindigkeiten gibt, so gibt es auch Drehwinkelgeschwindigkeiten bei rotierenden Systemen. Die Winkelgeschwindigkeit ist über einen Vektor einer bestimmten Länge definiert. Die Orientierung des Winkelgeschwindigkeitsvektors $\boldsymbol{\omega}$ ist parallel zur Drehachse und die Länge des Vektors ist proportional zur Winkelgeschwindigkeit.

Die Winkelgeschwindigkeit eines rotierenden Körpers ist an jedem Punkt des Körpers gleich. Die Bahngeschwindigkeit zweier Punkte dieses Körpers ist jedoch nicht unbedingt gleich. Die Bahngeschwin-

digkeit ist zusätzlich zur Winkelgeschwindigkeit $\boldsymbol{\omega}$ auch vom Radius $\mathbf{r}$ des Punktes zur Drehachse abhängig.

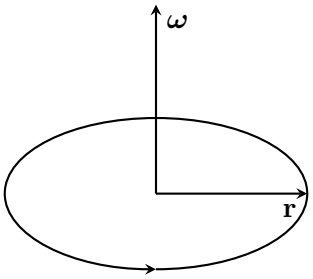

Abb. 3.4: Radius und Winkelgeschwindigkeit

Man kann die aktuelle Bahngeschwindigkeit $\mathbf{v}$ eines Punktes berechnen, der sich mit dem Radius $\mathbf{r}$ um eine Drehachse und der Winkelgeschwindigkeit $\boldsymbol{\omega}$ dreht. Dies geschieht über die Berechnung des Kreuzprodukts zwischen der Winkelgeschwindigkeit und dem Radiusvektor. Der Radiusvektor $\mathbf{r}$ ist hierbei der Vektor, der rechtwinklig auf der Drehachse beginnt und zu dem Punkt geht, an dem die Bahngeschwindigkeit berechnet werden soll.

$$\mathbf{v} = \boldsymbol{\omega} \times \mathbf{r} \tag{3.17}$$

3.2.1 Transformation in andere Koordinatensysteme

Wenn eine Winkelgeschwindigkeit $\boldsymbol{\omega}_b$ bezüglich eines Koordinatensystems S_b bekannt ist, so kann sie auch bezüglich eines anderen Koordinatensystems S_a dargestellt werden, sofern keine weiteren Drehwinkelgeschwindigkeiten wirken! Hierzu muss die Orientierung von S_b bezüglich S_a gegeben durch eine Rotationsmatrix ${}^aR_b$ bekannt sein. Die Winkelgeschwindigkeit bezüglich S_a ergibt sich dann zu:

$${}^a\boldsymbol{\omega}_b = {}^aR_b \cdot \boldsymbol{\omega}_b \tag{3.18}$$

Mit Hilfe der Rotationsmatrix wird der Winkelgeschwindigkeitsvektor $\boldsymbol{\omega}_b$ bezüglich des Koordinatensystems S_b in einen Winkelgeschwindigkeitsvektor ${}^a\boldsymbol{\omega}_b$ rotiert, so dass er in die gleiche Richtung, bezüglich des neuen Koordinatensystems S_a zeigt. Gleichzeitig bleibt aber auch die Länge von $\boldsymbol{\omega}_b$ durch die Orthonormalität der Rotationsmatrix ${}^aR_b$ erhalten. ${}^a\boldsymbol{\omega}_b$ ist also der gleiche Winkelgeschwindigkeitsvektor wie $\boldsymbol{\omega}_b$ nur bezüglich eines anderen Koordinatensystems.

3.2.2 Addition von Winkelgeschwindigkeiten

Wenn mehrere Winkelgeschwindigkeiten $\boldsymbol{\omega}_i$ gleichzeitig wirken, so können diese addiert werden. Hierbei ist jedoch darauf zu achten, dass alle Winkelgeschwindigkeiten bezüglich des gleichen Koordinatensystems S_0 gegeben sein müssen.

Seien die relativen Winkelgeschwindigkeiten ${}^{i-1}\boldsymbol{\omega}_i$ jeweils bezüglich ihres lokalen Koordinatensystems S_{i-1} und die Rotationsmatrizen ${}^{0}R_{i-1}$ der Orientierungen aller Koordinatensysteme bezüglich des Basiskoordinatensystems S_0 bekannt, so kann der gesamte Winkelgeschwindigkeitsvektor

$$\boldsymbol{\omega}_{ges} = {}^{0}\boldsymbol{\omega}_1 + {}^{0}R_1 \cdot {}^{1}\boldsymbol{\omega}_2 + ... + {}^{0}R_{n-1} \cdot {}^{n-1}\boldsymbol{\omega}_n \tag{3.19}$$

berechnet werden.

3.2.3 Abhängigkeit von Winkelgeschwindigkeiten

Bei mehreren gemessenen Winkelgeschwindigkeiten ist darauf zu achten, dass die Geschwindigkeiten unter bestimmten Voraussetzungen voneinander abhängig sein können. Winkelgeschwindigkeiten sind nur dann ohne weiteres voneinander unabhängig, wenn die Drehachsen zu einander orthogonal stehen.

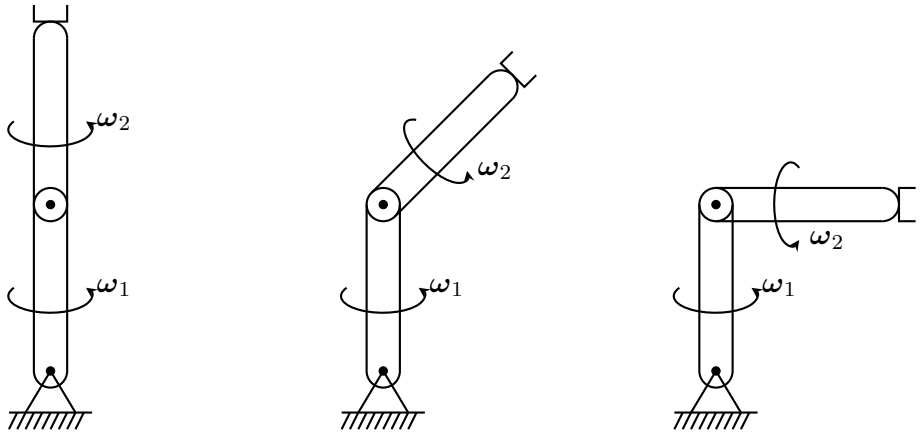

Abb. 3.5: Gelenkstellungsbedingte Abhängigkeit von Sensoren

Sollten die beiden Achsen, um die jeweils eine Drehwinkelgeschwindigkeit gemessen wird, nicht orthogonal zueinander stehen und beide Gyroskope in der gleichen Manipulatorkette sitzen, so ist ein Teil der gemessenen Geschwidigkeiten gleich. Im schlimmsten Fall sind beide gemessenen Drehgeschwindigkeiten sogar komplett gleich, was eine Reduktion um einen Sensor bedeuten würde.

Bild 3.5 zeigt einen Manipulatorarm mit zwei Gliedern. An jedem Manipulatorglied wird jeweils eine Drehwinkelgeschwindigkeit gemessen. Abhängig von der Gelenkstellung zwischen den beiden Gliedern sind die gemessenen Drehwinkelgeschwindigkeiten entweder gleich (links), voneinander abhängig (mitte) oder voneinander unabhängig (rechts).

Anmerkung: Hier sollte nur generell das Problem der Abhängigkeit von verschiedenen, gemessenen Drehwinkelgeschwindigkeiten angesprochen werden. Die Lösung des Problems wird genauer in Kapitel 5.2.4 erläutert.

3.3 Jacobi-Matrix

Die Jacobi-Matrix eines Manipulators ${}^0J_n$ (siehe Craig [27]) wird verwendet, um bei bekannten Sollgeschwindigkeiten der einzelnen Gelenke die anliegende Geschwindigkeit des Endeffektors bezüglich des Basiskoordinatensystems zu berechnen.

Bei der Jacobi-Matrix handelt es sich um eine $(6 \times n)$-Matrix, wobei n die Anzahl der Gelenke des Manipulators ist. Die Matrix selbst kann in zwei Teile untergliedert werden. Die oberen drei Zeilen dienen der Berechnung bezüglich der Geschwindigkeit der Position des Endeffektors und die unteren drei Zeilen der Winkelgeschwindigkeit der Orientierung.

Jede i-te Spalte ${}^0J_{n,i}$ der Jacobi-Matrix ist abhängig vom jeweiligen Gelenktyp (Drehgelenk oder Schubgelenk) und den Stellungen der Gelenke, sowie den Transformationen zwischen den Gelenken davor.

- Drehgelenk:

$$\underbrace{{}^0J_{n,i}}_{\in\mathbb{R}^6} = \begin{pmatrix} {}^0\mathbf{e}_{z_{i-1}} \times ({}^0\mathbf{r}_n - {}^0\mathbf{r}_{i-1}) & \} \in \mathbb{R}^3 \\ {}^0\mathbf{e}_{z_{i-1}} & \} \in \mathbb{R}^3 \end{pmatrix} \tag{3.20}$$

- Schubgelenk:

$$\underbrace{{}^0J_{n,i}}_{\in\mathbb{R}^6} = \begin{pmatrix} {}^0\mathbf{e}_{z_{i-1}} & \} \in \mathbb{R}^3 \\ \mathbf{0} & \} \in \mathbb{R}^3 \end{pmatrix} \tag{3.21}$$

Wobei:

- ${}^0\mathbf{e}_{z_i-1}$: Die Orientierung des dritten Einheitsvektors $((0,0,1)^T)$ im i-ten Gelenk bezüglich des Basiskoordinatensystems,
- ${}^0\mathbf{r}_n$: Der Positionsvektor des Endeffektors bezüglich des Basiskoordinatensystems,
- ${}^0\mathbf{r}_{i-1}$: Der Positionsvektor des $i-1$-ten Gelenks bezüglich des Basiskoordinatensystems und

- $\mathbf{0}$: Der Nullvektor $(0,0,0)^T$ ist.

Für die Winkelgeschwindigkeit des Endeffektors ist nur der jeweils untere Teil der Matrix relevant. Da für ein Schubgelenk in den unteren drei Zeilen immer nur der Nullvektor steht, ist die Orientierung ausschließlich von den Drehgelenken und somit von ${}^0\mathbf{e}_{z_i-1}$ abhängig. Die hier beschriebene Jacobi-Matrix basiert auf den Denavit-Hartenberg-Parametern [28], bei denen in den Drehgelenken stets um die z-Achse gedreht wird. Wenn jeweils die Orientierungen der Achsen, um die in allen Aktuatoren gedreht wird, bezüglich des Basiskoordinatensystems bekannt ist, kann der untere Teil der Jacobi-Matrix aufgestellt werden.

Der untere Teil der Jacobi-Matrix wird aufgestellt, indem man vorab für jeden Aktuator die Drehachse bezüglich des Basiskoordinatensystems S_0 berechnet. Danach wird, beginnend vom Anfang bis zum Ende der kinematischen Kette, nacheinander für jeden Aktuator der Vektor der Drehachse jeweils rechts an die so entstehende Matrix spaltenweise angehängt.

3.4 Jacobi-Modell

Mit dem direkten Jacobi-Modell bezeichnet man das Berechnen der Geschwindigkeiten des Endeffektors einer kinematischen Kette bei gegenebenen Geschwindigkeiten der Aktuatoren. Hierbei muss die Gleichung

$$\begin{pmatrix} {}^0\mathbf{v}_n \\ {}^0\boldsymbol{\omega}_n \end{pmatrix} = {}^0J_n(\mathbf{q}) \cdot \dot{\mathbf{q}}. \tag{3.22}$$

gelöst werden.

Hierbei ist

- ${}^0\mathbf{v}_n$ der lineare Geschwindigkeitsvektor des Endeffektors bezüglich des Basiskoordinatensystems,
- ${}^0\boldsymbol{\omega}_n$ der Winkelgeschwindigkeitsvektor des Endeffektors bezüglich des Basiskoordinatensystems,
- ${}^0J_n(\mathbf{q})$ die von den Gelenkkonfiguration $\mathbf{q}$ abhängige Jacobi-Matrix und
- $\dot{\mathbf{q}}$ der Vektor mit den Geschwindigkeiten der Gelenke.

3.4.1 Inverses Jacobi-Modell

Möchte man bei gegebener linearer Geschwindigkeit ${}^0\mathbf{v}_n$ und Drehwinkelgeschwindigkeit ${}^0\boldsymbol{\omega}_n$ des Endeffektors bezüglich des Basiskoordinatensystems die nötigen Geschwindigkeiten in den Gelenken

$\dot{q}_1, ..., \dot{q}_n$ berechnen, so kann das Gleichungssystem 3.22 auch nach dem Vektor $\dot{\mathbf{q}}$ gelöst werden. Hierbei spricht man vom inversen Jacobi-Modell.

Beim Lösen des Gleichungssystems ist jedoch darauf zu achten, dass die Lösung nicht eindeutig sein muss. Der Manipulatorarm kann zum Beispiel mehrere redundante Gelenke aufweisen, so dass es hierbei zu mehreren verschiendenen Geschwindigkeitsvektoren der Gelenke kommen kann, obwohl jeweils im Endeffektor die gleiche Geschwindigkeit anliegt. Die Lösung des Systems entspräche dann der Lösung eines unterbestimmten Gleichungssystems und damit unendlich vielen Lösungen. In diesem Fall können Zwangsbedingungen vorgegeben werden. Zum Beispiel können in diesem Fall verschiedene Gelenke festgehalten werden.

Die Lösung des Gleichungssystems kann jedoch auch zu einem überbestimmten Gleichungssystem führen. Dies kann der Fall sein, wenn das kinematische Modell ungünstig gewählt wurde und somit der Arbeitsraum des Roboterarms beschränkt ist. Aber auch bei einer günstigen Wahl des kinematischen Modells kann es zu einem überbestimmten Gleichungssystem kommen, falls die Gelenke ungünstig stehen und es zu einer kinematischen Singularität kommt. Das bedeutet, dass es zum Verlust eines Freiheitsgrades kommt. In diesem Falle kann eventuell eine Näherungslösung mit einem Löser gefunden werden, der auch überbestimmte Gleichungssysteme lösen kann.

Eine Möglichkeit, ein überbestimmtes Gleichungssystem zu lösen, wäre die Methode der kleinsten Quadrate [29] anzuwenden. Hierbei wird eine Lösung des Gleichungssystem approximiert und die Quadrate der Abweichungen minimiert. Das zu lösende Gleichunssystem 3.22 würde umgeformt werden zu:

$$\dot{\mathbf{q}} = \left({}^0J_n(\mathbf{q})^T \cdot {}^0J_n(\mathbf{q})\right)^{-1} \cdot {}^0J_n(\mathbf{q})^T \begin{pmatrix} {}^0\mathbf{v}_n \\ {}^0\boldsymbol{\omega}_n \end{pmatrix}. \tag{3.23}$$

${}^0J_n(\mathbf{q})^T$ ist hier die Transponierte der Jacobi-Matrix.

4 Grundlagen (Hardware- und Softwaretechnik)

Der Kern der vorliegenden Arbeit ist ein Softwareprogramm, mit dessen Hilfe der Endeffektor einer kinematischen Kette aufgrund von gemessenen Drehraten stabilisiert werden kann. Für die Entwicklung des Programms soll jedoch auch ein Hardwaresystem entwickelt werden, um die Algorithmen zu testen und nach der abgeschlossenen Implementierung auch zu evaluieren. Dieses Kapitel erläutert die Hardwarebausteine und die Theorie hinter der Funktionsweise und Verwendung der einzelnen Teilsysteme.

4.1 Sensoren

In dieser Diplomarbeit werden ausschließlich Gyroskope als Sensoren verwendet. Diese Art von Sensoren werden im Folgenden genauer beschrieben. Danach wird kurz in einem Unterkapitel zusätzlich auf Beschleunigungssensoren eingegangen. Zum Einen wird auf eine mögliche Verwendung dieser Sensoren im Ausblick am Ende der vorliegenden Diplomarbeit eingegangen, zum Anderen werden teilweise auch komplette inertiale Messeinheiten angeboten, die sowohl Gyroskope als auch Beschleunigungssensoren in allen drei Raumrichtungen vereinigen.

4.1.1 Gyroskope

Ein Gyroskop ist ein Instrument zum Messen der aktuellen Winkelgeschwindigkeit um eine Achse. In den ersten Gyroskopen befand sich ein schnell rotierender, symmetrischer und beweglich gelagerter Rotationskörper. Die Orientierung des Körpers im Raum bleibt wegen der Masseträgheit und der Drehimpulserhaltung konstant. Wenn nun eine äußere Kraft ($\mathbf{F}$) die Drehachse (y) des Rotationskörpers verändert, so kippt die Drehachse aufgrund der Corioliskraft senkrecht zur angreifenden Kraft, um den Gesamtimpuls zu bewahren. Wenn man diese Kippbewegung misst, so kann man durch einen direkten, linearen Zusammenhang auf die angreifende Kraft und die ursprüngliche Drehbewegung des Gyroskops schließen.

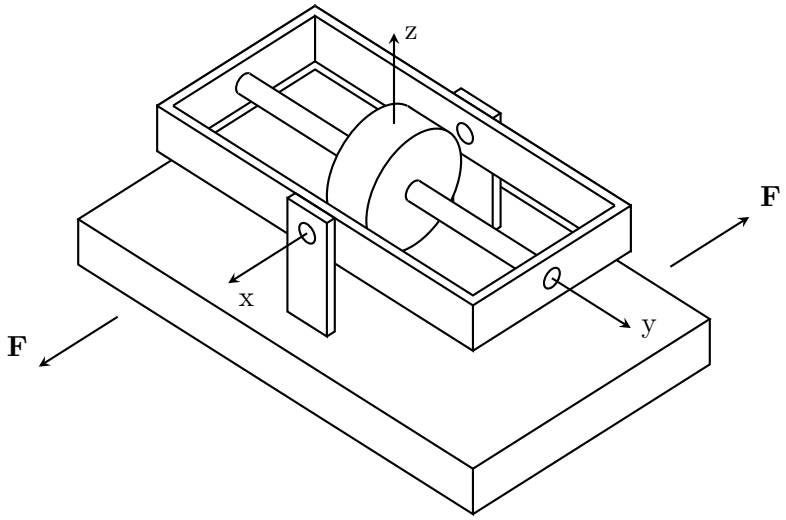

Abb. 4.1: Aufbau eines mechanischen Gyroskops (Quelle: [30])

Heute werden solche mechanischen Gyroskope in der Regel nicht mehr als Sensoren eingesetzt. In der Industrie werden mittlerweile hauptsächlich mikro-elektro-mechanische Systeme (MEMS)-Gyroskope benutzt. In der vorliegenden Arbeit wurden mikromechanische Gyroskope vom Typ ADXRS300 [31] der Firma Analog Devices verwendet. Bei diesen Gyroskopen gibt es keine rotatorische Schwungmasse, die eine Corioliskraft hervorrufen kann, sondern zwei vibrierende Strukturen, die elektrostatisch in Resonanz gebracht werden. Diese Strukturen rufen bei einer Drehrate, ähnlich der rotatorischen Schwungmasse eines mechanischen Gyroskops, eine Corioliskraft hervor. Am äußeren Ende dieser Strukturen sind orthogonal zur Vibrationsrichtung bewegliche Messfühler angebracht. Diese bewegen sich durch die Coriolisbewegung zwischen weiteren, fest angebrachten Fühlern. Zwischen den Fühlern wird eine Kapazität gemessen, durch die wiederum die ursprüngliche Corioliskraft und damit auch auf die Drehrate des Gyroskops berechnet werden kann.

Mikromechanische Gyroskope, wie sie heute eingesetzt werden, haben gegenüber den oben genannten mechanischen Gyroskopen den großen Vorteil, dass sie wesentlich kleiner produziert werden können und dass sie im Gegensatz zu ihren mechanischen Vorgängern praktisch geräuschlos arbeiten.

Datenausgang

Der Datenausgang eines MEMS-Gyroskops geschieht über eine Leitung, die je nach Drehwinkelgeschwindigkeit eine bestimmte Spannung führt. Vom Hersteller wird jeweils eine Minimal- und Maximalgrenze für die ausgehende Spannung und damit auch der proportionalen Drehwinkelgeschwindigkeit angegeben. Außerdem gibt der Hersteller auch eine Spannung für eine neutrale Drehwinkelgeschwindigkeit an.

Die Mittelstellung des Gyroskops ist aber minimal zum einen von der Fertigungstoleranz und zum anderen von der Temperatur des Gyroskops abhängig. Aus diesem Grund ist es nötig, dass die Mittelstellung des Gyroskops direkt vor der Verwendung gemessen wird. Dazu muss das ganze System in völliger Ruhe stehen, während über einen gewissen Zeitraum die Drehwinkelgeschwindigkeit gemessen wird. Das Ausgangssignal des Gyroskops kann über ein normales Signalrauschen verfälscht sein, weswegen bei der Initialisierung über einen Zeitraum das mittlere Signal gemessen werden muss. Im Idealfall sollten die Signalkabel vom Gyroskop bis zum verarbeitenden A/D-Wandler geschirmt sein, um das Signalrauschen zu minimieren.

Drift

Wenn die im vorigen Kapitel beschriebene Initialisierung des Gyroskops nicht absolut exakt durchgeführt wird, kommt es bei jeder Messung der Drehwinkelgeschwindigkeit zu einem Fehler, da nicht der exakte Nullpunkt der Drehwinkelgeschwindigkeit als Referenz herangezogen wird.

Über die Systemlaufzeit wird dieser Messfehler dann auf Dauer aufintegriert. Hierbei spricht man von einem *Sensordrift*. Dieser Drift könnte nur vermieden werden, wenn die Neutralstellung des Gyroskops absolut exakt gemessen wird. Leider ist das Sensorsignal des Gyroskops verrauscht, und das Gyroskop unterliegt während des Betriebs Temperaturschwankungen. Das Sensorsignal wird auch von einem analogen zu einem digitalen Signal gewandelt, wobei es diskretisiert werden muss und jeder Wert auf die nächstgelegene Stufe gerundet wird.

Eine genaue Bestimmung des Neutralpunkts eines Gyroskops ist in der Regel nicht ohne weiteres möglich, so dass es über einen längeren Zeitraum stets zu einem Drift kommt. Dadurch, dass bei kleinen Robotern und UAVs nur beschränkte Ressourcen im Sinne von Nutzlast, Stromversorgung und Rechenkapazität zur Verfügung stehen, fällt der genannte Drift zudem höher aus, als in anderen Systemen, wo wesentlich mehr Ressourcen zur Verfügung stehen.

Mit Hilfe eines Gyroskops kann ein System also je nach Typ und vorhandenen Ressourcen nur über einen bestimmten Zeitraum für die Stabilisierung eingesetzt werden, da das System ansonsten über die Zeit durch den Drift instabil werden würde. Um ein System bezüglich der Drehwinkelgeschwindigkeit dauerhaft zu stabilisieren, müssen zusätzlich externe Sensoren zum Einsatz kommen.

4.1.2 Beschleunigungssensoren

Beschleunigungssensoren werden in der vorliegenden Arbeit nicht verwendet. Dennoch soll an dieser Stelle kurz darauf eingegangen werden, da sie ebenfalls zu den Inertialsensoren gehören und als solche auch oft zusammen mit Gyroskopen auf kompletten IMU-Boards verkauft werden. Ausserdem wird

im Ausblick am Ende dieser Arbeit eine Erweiterung angesprochen, bei der diese Sensren zum Einsatz kommen könnten.

Beschleunigungssensoren messen in Richtung einer Achse die lineare Beschleunigung. Mit Hilfe dreier, paarweise rechtwinklig angeordneter Beschleunigungssensoren kann also ein beliebiger 3D-Beschleunigungsvektor aufgestellt werden. Um einen Geschwindigkeits- oder Positionsvektor zu erhalten muss ein, beziehungsweise zweimal nach der Zeit integriert werden.

Genau wie bei den Gyroskopen haben auch hier die MEMS-Sensoren eine große Bedeutung. Das Prinzip dieser Sensoren sind Feder-Masse-Systeme. Die Masse besteht aus Silicium und die Federn sind hierbei kleinste Silicium-Stege. Zwischen einer Bezugselektrode und dem federnd gelagerten Teil wird eine Kapazität gemessen, die sich bei einer Auslenkung durch eine Beschleunigung verändert.

4.2 Mikrocontroller

Die folgenden Unterkapitel befassen sich mit den Aspekten der Entwicklung von Software auf einem Mikrocontroller, die bei der vorliegenden Arbeit verwendet wurden. Als Quellen dienten die Handbücher der verwendeten Mikrocontroller [32, 33].

4.2.1 Interrupts

Interrupts werden genutzt, um auf einem Prozessor auf bestimmte Ereignisse zu reagieren. In der Regel handelt es sich hierbei um bestimmte Ein- oder Ausgabeereignisse oder um Interrupts, die in festen Zeitintervallen durchgeführt werden sollen.

Beim Auslösen eines Interrupts wird der reguläre Programmfluss unterbrochen, um eine interruptspezifische Routine abzuarbeiten. Der Rechenaufwand dieser Routine sollte gering gehalten werden, damit der Programmfluss möglichst schnell fortgesetzt werden oder ein weiterer Interrupt ausgelöst werden kann.

4.2.2 Timer

Teilweise ist es nötig, auf einem Mikrocontroller Aufgaben zu bestimmten Zeiten durchzuführen, zum Beispiel um Sensordaten in bestimmten Intervallen abzurufen. Die Taktfrequenz des Controllers ist in der Regel zu hoch für solche Aufgaben. Dennoch sollte die Frequenz zum Abarbeiten der Befehle möglichst konstant und stabil sein.

Für solche Aufgaben bieten sich sogenannte Timer an. Hierbei handelt es sich um spezielle Hardwareregister im Mikrocontroller, die von diesem bei jedem Takt und unabhängig vom eigentlich laufenden Programm hochgezählt werden. Sobald es bei dem Register zu einem Überlauf kommt, kann vom Controller automatisch ein Interrupt ausgelöst werden. Dieser kann im Programm behandelt und die gewünschte Aufgabe ausgeführt werden.

Das Timerregister besitzt eine feste Bittiefe b. Wenn das Register nach einem Überlauf immer mit 0 geladen wird, so wird der Interrupt des Überlaufs bei 2^b Takten ausgelöst. Falls diese Zeit für die gewünschte Aktion zu lang ist, so kann das Register auch bei jedem Überlauf mit einem festen Wert v geladen werden. Dann verkürzt sich die Timerdauer auf $2^b - v$ Takte. In der Regel ist diese Zeitdauer jedoch zu kurz. Für diesen Fall gibt es in Mikrocontrollern die Möglichkeit, einen sogenannten Vorteiler (Prescaler) ϕ zu wählen. Das Timerregister wird dann nicht bei jedem Takt des Controllers inkrementiert sondern je nach Wahl des Prescalers nur bei jedem ϕ-ten Takt.

Insgesamt ergeben sich also folgende Formeln:

$$f_{timer} = \phi(2^b - v) \tag{4.1}$$

$$t = \frac{f_{timer}}{f_{\mu C}}. \tag{4.2}$$

Hierbei ist f_{timer} die Anzahl der Taktschritte des Mikrocontrollers, nach der die Timerfunktion aufgerufen wird, t die Zeit, die zwischen zwei Timeraufrufen verstreicht und $f_{\mu C}$ die Taktfrequenz des Mikrocontrollers.

Hardwarebedingt können nur bestimmte Prescaler zum Einsatz kommen, die vom Hersteller des Mikrocontrollers vorgegeben werden. Mit Hilfe einer bestimmten Bitkombination bei der Initialisierung des Timers wird ein Prescaler ausgewählt. Sollte die Frequenz, die sich durch den Prescaler und die Bittiefe des Timerregisters ergibt, noch immer zu hoch sein, so kann man in der Behandlungsroutine entsprechend die gewünschte Aktion nur bei jedem ϕ-ten Aufruf durchführen.

4.2.3 A/D-Wandler

In der Sensortechnik werden Signale in der Regel als kontinuierliche, analoge Signale ausgegeben. Diese Signale müssen erst in digitale Daten umgerechnet werden, bevor sie von einem Prozessor verarbeitet werden können. Für eine solche Aufgabe werden A/D-Wandler verwendet.

Die Umrechnung eines kontinuierlichen Wertes in eine digitale Größe geschieht durch eine Quantisierung. Das heisst, dass der ursprüngliche Messbereich in verschiedene, in der Regel äquidistante Bereiche unterteilt und jedem Bereich ein digitaler Code zugewiesen wird. Wenn sich nun ein Analogwert innerhalb eines solchen Bereichs befindet, so wird ihm der zugehörige digitale Code zugewiesen.

Das kontinuierliche Signal wird dabei aber, je nach Auflösung des A/D-Wandlers und der Variablen, in der der Wert gespeichert wird, auf diskrete Werte reduziert. Durch eine A/D-Wandlung entsteht immer ein Informationsverlust.

4.2.4 UART

Da der Mikrocontroller weder über eine Tastatur zur Eingabe noch über einen Bildschirm zur Ausgabe verfügt, wird eine Kommunikation mit anderen Geräten nötig, die über solche Funktionen verfügen. In der Regel wird diese Kommunikation über eine oder mehrere serielle (UART)-Schnittstellen gewährleistet. Diese Schnittstellen halten sich an den Radio Sector 232 (RS-232)-Standard.

Dieser Standard schreibt auch die Signalpegel von $+3V$ bis $+15V$ für eine logische 0 und $-3V$ bis $-15V$ für eine logische 1 vor. Die in der vorliegenden Arbeit verwendeten Mikrocontroller werden mit einer Spannung von $5V$ betrieben und prinzipiell werden auf den Signalausgängen eine Spannung von $0V$ für eine logische 0 und $5V$ für eine logische 1 verwendet. Dies widerspricht dem RS-232-Standard. Um diesen Standard für die Kommunikation zum Host-PC verwenden zu können wird ein Pegelwandler benötigt. Bei den verwendeten Mikrocontrollern kommen zu diesem Zweck auf den Versuchsplatinen Schaltungen mit den Pegelwandlern MAX3313E [34] beziehungsweise MAX3221E [35] zum Einsatz. Mit Hilfe dieser Schaltungen kann die Kommunikation zum Host-PC ohne weitere Änderungen geschehen.

Die Kommunikation geschieht in der Regel asynchron und eventuell simultan in beide Richtungen. Während der Mikrocontroller Daten an einen Host-PC sendet, kann er über die gleiche Schnittstelle auch Daten von diesem erhalten. Damit eine Kommunikation zustande kommen kann, müssen sowohl auf dem Host als auch auf dem Mikrocontroller die gleichen Einstellungen von Baudrate, Datenbits, Parität, Stopbits und der Flusssteuerung gemacht werden.

Auf beiden Seiten der Schnittstelle werden die empfangenen Zeichen in einen Cache geschrieben und können dort nach dem First in First out (FIFO)-Prinzip ausgelesen werden. Das Auslesen des Cache kann entweder zu einer beliebigen Zeit im Programm implementiert sein, oder auf dem Mikrocontroller kann eine Interruptroutine aufgerufen werden, sobald ein Zeichen über die Schnittstelle empfangen wurde. In der Routine kann dann sofort auf die empfangenen Daten reagiert oder diese verarbeitet werden.

4.2.5 Half Duplex UART

Bei der Full-Duplex Kommunikation zwischen dem Mikrocontroller und einem weiteren Endgerät können die Daten zwischen beiden Geräten simultan gesendet und empfangen werden. Bei manchen

Geräten, wie zum Beispiel den Servos AX-12 von Dynamixel, ist keine so hohe Kommunikation zwischen den einzelnen Geräten nötig. Hier kann auf eine serielle Schnittstelle zurückgegriffen werden, bei der kein simultanes Senden und Empfangen der Daten möglich ist. Dadurch kann eine Datenleitung eingespart werden. Dynamixel schlägt in den Handbüchern zu den AX-12-Servos [36] eine Schaltung vor, mit der eine Half Duplex UART-Schnittstelle zu einem Mikrocontroller realisiert werden kann.

Da die Kommunikation hier nur über eine Datenleitung erfolgt, muss gewährleistet sein, dass beide Geräte sich untereinander in der Kommunikation nicht stören. Bei der Kommunikation mit den Servos geschieht das, indem der Mikrocontroller ständig über die Schnittstelle Daten empfängt, bei einem Steuerkommando seine Daten sendet und danach sofort wieder auf Empfang umschaltet. Die Servos im Gegenzug befinden sich auch ständig auf Empfang. Sobald Steuerbefehle über die Schnittstelle empfangen werden, sendet das entsprechende Servo ein Statuspaket zurück, um dann ebenfalls wieder sofort auf Empfang umzustellen.

5 Konzept

Die Aufgabe der vorliegenden Diplomarbeit bestand darin, ein System zu schaffen, das mittels Inertialsensoren den Endeffektor einer kinematischen Kette in seiner Orientierung stabilisiert. Eine Voraussetzung dieses Systems war es, das zugrundeliegende kinematische Modell möglichst einfach konfigurieren zu können, um das System auf verschiedenen mobilen Roboterplattformen nutzen zu können. Zusätzlich sollte es möglich sein, verwendete Hardwarekomponenten mit minimalen Veränderungen im Programmcode ersetzen zu können. Dies bedingt, dass das komplette Programm modular aufgebaut wird und einzelne Module ausgetauscht werden können. Im Anschluss an die Entwicklung eines solchen Stabilisierungsprogramms sollte dieses auch mit Hilfe einer Testumgebung evaluiert werden.

In den folgenden Unterkapiteln wird auf die einzelnen Teile der Aufgabenstellung und den Konzepten dahinter eingegangen. Gerade der Teil, der sich mit dem Stabilisierungsprogramm selbst befasst, fällt hierbei etwas umfangreicher aus, da die Theorie dahinter wesentlich komplexer ausfällt als die darauffolgende Realisierung im Programmcode.

5.1 Datenfluss zwischen den Komponenten

Im Blockschaltbild 5.1 ist der nötige Datenfluss zwischen den Grundkomponenten des Systems aufgeführt. Der Mikrocontroller mit dem laufenden Stabilisierungsprogramm steht in der Mitte der Abbildung. Er bekommt vom Host-PC die Steuerungskommandos c geschickt und sendet seinerseits die Statusnachrichten m an den Host. Von den Gyroskopen erhält der Mikrocontroller über den integrierten A/D-Wandler die gemessenen Drehwinkelgeschwindigkeiten $\omega_1, ...\omega_m$, die in einem gemeinsamen Vektor $\boldsymbol{\omega}$ zusammengefasst werden können. Das Stabilisierungsprogramm muss nun aus diesen Daten und den internen Modellen die für die Stabilisierung des Endeffektors nötigen Sollgeschwindigkeiten $\dot{q}_1, ..., \dot{q}_n$ und -positionen $q_1, ..., q_n$ für die Aktoren berechnen. Diese können auch wieder in die Vektoren $\dot{\mathbf{q}}$ und $\mathbf{q}$ zusammengefasst werden. Optional könnte das Hostsystem mit einer Kamera und zusätzlicher Software ausgestattet sein, die zusätzlich die Verdrehung der Kamera misst. Der Host kann dann eine Langzeitstabilisierung durch diese Software durchführen, indem es neue Sollpositionen der Stabilisierungsservos an das Stabilisierungsprogramm schickt.

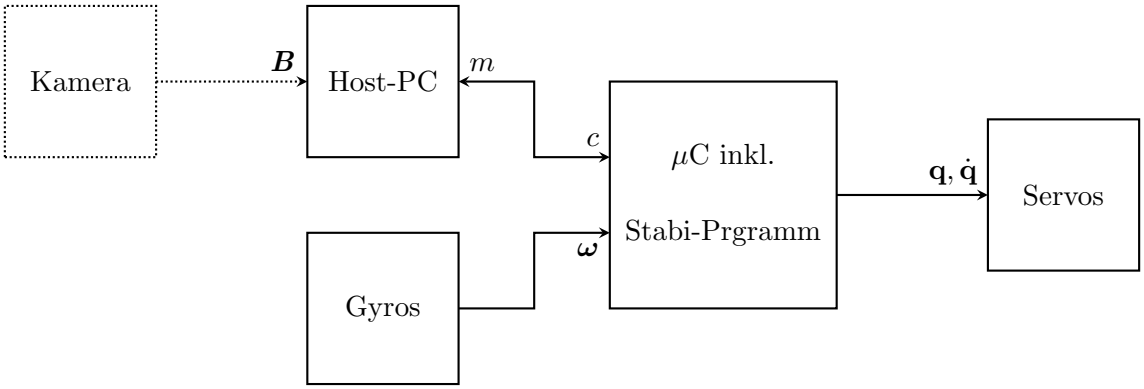

Abb. 5.1: Blockschaltbild des Gesamtsystems

5.2 Stabilisierungsprogramm

Das Stabilisierungsprogramm bildet den Kern der vorliegenden Diplomarbeit. Der Ablauf des Programms verläuft prinzipiell in zwei aufeinanderfolgenden Phasen.

In einer ersten Phase müssen neben den grundsätzlichen Einstellungen und Initialisierungen auch die Gyroskope in Form einer Nullpunktbestimmung initialisiert werden. Erst wenn diese Nullpunkte möglichst genau ermittelt wurden, können die Gyroskope als Sensoren verwendet werden. Das Vorgehen ist hierbei, dass alle Gyroskope in absoluter Ruhe gehalten werden und über eine feste Zeitspanne mehrere Sensorsignale gemittelt werden. Dies entspricht einer Tiefpassfilterung. Diese Filterung ist nötig, da das Sensorsignal mit einem Signalrauschen behaftet ist. Würde man nur einen Wert auslesen, so wäre es wahrscheinlich, dass man durch das Rauschen nicht den exakten Nullpunkt des Gyroskops initialisiert. Eine möglichst exakte Bestimmung ist jedoch notwendig, um den Sensordrift des Gyroskops zu minimieren.

Sobald die Nullpunkte der Gyroskope bestimmt wurden, kann das Programm in der zweiten Phase seiner eigentlichen Bestimmung gemäß verwendet werden. Hierbei werden vom Programm auftretende Drehraten in den Gyroskopen durch dir Aktoren so kompensiert, dass der Endeffektor der kinematischen Kette bestmöglichst in seiner Orientierung gehalten wird.

Hierfür muss das Programm in möglichst kurzen Intervallen eine Reihe von Aktionen ausführen. Zuerst müssen die Drehraten des Gyroskope ausgelesen werden. Diese Drehraten liegen zunächst in Form eines Maßstabes vor, der durch die Eigenschaften des A/D-Wandlers und der Gyroskope vorgegeben ist. Diese Daten werden zunächst in eine intuitivere Dimension $[°/s]$ gebracht. Danach müssen die gemessenen Drehraten in ein gemeinsames Basiskoordinatensystem transformiert werden. Hierbei ist darauf zu achten, dass es zu Abhängigkeiten (siehe 3.2.3) zwischen den gemessenen Sensorwerten kommen kann, falls die Drehachsen der Sensoren nicht paarweise rechtwinklig zueinander stehen. Im

Falle von Abhängigkeiten müssen die redundanten Informationen bei der Transformation verworfen werden.

Sobald die Drehwinkelgeschwindigkeit bezüglich des Basiskoordinatensystems bekannt ist, kann mit Hilfe des inversen Jacobi-Modells die nötige Geschwindigkeit der einzelnen Aktoren bestimmt werden, um eine Stabilisierung des Endeffektors durchzuführen. Nachdem diese Geschwindigkeiten berechnet wurden, können sie direkt mit den neuen Sollpositionden an die Aktoren gesendet werden.

Neben der Stabilisierung soll es auch möglich sein, dass durch ein externes Host-System neue Positionen für die Aktoren vorgegeben werden können. Die Positionsänderung kann gewünscht sein, um die externen Sensoren, die eventuell im Endeffektor der kinematischen Kette installiert sind, neu auszurichten oder um eine Langzeitstabilisierung durch ein weiteres, externes Stabilisierungssystem durchzuführen. Während der Positionsänderung der Aktoren wird die Positionsstabilisierung und -regelung kurzzeitig angehalten und neu gestartet, sobald die Aktoren ihre neue Sollposition erreicht haben.

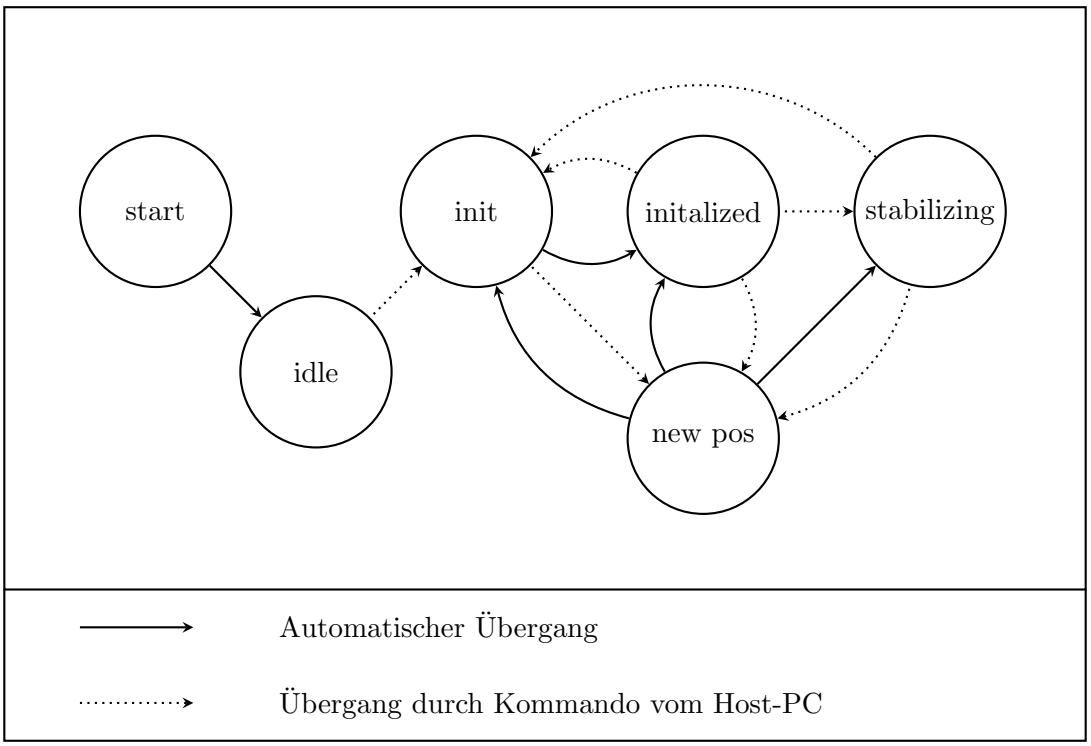

Abb. 5.2: Zustandsübergangsgraph für das Stabilisierungsprogramm

Abbildung 5.2 zeigt das Zustandsübgergangsdiagramm, das als Grundlage für das Stabilisierungsprogramm dienen soll. Hierbei gibt es zwei verschiedene Arten von Transitionen. Zum einen gibt es Übergänge, die automatisch von einem in den nächsten Zustand ablaufen, wenn im ursprünglichen Zustand die jeweilige Aufgabe abgearbeitet wurde. Zum anderen gibt es aber auch Übergänge, die vom Benutzer am Host-PC gestartet werden können.

Das Programm führt direkt beim Start einige interne Variablendeklarationen aus und befindet sich dann in einem Ruhezustand. Sobald der Benutzer das Kommando für die Initialisierung der Gyroskope schickt, geht das Programm in den Initialisierungszustand. Nachdem die Initialisierung abgeschlossen ist, wird automatisch in einen weiteren Ruhezustand umgeschaltet, in dem nun jedoch die Gyroskope initialisiert sind. Nun kann der Benutzer ein Kommando zum Starten der Stabilisierung senden.

Während der Initialisierung der Gyroskope, nachdem die Gyroskope initialisiert wurden und während der Stabilisierung, kann der Benutzer durch ein Kommando vom Host-PC eine neue Initialisierung starten. Zudem hat er die Möglichkeit, in diesen Zuständen die Position der Servos zu ändern. In diesem Fall kehrt das Programm wieder in den ursprünglichen Zustand zurück, nachdem die vom Benutzer vorgegebene Position vom Servo erreicht wurde.

5.2.1 Initialisierungsphase

Wie im vorigen Unterkapitel beschrieben ist es nötig, bei jedem Programmstart die Nullpunkte der Gyroskope zu bestimmen. Dies geschieht durch die Mittelung mehrerer Sensorsignale der Gyroskope, während sich diese in einer Ruhelage befinden. Die Bestimmung eines exakten Nullpunkts ist immens wichtig, um den Sensordrift zu minimieren. Die Bestimmung des Nullpunkts wird genauer, wenn möglichst viele Signale in die Mittelung einfließen. Dies bedingt aber auch eine längere Initialisierungszeit. Um einen möglichst guten Kompromiss zwischen Initialisierungszeit und -genauigkeit zu finden, werden alle Einstellungen und Intervallgrenzen mittels Präprozessoranweisungen definiert, so dass diese möglichst leicht verändert werden können.

5.2.2 Stabilisierungsphase

Zunächst müssen die Sensordaten aus allen Gyroskopen ausgelesen werden. Hierbei wird, wie bei der Bestimmung der Nullpunkte auch, ein Tiefpassfilter zum Einsatz gebracht, um das Signalrauschen zu kompensieren. Genau wie während der Initialisierungsphase werden hierzu mehrere Sensorwerte gemittelt. Auch hier ist eine Präprozessoranweisung definiert, um die Grenze des Filters variieren zu können.

Die Stabilisierung des Endeffektors wird mittels des inversen Jacobi-Modells durchgeführt. Im Programmcode selbst ist der Aufbau der kinematischen Kette hinterlegt, der neben der gewünschten Drehwinkelgeschwindigkeit im Basiskoordinatensystem nötig ist, um das Gleichungssystem in Formel 3.22 nach $\dot{\mathbf{q}}$ zu lösen. Im Fall der Stabilisierung des Endeffektors sollte diese Geschwindigkeit insgesamt 0 sein. Wenn jedoch durch externe Einflüsse eine Winkelgeschwindigkeit $\boldsymbol{\omega}_e$ im Endeffektor anliegt, so muss diese Geschwindigkeit durch die Aktoren ausgeglichen werden.

5.2.3 Transformation der Sensordaten

Die Gyroskope messen die Drehwinkelgeschwindigkeit um eine feste Achse. Zur Stabilisierung des Systems muss die Drehwinkelgeschwindigkeit im Basiskoordinatensystem der kinematischen Kette bekannt sein. Um eine Regelung durchführen zu können und um eine möglichst große Konfigurierbarkeit des Systems zu gewährleisten, muss man jedoch davon ausgehen, dass die Gyroskope nicht fest im Basiskoordinatensystem der kinematischen Kette installiert sind.

Die Drehwinkelgeschwindigkeit kann, wie in Kapitel 3.2.1 beschrieben, in andere Koordinatensysteme transformiert werden. Die Transformationen, die durch die kinematische Kette entstehen, bewirken eine Aneinanderreihung von fest vorgegebenen und durch die Gelenke variablen Rotationen.

Notation 2 *Bei Denavit-Hartenberg werden jeweils vier Transformationsmatrizen zu einer Transformationsmatrix* ${}^{i-1}T_i$ *zusammengefasst, die die Transformation von Gelenk* $i-1$ *zu Gelenk* i *bezüglich des Gelenks* $i-1$ *beschreibt. Da es sich hierbei auch um translatorische Transformationen handelt, die keinen Einfluss auf die Orientierung haben, müssen bei der vorliegenden Arbeit jeweils nur zwei Teilmatrizen betrachtet werden. Dies sind eine feste Rotationsmatrix* ${}^{i-1}R_{f,i}$*, die die Orientierung der Drehachse von Gelenk* i *bezüglich des Bezugskoordinatensystems* S_{i-1} *beschreibt und vor dem Programmstart vorberechnet werden kann. Zusätzlich gibt es für jedes Gelenk eine Rotationsmatrix* ${}^{f,i}R_i(q_i)$*, die die Rotation des* i*-ten Gelenks beschreibt und von der Gelenkwinkelstellung dieses Gelenks abhängig ist. Die Matrix* ${}^{f,i}R_i(q_i)$ *muss während der Laufzeit vom Mikrocontroller aufgestellt werden. Als Gesamtrotationsmatrix vom* $i-1$*-ten zum* i*-ten Gelenk ergibt sich also die Matrix*

$$ {}^{i-1}R_i = {}^{i-1}R_{f,i} \cdot {}^{f,i}R_i(q_i) \tag{5.1} $$

Als Formel ausgedrückt kann für die Transformation der Drehwinkelgeschwindigkeit von Gyroskop j ins Basiskoordinatensystem folgender Term aufgestellt werden:

$$ {}^{0}\boldsymbol{\omega}_{g,j} = {}^{0}R_{f,1} \cdot {}^{f,1}R_1(q_1) \cdot {}^{1}R_{f,2} \cdot {}^{f,2}R_2(q_2) \cdot \ldots \cdot {}^{m-1}R_{f,m} \cdot {}^{f,m}R_m(q_m) \cdot \boldsymbol{\omega}_{g,j} \tag{5.2} $$

Hierbei ist $\boldsymbol{\omega}_{g,j}$ die gemessene Drehwinkelgeschwindigkeit des j-ten Gyroskops, $R_{g,i}(q_i)$ die von der Gelenkwinkelstellung (q_i) abhängige und zur Laufzeit berechenbare Rotationsmatrix und $R_{f,i}$ eine feste Rotationsmatrix, die die Transformation zwischen der Drehachse des $i-1$-ten und i-ten Gelenks beschreibt. Das j-te Gyroskop ist am m-ten Glied der kinematischen Kette angebracht. Das Ergebnis ${}^{0}\boldsymbol{\omega}_j$ ist dann die Drehwinkelgeschwindigkeit des j-ten Gyroskops bezüglich des Basiskoordinatensystems.

Die Gyroskope sollten rechtwinklig zueinander angeordnet und fest miteinander verbungen sein, so dass es zu keinen Abhängigkeiten kommen kann. Sobald zwischen zwei Gyroskopen ein Gelenk liegt, ist diese rechtwinklige Anordnung während des laufenden Betriebs nicht mehr zwangsweise gegeben. Die

Gyroskope sollten also gemeinsam fest an einem Glied der kinematischen Kette installiert werden, ohne dass sich Gelenke zwischen den einzelnen Gyroskopen befinden. Dadurch wird zum einen das Problem der Abhängigkeiten zwischen den einzelnen gemessenen Drehwinkelgeschwindigkeiten minimiert und zum anderen kann dadurch auch bei der Transformation Rechenzeit gespart werden, indem man für alle Gyroskope eine gemeinsame Gesamtrotationsmatrix berechnet:

$$R_{ges,i} = {}^{0}R_{f,1} \cdot {}^{f,1}R_1(q_1) \cdot {}^{1}R_{f,2} \cdot {}^{f,2}R_2(q_2) \cdot ... \cdot {}^{m-1}R_{f,m} \cdot {}^{f,m}R_m(q_m) \tag{5.3}$$

Falls alle Gyroskope am Endeffektor angebracht sind und die Anzahl der Glieder der kinematischen Kette n ist, so gilt zudem $m = n$.

5.2.4 Redundanzen entfernen

Ein größeres Problem als die simple Transformation stellt die Abhängigkeit von Drehwinkelgeschwindigkeiten dar. Zu diesen Abhängigkeiten kommt es, wie in Kapitel 3.2.3 beschrieben, falls die Achsen, um die gemessen wird, nicht direkt orthogonal zueinander stehen. Bei einem Manipulatorarm mit Drehgelenken in allen drei Raumrichtungen und jeweils einem Gyroskop an einem Glied kommt es zwangsläufig zu Abhängigkeiten oder gar dem Verlust eines Sensors, wenn sich alle drei Gelenke bewegen.

Die Abhängigkeiten zwischen den einzelnen Gyroskopen bedeuten auch gleichzeitig eine Redundanz der Daten. Sollte man einfach alle gemessenen Drehwinkelgeschwindigkeiten aufaddieren, so kann es zu einer Verfälschung der Messungen kommen. In Kapitel 3.2.3 ist eine ungünstig gewählte Manipulatorkonfiguration beschrieben. Bei einer Transformation in ein Koordinatensystem und anschließendem Addieren der Drehwinkelgeschwindigkeiten beider Gyroskope würde man eine Drehwinkelgeschwindigkeit erhalten, die im ungünstigsten Fall doppelt so hoch ist wie die tatsächlich anliegende Winkelgeschwindigkeit.

Um die redundanten Daten zwischen den einzelnen Gyroskopmessungen herauszufiltern, muss man zuerst alle Messungen in ein gemeinsames Koordinatensystem transformieren. Danach muss sukzessive, beginnend mit der am nächsten zum Endeffektor gemessenen Drehrate $\boldsymbol{\omega}_{g,m}$ der rechtwinklige Anteil aller weiteren Drehwinkelgeschwindigkeiten $\boldsymbol{\omega}_{g,m-1}, ..., \boldsymbol{\omega}_{g,1}$ aufaddiert werden. Die Berechnung startet mit dem Startwert $\boldsymbol{\omega}_0 = \boldsymbol{\omega}_{g,m}$. Nun wird für jede weitere Sensormessung nur jeweils der Teil von $\boldsymbol{\omega}_i$ aufaddiert, der rechtwinklig zur aktuellen Gesamtsensormessung $\boldsymbol{\omega}_0$ steht. Berechnet wird dieser rechtwinklige Anteil durch $\boldsymbol{\omega}_{i,r} = \boldsymbol{\omega}_i - \langle \boldsymbol{\omega}_i, \boldsymbol{\omega}_0 \rangle \cdot \boldsymbol{\omega}_0$.

Durch diesen Algorithmuss werden unter Umständen zu viele Daten verworfen. Bei einer ungünstigen Manipulatorkonfiguration ist dies jedoch nicht zu vermeiden.

Desweiteren kommt es dazu, dass hierbei das System einen Vorzugssensor besitzt. Der Sensor, der zuletzt ausgelesen und addiert wird, geht immer in vollem Maße in die Rechnung ein, während von den anderen Sensoren unter Umständen Daten verworfen werden, die nicht rechtwinklig zu diesem letzten Sensor stehen. Die Reihenfolge wurde hier so gewählt, da hierbei der Sensor als Vorzugssensor verwendet wird, der zum Endeffektor der kinematischen Kette am nächsten steht. Da der Endeffektor stabilisiert werden soll, sollte dieser Sensor dann entsprechend die genauesten Daten liefern und am wenigsten vom Spiel in den Gelenken verfälscht sein. Dennoch ist es denkbar, dass man den Algorithmus auch beginnend vom Endeffektor starten lässt. Dann würde als Vorzugssensor derjenige Sensor dienen, der am nächsten an der Basis liegt.

5.2.5 Transformation der Drehachsen

Neben den Drehwinkelgeschwindigkeiten müssen, zum Aufstellen der Jacobi-Matrix, auch die Drehachsen der Aktuatoren ins Basiskoordinatensystem transformiert werden. Dies geschieht für den k-ten Aktuator nach der Formel:

$$ {}^{0}\mathbf{e}_{k-1} = {}^{0}R_{f,1} \cdot {}^{f,1}R_1(q_1) \cdot {}^{1}R_{f,2} \cdot {}^{f,2}R_2(q_2) \cdot \ldots \cdot {}^{k-2}R_{f,k-1} \cdot \mathbf{e}_z \tag{5.4} $$

Hierbei sind, wie bei der Transformation der Drehwinkelgeschwindigkeiten auch, $R_{f,i}$ und $R_{g,i}$ die feste Rotationsmatrix der Drehachsen zwischen dem $i-1$-ten und dem i-ten Gelenk sowie die Rotationsmatrix, die die Gelenkwinkelstellung des i-ten Gelenks beschreibt. $\mathbf{e}_z$ ist nach Denavit-Hartenberg die Drehachse des Gelenks also der Einheitsvektor $(0,0,1)^T$. Das Ergebnis ${}^{0}\mathbf{e}_{k-1}$ ist die ins Basiskoordinatensystem transformierte Drehachse des k-ten Gelenks.

Um nun Rechenzeit zu sparen, kann man statt der z-Achse eine beliebige Achse als Drehachse für die einzelnen Gelenke zulassen und kann die letzte Matrix-Vektor Multiplikation ($\mathbf{d}_{k-1} = R_{f,k-1} \cdot \mathbf{e}_z$) vorberechnen. Somit erhält man die leicht modifizierte Formel:

$$ {}^{0}\mathbf{e}_{k-1} = {}^{0}R_{f,1} \cdot {}^{f,1}R_1(q_1) \cdot {}^{1}R_{f,2} \cdot {}^{f,2}R_2(q_2) \cdot \ldots \cdot {}^{k-3}R_{f,k-2} \cdot {}^{f,k-2}R_{k-2}(q_{k-2}) \cdot \mathbf{d}_{k-1}. \tag{5.5} $$

5.2.6 Inverses Jacobimodell

Sobald alle gemessenen Drehwinkelgeschwindigkeiten ins Ursprungskoordinatensystem transformiert und die Redundanzen entfernt wurden, kann mit Hilfe des inversen Jacobimodells die Stabilisierung durchgeführt werden. Nach diesen Schritten liegt ein Drehwinkelgeschwindigkeitsvektor im Basiskoordinatensystem vor. Dieser Vektor beschreibt die Rotation, in der sich das System beziehungsweise die dort installierten Gyroskope befinden. Um das System zu stabilisieren, sollte im nächsten Moment eine genau entgegengesetzte Rotation wirken. Dies erreicht man, indem man den gemessenen Vektor komponentenweise negiert.

Zusätzlich zur gewünschten Drehwinkelgeschwindigkeit benötigt man für die Berechnung des inversen Jacobimodells die Orientierung der Drehachsen der Sensoren. Die Berechnung dieser Drehachsen geschieht, wie in Kapitel 5.2.5 beschrieben. Nachdem alle Drehachsen bezüglich des Basiskoordinatensystems berechnet wurden, kann der untere, für die Orientierung verantwortliche Teil der Jacobimatrix des Systems aufgestellt werden.

Wenn drei oder mehr Drehachsen in der kinematischen Kette vorhanden sind, so kann aus der Jacobimatrix und der nötigen, vorher berechneten Drehwinkelgeschwindigkeit für die Stabilisierung das Gleichungssystem

$$^{0}\boldsymbol{\omega}_{n} = {}^{0}J_{n,\omega} \cdot \dot{\mathbf{q}} \tag{5.6}$$

aufgestellt werden. Hierbei ist $^{0}\boldsymbol{\omega}_{n}$ die nötige Drehwinkelgeschwingikeit für die Stabilisierung, $^{0}J_{n,\omega}$ der untere Teil der Jacobimatrix und $\dot{\mathbf{q}}$ der zu berechnende Vektor der Drehgeschwindigkeiten in den Gelenken.

Sofern in der Manipulatorstellung keine Singularität entsteht und somit Freiheitsgrade verloren gehen, ist das Gleichungssystem auf diese Weise lösbar. Sollten mehr als drei Drehachsen im Manipulator vorhanden sein, so kann man solche Singularitäten einfacher umgehen. Jedoch ist dann im Normalfall eine Beschränkung auf drei Servos nötig, die sich in einem Schritt bewegen sollen. In der Regel können hier drei beliebige Servos gewählt werden, sofern dies dann nicht zum Dimensionsverlust in der Jacobimatrix führt. In Kapitel 3.4.1 ist beschrieben, wie im Falle von Singularitäten oder generell einem überbestimmten Gleichungssystem verfahren werden kann. Mit Hilfe der Methode der kleinsten Quadrate kann der Fehler beim Lösen dieser Gleichungssysteme minimiert werden.

5.2.7 Driftkompensation und Richtungsänderung

In Kapitel 4.1.1 ist beschrieben, dass es bei den in dieser Arbeit zum Einsatz kommenden internen Sensoren zu einem unvermeidbarem Drift kommt. Gyroskope können nicht zur Langzeitstabilisierung zum Einsatz kommen, da sie aus der Umwelt keine feste Referenz zu Nutze ziehen können. Falls der Nullpunkt der Gyroskope bei der Initialisierung nicht exakt gefunden wurde, so werden sich die Aktuatoren auch mit einer vorhandenen Stabilisierung auf Dauer in eine Richtung bewegen. Mit Hilfe eines weiteren, externen Stabilisierunggystems wäre es möglich, zum Beispiel mit Hilfe einer Kamera, den Drift zu ermitteln und eine neue Servoposition an das vorliegende Stabilisierungsprogramm zu senden.

Desweiteren ist beim Einsatz des Systems in einem Roboterkopf nicht nur eine Richtungsstabilisierung gewünscht. Vielmehr möchte man die Aktuatoren zwischen Roboterkörper und Kamera dazu nutzen,

um die Umwelt zu beobachten. So ist es zum Beispiel denkbar, dass man mit der Kamera ein Objekt verfolgen oder aus anderen Gründen den Kopf bewegen möchte. Wenn mit den Aktuatoren ausschließlich eine Stabilisierung durchgeführt wird, so wäre diese Aufgabe von einem solchen System nicht zu lösen, da es immer nur in eine feste Richtung schaut.

Um nun das Driftproblem und eine Steuerung der kinematischen Kette zu gewährleisten, ist es möglich, von einem externen System ein Kommando an das Stabilisierungsprogramm zu schicken, um die Aktuatoren an eine bestimmte Stelle fahren zu lassen.

Während normalerweise nach dem Lösen des inversen Jacobi-Modells direkt die neuen Sollgeschwindigkeiten und -positionen an die Aktoren gesendet werden, wird nun die Regelung und Stabilisierung kurzzeitig unterbrochen. Danach werden die erhaltenen Aktuatorstellungen direkt und ohne Veränderung an die Servos weitergeleitet. Sobald die Servos die gewünschte Position erreicht haben, wird die komplette Stabilisierung mit den vorherigen Initialisierungswerten wieder gestartet.

5.3 Konfiguration des kinematischen Modells

Ein Teil der Aufgabe war es, ein möglichst flexibles System zu schaffen, was die Anordnung von Aktuatoren und Sensoren angeht. Einem Benutzer sollte es möglichst einfach gemacht werden, die Konfiguration des Manipulators in Form von Transformationen zwischen den einzelnen Gelenken und Sensoren definieren zu können. Da der Speicherplatz und die Rechenleistung eines Mikrocontrollers stark begrenzt ist, ist es nicht ratsam, ein universelles Programm zu entwickeln, das über verschiedene Einstellungen zur Laufzeit die einzelnen Optionen durchläuft. Vielmehr wird hier ein leichtgewichtiges Programm nötig, das für das jeweilige kinematische Modell speziell angepasst auf den Controller geladen wird.

Die Beschränkung des Programms auf speziell ein kinematisches Modell widerspricht der Anpassung hinsichtlich verschiedener Robotermodelle. Um wirklich stets ein spezielles Programm auf den einzelnen Robotern laufen lassen zu können, müsste im Programmcode eine nichttriviale Änderung gemacht werden. Aus diesem Grund soll ein Konfigurationsprogramm entwickelt werden, mit dessen Hilfe man intuitiv das kinematische Modell sowie die Lage und Anordnung der Sensoren eingeben kann. Im Anschluss ist es dann möglich, mit Hilfe des Programms den nötigen Code für das Stabilisierungsprogramm automatisch generieren zu lassen.

Es wäre wünschenswert, wenn das Konfigurationswerkzeug unter allen gängigen Betriebssystemen wie Windows, Linux oder MacOS eingesetzt werden könnte. Aus diesem Grund fiel die Wahl der Programmiersprache auf Java. Sobald die Java-Laufzeitumgebung (JRE) auf dem Rechner installiert ist, kann innerhalb kürzester Zeit und ohne große Installationsroutinen das Konfigurationswerkzeug benutzt werden.

5.4 Modularität

Bei der Entwicklung von Robotersystemen kommt es häufig dazu, dass im Laufe der Zeit verschiedene Hardwarekomponenten durch andere Typen ausgetauscht werden, sei es, dass eine Firma ein Produkt nicht mehr weiter vertreibt, weil bereits ein Nachfolgemodell auf dem Markt ist oder weil es auf dem Markt ein anderes Modell eines anderen Herstellers mit besseren Eigenschaften verfügbar ist. In der Regel sind dann Anpassungen am Programmcode vorzunehmen, da die einzelnen Typen nicht hundertprozentig kompatibel zueinander sind. Das entwickelte Stabilisierungsprogramm sollte deshalb möglichst modular aufgebaut werden, so dass einzelne Codeteile für die jeweiligen Hardwarekomponenten spezifisch programmiert werden und diese gezielt ausgetauscht werden können.

Mit einer objektorientierten Programmiersprache wie C++ oder Java wäre es möglich, mittels Vererbung und Interfaces für jede Hardwarekomponente eine Klasse zu programmieren, die dann ausgetauscht werden kann, sobald sich an der Hardware dahingehend etwas ändert, dass neuer Programmcode nötig ist. Für Mikrocontroller stellen die Hersteller jedoch teilweise lediglich Bibliotheken in reinem C-Code zur verfügung. Des weiteren ist das zu entwickelnde Stabilisierungsprogramm nicht so komplex, dass man zwingend eine Entwicklungsumgebung mit C++ verwenden müsste. Deswegen soll eine prozedurale Schnittstelle entwickelt werden. Die Module in der vorliegenden Arbeit werden so aussehen, dass für jede Hardwarekomponente ein separates Dateipaar (Header- und C-Datei) entwickelt wird. Der Code kann dann dahingehend einfach ausgetauscht werden, dass einfach eine andere Header-Datei im Hauptprogramm eingebunden wird. Die Hauptaufgabe besteht hierbei darin, dass die Schnittstellen, in Form von Funktionsdeklarationen, der einzelnen Komponenten genau definiert und beschrieben werden und diese dann beim Erstellen von neuem Code auch eingehalten werden.

In Abbildung 5.3 sind die Modelle aufgeführt, die für das Stabilisierungsprogramm benötigt werden. Zum einen sind dies die Implementierungen für die einzelnen Hardwarekomponenten (Mikrocontroller, Gyroskope und Servomotoren), die in Form von Quellcode vorhanden sein müssen. Da es nicht abzusehen ist, welche Merkmale andere Komponenten in Zukunft eventuell haben könnten, kann dieser Quellcode nicht im Voraus und mit Parameteränderungen erstellt werden. Dazu wird auch das kinematische Modell des Manipulators benötigt. Dieses soll, wie bereits in Kapitel 5.3 beschrieben, durch ein Konfigurationswerkzeug verändert werden können. Der nötige Quellcode wird dann nach den Benutzereingaben automatisch von diesem Werkzeug generiert.

5.5 Testsystem und Evaluierung

Der Kern der vorliegenden Arbeit ist das zu entwickelnde Stabilisierungsprogramm. Dieses muss während der Entwicklung und auf im Nachhinein auf seine Funktionsfähigkeit getestet werden. Für diesen Zweck wird ein Testsystem bestehend aus mehreren Servos und Gyroskopen entwickelt. Diese

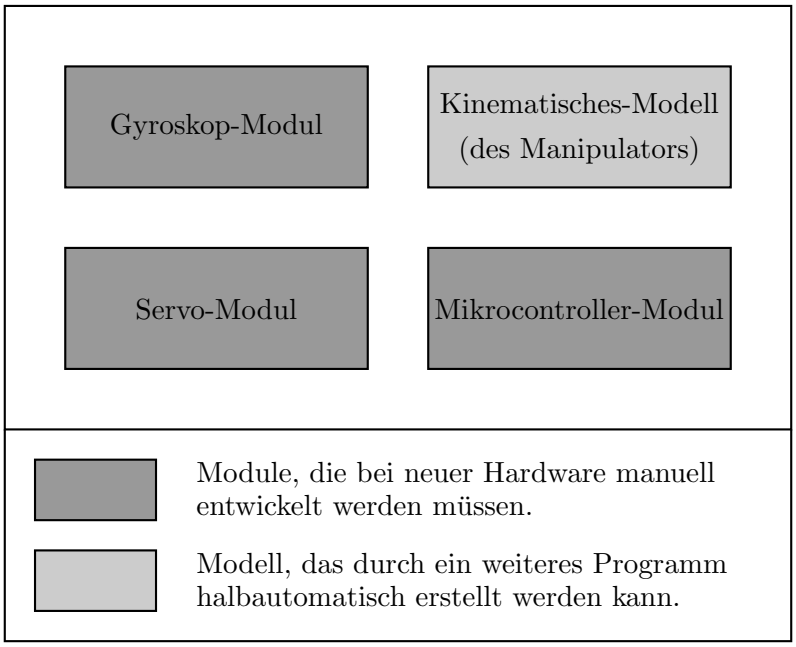

Abb. 5.3: Im Programm hinterlegte Modelle / Module

Aktoren und Sensoren werden an einen Mikrocontroller angeschlossen, auf dem auch das Stabilisierungsprogramm ausgeführt wird.

Nachdem das Stabilisierungsprogramm entwickelt wurde, ist das ganze System zu evaluieren, um die Genauigkeit und Geschwindigkeit der Stabilisierung zu bestimmen. Zum einen muss überprüft werden, wie genau das System nach der Stabilisierung wieder in den Ursprungszustand fährt. Nachdem eine Lagestabilisierung durchgeführt wurde und das System sich danach wieder in Ruhe befindet, sollte es wieder in die gleiche Richtung schauen wie vor der Stabilisierung. Durch den Drift der Gyroskope wird der Endeffektor voraussichtlich nicht genau die Orientierung besitzen, wie zu Beginn des Tests. Neben dieser Rückstellgenauigkeit soll auch überprüft werden, wie schnell das System die Lagestabilisierung durchführen kann. Das System kann nur auf eine, durch die Gyroskope gemessene, Orientierungsänderung reagieren. Dies geschieht zeitverzögert, da die Signale ausgewertet werden müssen und danach erst die Steuersignale an die Aktuatoren gesendet werden können. Es muss also auch während der Stabilisierung überprüft werden, wie schnell das ganze System die Lagestabilisierung durchführen kann und wie groß hierbei die abweichenden Winkel sind, die auftreten können. Es ist davon auszugehen, dass die Geschwindigkeit der Stabilisierung entscheidend von der Qualität des verwendeten Reglers abhängig ist.

Die Gyroskope besitzen einen Sensordrift. Dieser ist bei der Evaluierung zu berücksichtigen. In der Regel wird der Nullpunkt der Gyroskope bei deren Initialisierung nicht perfekt erreicht werden, so dass immer eine Drehwinkelgeschwindigkeit gemessen wird, auch wenn die Gyroskope selbst sich in einer Ruhelage befinden. Für das Stabilisierungssystem selbst ist dies nicht ohne weitere externe Sen-

soren zu ermitteln, so dass es davon ausgehen muss, dass tatsächlich eine Drehwinkelgeschwindigkeit anliegt. Vor der eigentlichen Evaluierung muss überprüft werden, wie groß der mittlere Sensordrift der Gyroskope ist. Dieser Wert muss bei der Evaluierung berücksichtigt werden, denn dieser Fehler ist immer im System vorhanden.

Das Testsystem ist ausschließlich mit Inertialsensoren ausgestattet, die durch den schon beschriebenen Drift nicht zur Evaluierung des Systems eingesetzt werden können. Die Rückstellgenauigkeit kann jedoch mit dem vorhandenen Testsystem überprüft werden, indem man die Positionen der Servos als Messwert heranzieht. Für einen solchen Test muss die komplette Plattform vor und nach der Stabilisierung in einer fest vorgegebenen Orientierung liegen. Vor der Stabilisierung stehen die Aktuatoren in einer vorher definierten Nullstellung. Wenn sie nach der Stabilisierung und nachdem die Plattform wieder die Ursprungsorientierung erreicht hat, wieder in dieser Nullstellung stehen, so entspricht dies einer idealen Rückstellgenauigkeit. Jede Abweichung von dieser Nullstellung entspräche dann einem Fehler, der zum einen durch den Sensordrift der Gyroskope, zum anderen aber auch durch die Stabilisierung selbst aufgetreten sein kann. Die Servos sind mit Dekodierern ausgestattet, die abtriebsseitig angebracht sind. Sollte ein Spiel in den Getrieben auftreten, so liefern die Dekodierer dennoch die exakte Stellung des Gelenks.

Um diese Tests möglichst genau wiederholen zu können wäre ein übergeordnetes System denkbar, das die komplette Plattform in eine vorher festgelegte Lage kippt. Dafür wären jedoch ziemlich starke Servos nötig, die das komplette Gewicht der Testumgebung halten können. Deswegen ist es denkbar, dass nicht die komplette Plattform durch ein solches System gesteuert wird, sondern nur der nötigste Teil der kinematischen Kette. Man könnte die Kette dahingehend erweitern, dass an der Basis weitere Servos eingefügt werden, die nicht vom Stabilisierungssystem angesprochen werden sondern von einer übergeordneten Steuerungsschleife oder von einem zweiten Mikrocontroller. Diese Servos würden dann eine vorgegebene Trajektorie abfahren. Somit hätte man ein System geschaffen, mit dem reproduzierbare Tests durchgeführt werden können.

Um die Geschwindigkeit des Systems während der Stabilisierung zu messen, wäre es möglich, bei einer bekannten Orientierung des Gesamtsystems die Position der Servos abzufragen und daraus die Orientierung des Endeffektors zu errechnen. Eine weitere Möglichkeit wäre eine externe Referenz heranzuziehen und diese mit einem externen Sensor aufzunehmen, der im Endeffektor installiert ist. Denkbar wäre es, eine Kamera im Endeffektor der kinematischen Kette zu installieren und diese auf ein Muster auszurichten. Während der Stabilisierung kann die Kamera einen Film aufzeichnen, dessen Bilder später ausgewertet werden können. Man könnte dann aus dem Muster die Orientierung des Endeffektors berechnen.

6 Realisierung

Nach dem Abschluss des konzeptionellen Teils der Arbeit wurde mit der Realisierung begonnen. Es wurde in einem ersten Schritt die Hardware zusammengefügt und eine Versuchsplatine gelötet. Nachdem die komplette Hardware, wie gewünscht, so miteinander kommunizieren konnte, wurde mit der Programmierung des Stabilisierungsprogramms begonnen. Während den Arbeiten hat sich gezeigt, dass der Mikrocontroller für die Aufgabe nicht ausreichend war und ausgetauscht werden musste. Nun wurde die Platine durch ein vom Hajime Research Insititute entwickeltes Board ersetzt. Hierauf verrichtet ein schnellerer Controller seine Arbeit. Dieses Board kommt auch momentan bei den Vierbeinern des Fachbereichs zum Einsatz.

Dieses Kapitel beschreibt nun die Realisierung und Entwicklung des Testsystems und des Stabilisierungsprogramms. Es geht auf die gewünschte Teilaufgabe der Modularität des Programms ein und beschreibt dann noch das entwickelte Konfigurationswerkzeug zur Anpassung des kinematischen Modells. Die Tests und die Durchführung der Evaluierung werden später in Kapitel 7 behandelt.

6.1 Testsystem

Für die Entwicklung und die nötigen Tests wurde ein Testsystem bestehend aus einem Mikrocontroller sowie jeweils drei Servos und Gyroskopen entwickelt. In der Entwicklungsphase kam als Mikrocontroller ein ATMega128 von Atmel zum Einsatz, der dann während der Testphase durch den schnelleren SH7125 der Firma Renesas ersetzt wurde. Als Aktuatoren wurden stets Servos vom Typ AX-12 von Dynamixel verwendet. Von der Firma Analog Devices wurden Gyroskope vom Typ ADXRS300 für die Entwicklung und Tests des Stabilisierungssystems verwendet.

Der Code für alle benutzten Hardwarekomponenten steht jeweils in einer Header- und einer Quellcodedatei zur Verfügung. Die einzelnen Hardwarekomponenten mit den zugehörigen Dateien sind nochmals in Tabelle 6.1 aufgelistet. Um eine andere Hardwarekomponente im Testsystem zu verwenden, muss im Quellcode lediglich eine andere Headerdatei eingebunden werden. Für andere Hardwarekomponenten als die hier beschriebenen stehen keine entsprechenden Dateien zur Verfügung. Es ist jedoch möglich,

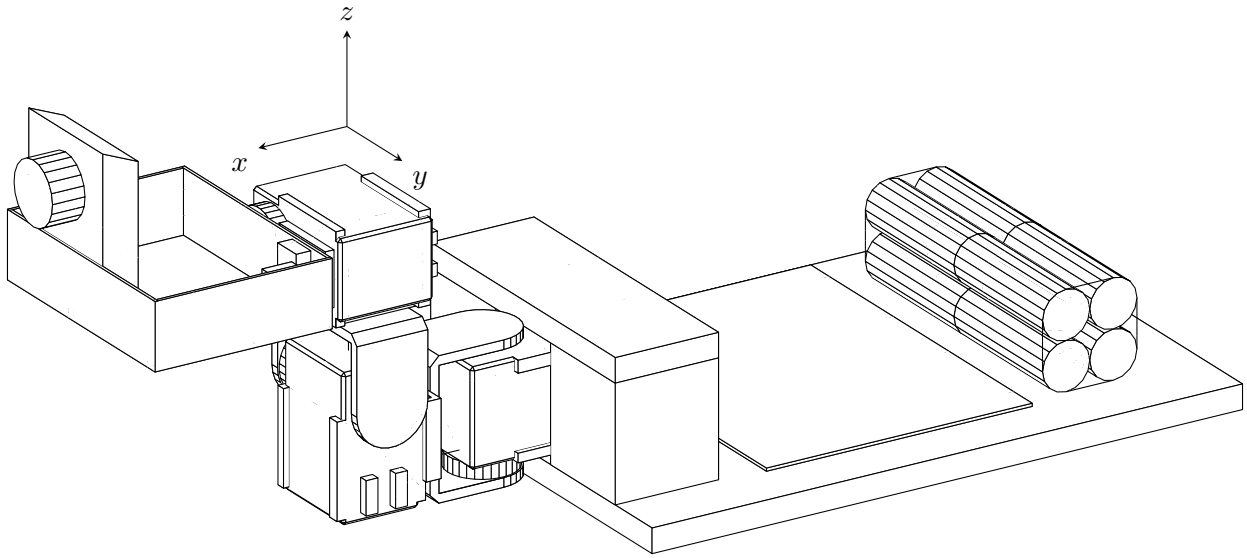

Abb. 6.1: Skizze des Testsystems

den Code für weitere Komponenten zu erstellen. Dafür müssen die Schnittstellen für den gewünschten Komponententyp bei der Implementierung eingehalten werden. Diese Schnittstellen sind in Kapitel 6.3 genau beschrieben.

Hardware	Hardwaretyp	Header	Quellcode
ATMega128	μC	io128.h	io128.c
SH7125	μC	io7125.h	io7125.c
AX-12	Servo	ax12.h	ax12.c
ADXRS300	Gyroskop	adxrs300.h	adxrs300.c

Tab. 6.1: Code-Dateien für alle benutzten Hardwarekomponenten

Die Drehachsen der Aktuatoren im Testsystem sind paarweise orthogonal zueinander angeordnet, so dass das System um drei Raumachsen stabilisiert werden kann. Die drei verbauten Gyroskope wurden direkt am Endeffektor der kinematischen Kette angebracht. Sie wurden so installiert, dass sie ebenfalls jeweils paarweise orthogonal zueinander stehen. Damit können die Drehwinkelgeschwindigkeiten um alle drei Raumrichtungen gemessen werden, und es treten keine Redundanzen auf, die gefiltert werden müssen. Dies bewirkt einen Geschwindigkeitsvorteil bei der Transformierung der Drehrate des Basiskoordinatensystems.

6.1.1 Verwendete Hardware

Im Folgenden wird kurz auf die verwendeten Hardwarekomponenten und deren Besonderheiten und Einstellungen eingegangen, wie sie in der vorliegenden Arbeit verwendet wurden. Bei den beschriebe-

nen Komponenten handelt es sich um ein Gyroskop, einen Servo und zwei verschiedene Mikrocontroller. Die Controller wurden nicht gleichzeitig im System verwendet, sondern nacheinander wurde ein Controller durch den anderen ausgetauscht, da er zum einen über eine höhere Taktfrequenz, aber auch über eine 32 Bit Arithmetik verfügt.

Gyroskop ADXRS300

Der ADXRS300 ist ein einachsiges Gyroskop, das Drehwinkelgeschwindigkeiten von bis zu $\pm 300°/s$ messen kann. Das Gyroskop ist durch seine Bauweise in einem Chip sehr klein, leicht und kostengünstig.

Auf den Ausgängen des Gyroskops liegt jeweils eine Spannung zwischen $0 - 5V$ an, die über einen A/D-Wandler in den Mikrocontroller eingelesen werden kann. Im Datenblatt des Gyroskops sind die Bereiche angegeben, die auf diesen Spannungsbereich abgebildet werden.

Servo AX-12

Als Aktuatoren für die Inertialstabilisierung kommen in dieser Arbeit Servomotoren vom Typ AX-12 der Firma Dynamixel zum Einsatz. Die Servomotoren können mit einer Auflösung von $0,35°$ und einem maximalen Arbeitsbereich von $\pm 150°$ betrieben werden.

Als weitere Features bieten diese Servomotoren die Möglichkeit, mit vorgegebenen Geschwindigkeiten zu fahren, feedback über den Fehlerstatus zu bekommen, einen Broadcast an alle Servos zu schicken oder Ähnliches.

Die Kommunikation zu den Servomotoren geschieht über einen Half Duplex UART-Bus. Hierbei hängen alle Servos mit einer Datenleitung an der UART-Schnittstelle des Mikrocontrollers. Über den Bus kann gezielt mit einzelnen Servos kommuniziert werden, indem jeder Servomotor vor der Installation eine eindeutige Identifikationsnummer (ID) erhält. In den Instruktionspaketen an die Servos ist eine ID angegeben, so dass nur der gewünschte Servo auf die Befehle reagiert. Es gibt aber auch die Möglichkeit, eine Broadcast-ID zu verwenden, so dass alle Servos auf den Befehl ansprechen.

Um die Kommunikation auf dem Datenbus zu minimieren, wurden die Servos so eingestellt, dass sie lediglich Daten an den Mikrocontroller zurücksenden, wenn von diesem das Kommando “READ_DATA” gesendet worden ist. Mit diesem Kommando können Statusdaten, wie zum Beispiel die aktuelle Geschwindigkeit oder Position vom Servo abgerufen werden.

Mikrocontroller ATMega128 / Crumb128 V2.0

Beim Crumb128 handelt es sich um ein Rapid Prototyping Modul mit einem integrierten Atmel ATmega128 Mikrocontroller. Das Board besitzt neben dem Mikrocontroller verschiedene Schnittstellen, die für die Programmierung und Kommunikation mit dem Controller genutzt werden können. Zusätzlich sind alle Pins des Controllers auf externe Stiftleisten geführt. Dort können die Signale dann einfacher auf der Platine abgegriffen werden. Für den Controller ist auf dem Board bereits ein 14,7456MHz Quarz integriert, das die Prozessorfrequenz vorgibt.

Das Board stellt einen speziellen 6-Pin InSystemProgramming (ISP) Port für die Programmierung mittels Programmierkarte zur Verfügung. Weitere Schnittstellen sind ein serielles RS232-, sowie ein USB2.0-Interface zur Kommunikation mit einem externen Host-PC. Über die externen Stiftleisten gibt es die Möglichkeit, auch die zweite serielle UART-Schnittstelle des ATmega128 zu verwenden. Diese Stiftleiste kann auch benutzt werden, um einzelne Steuersignale des Controllers abzugreifen.

Der ATMega128 ist ein 8 Bit Mikrocontroller, der über keine Fließkommaarithmetik verfügt. Sollten Ganzzahlen mit einer höheren Bittiefe als 8Bit oder Fließkommazahlen im Programm verwendet werden, so müssen diese Kommandos durch die Software emuliert werden. Diese Emulierung wird vom Compiler automatisch berechnet. Während der Laufzeit des Programms bedeutet dies jedoch eine längere Rechenzeit.

In der letzten Version, in der der ATMega128 als Mikrocontroller zum Einsatz kam, wurde ein Timer mit den Eintellungen $v = 112$, $b = 8$ und $\phi = 256$ verwendet. Dies ergibt nach Formel 4.2 bei einer Taktrate des Controllers von $f_\mu C = 14.745.600$ eine Zeitdauer von

$$t = \frac{\phi(2^b - v)}{f_{\mu C}} = 2,5ms \tag{6.1}$$

und damit eine Frequenz von $400Hz$. Dies war jedoch noch zu kurz für die Berechnungen. Deshalb wurde nur bei jedem 20-ten Aufruf wirklich die Timerfunktion abgearbeitet. Dies ergibt dann also letztendlich eine Frequenz von $20Hz$, mit der die Stabilisierungsfunktion abgearbeitet wurde.

Mikrocontroller SH7125 / Hajime Controller6

Das Board “Hajime Controller6”, bestückt mit einem SH7125 Mikrocontroller der Firma Renesas, wurde genau für den Einsatz mit AX-12 Servos konzipiert. Es bietet die Möglichkeit, über fünf Stecker mehrere Servos anzuschließen. Durch das Bussystem können mit diesen Servos auch wieder mehrere

Servos verbunden werden. Gleichzeitig bietet das Board die Möglichkeit, über A/D-Wandler sechs Analogsignale zu verarbeiten und zusätzlich Digitalsignale über mehrere Ports auszugeben.

Bei dem SH7125 handelt es sich um einen 32Bit Mikrocontroller, der ebenfalls nicht über eine integrierte Fließkommaarithmetik verfügt. Betrieben wird der Controller in der vorliegenden Version mit einer Taktrate von 32MHz. Durch die höhere Taktrate und die Möglichkeit, größere Ganzzahlen in einem Schritt zu verarbeiten, ist dieser Mikrocontroller um einiges schneller als der ATMega128. Die Frequenz, mit der die Gyroskope ausgelesen werden, kann dadurch wesentlich erhöht werden. Außerdem sind die A/D-Wandler dieses Mikrocontrollers im Vergleich zum ATMega128 schneller einsatzbereit und liefern schneller ein verwertbares Sensorsignal. Dadurch kann die Initialisierungszeit der Gyroskope beschleunigt werden.

Beim Mikrocontroller von Renesas wurde das Konzept vom Vorladen des Timerregisters ein wenig verändert. Das Timerregister wird bei jedem Takt mit einem weiteren Register verglichen. Sind beide Register identisch, so wird der Interrupt für den Timer ausgelöst und das Timerregister wieder zurückgesetzt. Dies bringt die Möglichkeit, dass man direkt den gewünschten Wert für den Timer einstellen kann. Als Wert für dieses Vergleichsregister wurde 19999, bei einem Prescaler von $\phi = 8$ eingestellt. Zusammen mit der Frequenz von $32MHz$ des Controllers ergibt dies eine Zeit von $t = \frac{8 \cdot 20000}{32.000.000Hz} = 5ms$ und eine Frequenz von $200Hz$.

Während der Tests wurden zusätzlich die Steuerungsaufgaben auf dem Mikrocontroller ausgeführt. Deswegen konnte die Timerfunktion nicht mit einer so hohen Frequenz aufgerufen werden. Das Vergleichsregister wurde auf 39999 gesetzt. Dies entspricht einer Zeit von $t = 10ms$ und einer Frequenz von $100Hz$ für den Timer.

6.1.2 Schnittstellen

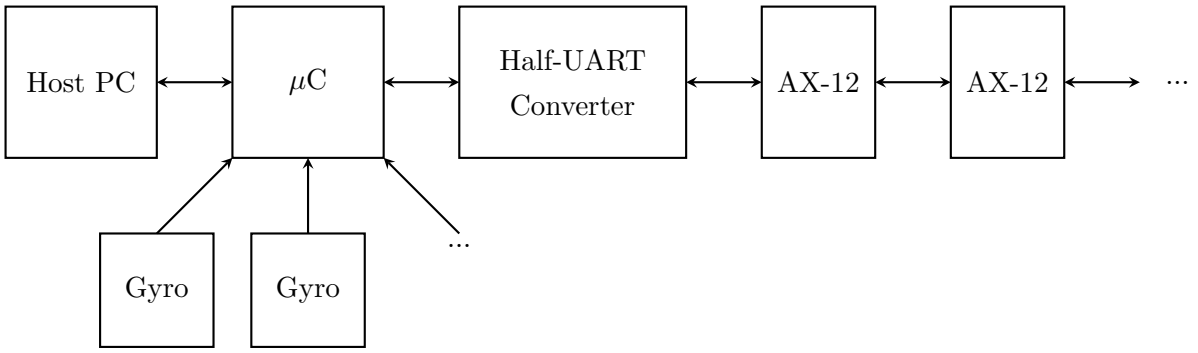

Abb. 6.2: Schnittstellen der einzelnen Hardwarekomponenten

Das Testsystem besteht aus mehreren in sich abgeschlossenen Hardwarekomponenten, wie zum Beispiel einem Host-PC, der gleichzeitig die Benutzerschnittstelle bildet, einem Mikrocontroller, auf dem die eigentliche Sensordatenverarbeitung und -regelung läuft, sowie den einzelnen Aktoren und Sensoren.

Der Host-PC ist mit dem Mikrocontroller über eine serielle UART-Schnittstelle verbunden, so dass über ein normales Terminalprogramm alle nötigen Daten in Form von ASCII-Zeichen übertragen werden. Die Kommandos und Sollwerte werden vom PC an die Hauptschleife des Programms auf dem Mikrocontroller übertragen, während alle Messergebnisse und auch ein schlichtes Benutzermenü als Zeichenkette vom Mikrocontroller an den PC geschickt und dort direkt angezeigt werden können.

Name	Einstellung
Baudrate	38400
Datenbits	8
Parität	Keine
Stopbits	1
Flusssteuerung	Keine

Tab. 6.2: Anschlusseinstellungen Mikrocontroller-Host

Die verwendeten Dynamixel-Aktuatoren müssen mit einer Half-Duplex-UART-Schnittstelle angesprochen werden, während auf dem Mikrocontroller nur eine weitere Full-Duplex-Schnittstelle vorhanden ist. Um die Aktuatoren dennoch verwenden zu können, schlägt die Firma Dynamixel eine Schaltung für einen Full-Duplex to Half-Duplex-UART Converter vor [36]. Dieser Converter regelt mittels eines digitalen Ausgangs des Mikrocontrollers die Kommunikationsrichtung der Schnittstelle. Im Programm des Mikrocontrollers wird dieser Ausgang jedesmal gesetzt, wenn eine Nachricht an die Aktuatoren geschickt wird. Ansonsten ist die Schnittstelle zum Mikrocontroller im Empfangsmodus, und es können Daten empfangen werden. Untereinander sind die Aktuatoren selbst mit einem Bussystem verbunden, so dass direkt alle Servos die Nachrichten vom Mikrocontroller erhalten. Mittels einer eindeutigen ID im gesendeten Kommando können auch direkt einzelne Servos angesprochen werden.

Name	Einstellung
Baudrate	1000000
Datenbits	8
Parität	Keine
Stopbits	1
Flusssteuerung	Keine

Tab. 6.3: Anschlusseinstellungen Mikrocontroller-Servos

Alle Gyroskope sind jeweils über einen separaten Eingangsport mit dem Mikrocontroller verbunden, der die Signale mittels Analog-Digitalwandlung verarbeiten kann. Die Gyroskope erhalten keine Daten. Sie schicken ständig ein analoges Signal an den Mikrocontroller, der diesen Analogwert dann für die

weitere Verarbeitung in einen digitalen Wert konvertieren muss. Hierzu wird der Eingang im Programm in festen Intervallen abgefragt.

6.1.3 Kommunikation zu Host-System

Wie bereits erwähnt, besitzen die verwendeten Mikrocontroller ausser den binären Digitaleingängen und den A/D-Wandlern keine direkte Möglichkeit, um vom Benutzer Daten einzugeben. Die Dateneingabe beziehungsweise die Steuerung des Programms muss somit von einem externen Host-System geschehen. Eine serielle Schnittstelle des Controllers wird für diese Aufgabe verwendet.

Damit auf dem Hostsystem keine spezielle Software installiert werden muss und die Steuerung des Stabilisierungssystems über ein einfaches Terminalprogramm geschehen kann, werden alle Benutzeranzeigen in Form eines Menüs und Statusmeldungen vom System selbst über die serielle Schnittstelle an das Host-System gesendet. Dort werden die empfangenen Daten angezeigt, und der Benutzer kann aus dem angezeigten Menü eine Auswahl treffen.

Der Benutzer kann nun über das Terminalprogramm durch Drücken einer Taste die gewünschte Aktion auf dem Mikrocontroller starten. Auf dem Mikrocontroller wird ständig die serielle Schnittstelle zum Hostsystem abgerufen. Sollten dort Daten eingegangen sein und das empfangene Zeichen einem internen Kommando entsprechen, so wird auf dem Controller ein Unterprogramm gestartet und eine Rückmeldung an das Host-System gesendet.

6.2 Stabilisierungsprogramm

Nachdem das komplette Hardwaresystem fertiggestellt war, konnte mit der Entwicklung des Stabilisierungsprogramms begonnen werden. Begonnen wurde die Arbeit damit, dass Kommunikations- und Abfrageroutinen für die einzelnen Komponenten entwickelt wurden. Diese Routinen wurden für das spätere Stabilisierungsprgramm benötigt. Damit konnte vorab schon getestet werden, ob das Zusammenspiel der einzelnen Komponenten gewährleistet war. Danach wurden die einzelnen, benötigten Funktionen des Mikrocontrollers, wie Interrupts und Timer implementiert. Nachdem alle benötigten Funktionalitäten in einzelnen Testprogrammen programmiert und getestet waren, wurde mit der Entwicklung des Stabilisierungsprgramm begonnen.

6.2.1 Programmablauf

Das Programm gestaltet sich in zwei verschiedene Schleifen. In der Hauptschleife wird die Benutzerkommunikation zum Host-System verarbeitet, und es werden die nötigen Programmzustände gesetzt. Die serielle Schnittstelle wird ständig abgefragt und danach werden die Benutzerkommandos, wie die Initialisierung oder das Starten der Stabilisierung über den Programmzustand, verarbeitet.

Als zweite Schleife wird eine Timerfunktion verwendet, die in vorher definierten, festen Intervallen aufgerufen wird. In dieser Funktion wird die Sensorabfrage und Stabilisierung durchgeführt. Es gibt auch hier wieder zwei verschiedene Zweige, die aufgerufen werden können. Im ursprünglichen Programmzustand wird in der Timerfunktion keine Aktion ausgeführt.

Sobald der Benutzer das Kommando zur Initialisierung der Gyroskope gesendet hat, wird in der Hauptschleife der Programmzustand für die Initialisierung gesetzt und die dafür nötigen Variablen werden gelöscht. Jedesmal wenn nun die Timerfunktion startet, wird ein Unterprogramm zur Bearbeitung der Initialisierung aufgerufen, bis die Initialisierung abgeschlossen ist. Danach wird wieder automatisch in einen Wartezustand umgeschaltet, bis der Benutzer eine neue Initialisierung oder die Stabilisierung des Systems startet.

Nachdem die Initialisierung der Sensoren abgeschlossen ist, kann der Benutzer in der Hauptschleife das Kommando zur automatischen Stabilisierung geben. Durch das Kommando wird in den Programmzustand Stabilisierung umgeschaltet. Bei jedem Aufruf der Timerfunktion wird nun ein Unterprogramm zur Stabilisierung und Lageregelung des Systems gestartet. Dieser Programmzustand bleibt erhalten, bis wieder ein Initialisierungskommando vom Benutzer geschickt, die Stabilisierung manuell vom Benutzer beendet oder eine Sollposition für die Servos vorgegeben wird.

6.2.2 Initialisierung

In den Datenblättern von Gyroskopen wird vom Hersteller eine Spannung angegeben, die anliegen sollte, wenn das Gyroskop um die Messachse keiner Drehung unterliegt. Dies ist aber leider nur ein ungefährer Richtwert, denn Fertigungstoleranzen, sowie Spannungs- und Temperaturschwankungen bedingen, dass sich diese Spannung minimal in eine Richtung verschieben kann. Da es bei diesen Sensoren zu einem Drift kommen kann, muss eine genaue Initialisierung dieses Spannungswerts erfolgen, um diesen Drift zu minimieren. Die Initialisierung muss für jedes Gyroskop einzeln und bei jedem Start des Systems erfolgen. Das Signalrauschen ist nicht ausschließlich von den Gyroskopen abhängig. Auch durch die Signalleitungen, Lötpunkte und A/D-Wandler kann ein Signalrauschen entstehen oder verstärkt werden.

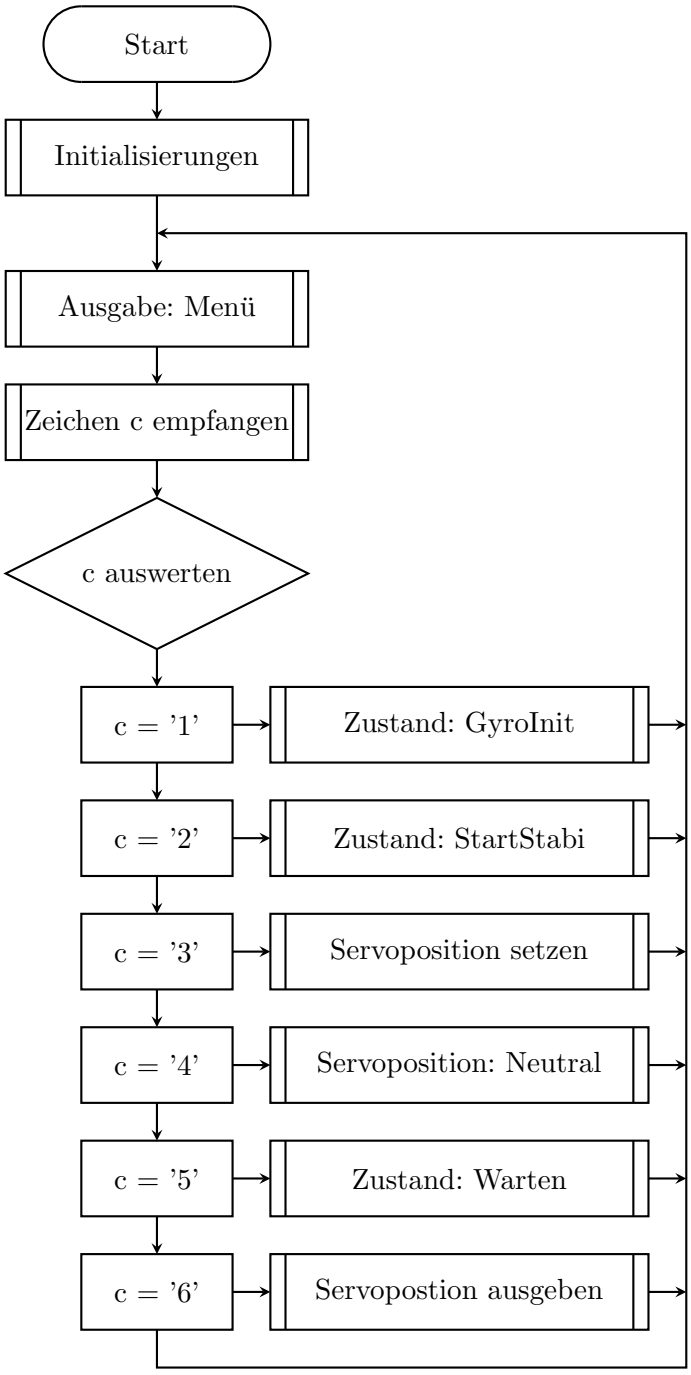

Abb. 6.3: Programmablaufplan - Hauptschleife

Die A/D-Wandler in den verwendeten Mikrocontrollern (ATMega128 und SH7125) besitzen eine Auflösung von 10 Bit. Der eingehende Analogwert wird also auf 1024 äquidistante Werte quantisiert. Die in Entwicklung und Test benutzten Gyroskope liefern über den Ausgang einen maximalen Bereich der Drehwinkelgeschwindigkeit von $\pm 300°/s$. Die A/D-Wandler haben also im Hinblick auf die Drehrate eine maximale Auflösung von ca. $0,586°/s$. Wenn bei der Initialisierung nur ein um eine Stelle falscher Wert entsteht, so würde dies zu einem Drift von $35°/min$ führen. Unter solchen Voraussetzungen wäre keinerlei sinnvolle Stabilisierung möglich.

Die Initialisierung eines Gyroskops wird durchgeführt, indem das Gyroskop in absoluter Ruhestellung gehalten wird und gleichzeitig über einen bestimmten Zeitraum die Sensordaten abgerufen, in ein Digitalsignal umgewandelt und im Anschluss gemittelt werden. Ein einmaliges Abrufen der Sensoren ist hierbei nicht ausreichend, da die Signalleitungen und somit auch die Sensordaten am Analogeingang des Controllers mit einem Rauschen behaftet sind.

Beim Abrufen der A/D-Wandler wird im bestehenden Code für die Mikrocontroller eine Mittelung über mehrere Sensorwerte durchgeführt. Hierbei wird direkt hintereinander mehrmals der Analogeingang des Controllers abgerufen und danach eine Mittelung der Werte durchgeführt. Dies ist die erste Stufe der Rauschunterdrückung. Sie entspricht einem Tiefpassfilter, bei dem ein Teil des Signalrauschens herausgefiltert werden soll. Die Anzahl der zu mittelnden Werte kann im Programm mit Hilfe einer Präprozessoranweisung geändert werden. Es hat sich gezeigt, dass es ausreichend ist, wenn jeweils zwei Werte gemittelt werden.

In einer zweiten Stufe der Rauschminderung wird über mehrere Timeraufrufe der ausgelesene Sensorwert der Gyroskope gemittelt. Hierbei werden die Sensorsignale bei mehreren aufeinander folgenden Timeraufrufen aufsummiert. Dadurch werden hohe Frequenzen entfernt. Auch hier kann die Anzahl der zu summierenden Werte vorgegeben werden.

Probleme mit dem ATMega128

Ein Problem, das während der Entwicklung mit dem ATMega128 aufgetreten ist, war die Tatsache, dass die A/D-Wandler nach dem Start des Mikrocontrollers sehr unterschiedliche Werte ausgelesen haben. Wenn jeweils die erste Messung verwendet worden wäre, dann hätte dies zum Ergebnis gehabt, dass ein um mehrere $°/s$ verschobener Nullpunkt initialisiert worden wäre und die Servos innerhalb von Sekunden an die Grenzen ihres Arbeitsbereichs gefahren wären.

Eine erste Vermutung war, dass es sich bei dem verschobenen Nullpunkt um eine Verschiebung des Nullpunkts aufgrund von Temperaturschwankungen im Gyroskop handelt. Zum einen gibt der Hersteller keinerlei Auskunft über den Einfluss von unterschiedlichen Temperaturen auf den ausgegebenen Sensorwert, zum anderen gibt er jedoch an, dass der Sensorwert relativ stabil gegenüber Temperatur-

Abb. 6.4: Programmablaufplan - Initialisierung

schwankungen wäre. Die Gyroskope bieten die Möglichkeit, zu der Drehwinkelgeschwindigkeit auch die jeweils anliegende Temperatur zu messen. Nachdem zusätzlich zu den Sensorsignalen auch jeweils der zugehörige Temperaturwert ausgelesen wurde, musste diese Vermutung wieder verworfen werden, da die Temperatur der Gyroskope über den sich ändernden Sensorwert sehr stabil war.

Wenn nach einer Laufzeit von wenigen Minuten die Initialisierung durchgeführt wurde, dann konnte die Nullstellung der Gyroskope relativ gut initialisiert und mit diesen Werten konnte eine Stabilisierung durchgeführt werden. Dieses Vorgehen erforderte jedoch, dass der Benutzer nach einer nicht genauer spezifizierten Zeit die Initialisierung startete. Gegebenenfalls musste eine neue Initialisierung durchgeführt werden, wenn bei der ersten nicht die richtige Neutralstellung gefunden wurde.

Es wäre eher wünschenswert, dass innerhalb des Programms die Möglichkeit bestünde, dass in jedem Fall die richtige Neutralstellung gefunden wird und der Benutzer diese nicht kontrollieren muss. Deswegen wurde ein Automatismus entwickelt, der den Initialisierungswert automatisch kontrolliert. Hierbei wird vom Programm nicht nur eine Initialisierung durchgeführt, sondern mehrere Initialisierungen hintereinander. Im Programm kann definiert werden, wieviele Initialisierungen jeweils maximal gleiche Werte liefern müssen. Anhand dieses definierten Wertes werden die letzten Initialisierungen gespeichert und miteinander verglichen. Es wird jeweils der minimale und maximale Initialisierungswert bestimmt. Anhand eines weiteren vordefinierten Schwellwertes werden diese beiden Werte für jedes Gyroskop miteinander verglichen. Wenn für alle Gyroskope die Werte nahe genug beieinander liegen, so gilt die Initialisierung als abgeschlossen. Zusätzlich wird für die endgültige Initialisierung das Ergebnis dieser letzten Vorinitialisierungen gemittelt.

Dieser Algorithmus bringt eine längere Initialisierungszeit mit sich. Allerdings muss der Benutzer hierbei nicht in die Initialisierung eingreifen, und es wird auch keine Erfahrung für die Initialisierung benötigt. Die Initialisierung kann also komplett automatisch geschehen. Nachdem vom ATMega128 auf den SH7125 Mikrocontroller umgestellt wurde, sind diese Probleme nur noch minimal aufgetreten. Bereits direkt die zweite Initialisierung in Folge lieferte eine gute Näherung des Nullpunkts. Die Initialisierung wurde jedoch im Programm bestehen gelassen. Es wurde nur die Anzahl der nötigen übereinstimmenden Vorinitialisierungen herabgesetzt, um die komplette Initialisierungszeit wieder zu minimieren.

6.2.3 Stabilisierungsschleife

Nachdem die Initialisierung der Gyroskope durchgeführt wurde, bei der das ganze System sich in einer Ruhelage befinden sollte, kann die eigentliche Stabilisierung durchgeführt werden. Hierbei wird der Endeffektor der kinematischen Kette auf Grund der gemessenen Drehraten in den Gyroskopen bestmöglich stabilisiert. Die Theorie zur Stabilisierung ist ausführlich in Kapitel 5.2.2 beschrieben.

Deswegen wird hier nur kurz auf den Ablauf der einzelnen Schritte der Stabilisierungsschleife eingegangen.

Zuerst müssen die A/D-Wandler mit den Sensorwerten der Gyroskope ausgelesen werden. Hierbei findet, wie bei der Initialisierung auch, ein zweistufiger Tiefpassfilter seine Anwendung. In einer ersten Stufe werden direkt beim Abrufen des A/D-Wandlers direkt mehrere Werte gemittelt. Hierbei kommt die gleiche Einstellung wie bei der Initialisierung zum Einsatz. In der zweiten Stufe werden über mehrere Zeitschritte jeweils die Sensorwerte aufsummiert und dann der Mittelwert gebildet. Die Anzahl der zu mittelnden Werte ist hier jedoch unabhängig von der Anzahl wie sie bei der Initialisierung verwendet wird. Es steht eine eigene, variable Präprozessoranweisung zur Verfügung.

Sobald die Timerfunktion oft genug aufgerufen und eine Mittelung beziehungsweise Tiefpassfilterung der Sensorwerte durchgeführt wurde, beginnt der eigentliche Teil der Stabilisierungsberechnungen. Zunächst wird aus den gemessenen Drehraten der Gyroskope, wie in den Kapiteln 5.2.3 und 5.2.4 beschrieben, die bestmögliche Drehrate im Basiskoordinatensystem ermittelt. Direkt im Anschluss wird der untere Teil der Jacobi-Matrix bestimmt, indem die Lage der Drehachsen der Aktoren, wie in Kapitel 5.2.5 beschrieben, berechnet wird.

Nachdem die Drehrate im Basiskoordinatensystem und die Lage der Drehachsen der Aktoren berechnet worden sind, wird das inverse Jacobi-Modell gelöst. Das Ergebnis ist die nötige Sollgeschwindigkeit in den Gelenken. Die verwendeten Servos bieten zum einen die Möglichkeit, eine neue Sollposition anzugeben. Gleichzeitig kann aber auch eine Sollgeschwindigkeit übermittelt werden, mit der die Servos in die neue Sollposition fahren. Die neue Sollposition der Servos ergibt sich aus dem Produkt der neuen Sollgeschwindigkeit und dem Zeitintervall der Timerfunktion.

6.2.4 Datenstrukturen

Im Programm des Mikrocontrollers werden, wie in den Grundlagenkapiteln schon beschrieben, hauptsächlich Matrizen und Vektoren für die nötigen Stabilitätsberechnungen verwendet. Programmintern werden sowohl die Matrizen als auch die Vektoren in Variablenarrays abgelegt. Im Falle der Vektoren stehen jeweils die x-, y- und z-Komponenten jeweils in drei aufeinander folgenden Arrayeinträgen.

Matrizen werden in den Arrays gespeichert, indem jeweils die einzelnen Komponenten der Zeilen von links nach rechts nachfolgend im Array gespeichert werden und die Zeilen von oben nach unten auch jeweils nachfolgend. Um auf einen Eintrag in der i-ten Zeile und der j-ten Spalte zuzugreifen, wird also die Anweisung $pos = i \cdot 3 + j$ verwendet.

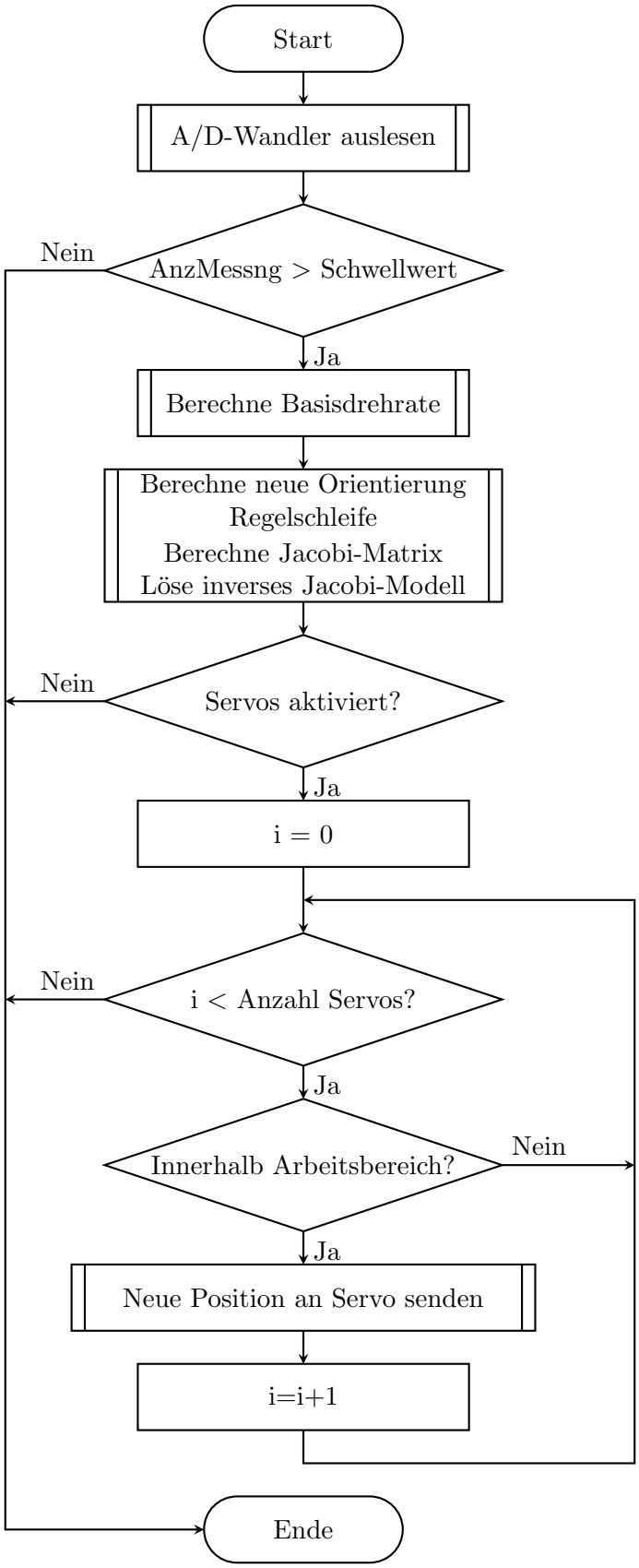

Abb. 6.5: Programmablaufplan - Stabilisierung

Wenn mehrere Matrizen ($n = 9$) oder Vektoren ($n = 3$) in einem Array verwendet werden sollen, so wird eine weitere Indizierung nötig. Wenn das Array m Strukturen beinhalten soll, so muss im Speicher Platz für $n \cdot m$ Einträge reserviert werden.

Um auf die i-te Komponente im k-ten Vektor zuzugreifen, wird die Anweisung $pos = k \cdot 3 + i$ benutzt. Möchte man in der k-ten Matrix auf die i-te Zeile und die j-te Spalte zuzugreifen, wird die Anweisung $pos = k \cdot 9 + i \cdot 3 + j$ verwendet.

6.3 Softwaremodularität

Beim Erstellen der Software wurde darauf Wert gelegt, dass die einzelnen Elemente modular aufgebaut sind. Da die Software in reinem (ANSI)-C programmiert wurde, sind diese Module nicht als einzelne Objekte implementiert sondern als Paar einer Header- und C-Datei. Für jede Hardwarekomponente wurde ein solches Dateipaar entwickelt. In jeder Header-Datei sind Funktionsprototypen deklariert, die in ihrer Signatur die jeweils nötigen Übergabeparameter beinhalten. In der C-Datei steht dann der hardwarenahe Code, der die Funktionalität gewährleistet. Alle hardwarespezifischen Paramater sind ebenfalls in der Header-Datei deklariert. Eine genaue Beschreibung der für jedes Modul nötigen Funktionen ist in Anhang E aufgelistet.

Sobald eine Hardwarekomponente ausgetauscht wird, muss für diese Komponente ein neues Modul in Form von einem Paar von Header- und C-Datei geschrieben werden. Hierbei ist darauf zu achten, dass in der Header-Datei alle nötigen Funktionsprototypen stehen und diese in der entsprechenden C-Datei auch beschrieben sind. Im Sourcecode der Haupt-C-Datei muss schließlich nur die eingebundene Header-Datei für die neue Hardwarekomponente geändert werden. Dies bedeutet eine minimale Änderung im eigentlichen Programmsourcecode, falls eine neue Komponente getestet werden soll oder aus anderen Gründen verwendet werden sollte.

Ein weiterer Vorteil der Modularität der Software liegt darin, dass man nicht auf eine bestimmte Ein- oder Ausgabe festgelegt ist. Momentan wurde die Software auf dem Microcontroller so programmiert, dass sowohl die Ein- als auch die Ausgabedaten über die serielle Schnittstelle von einem Hostcomputer geschickt werden. Es wäre aber genauso denkbar, dass statt des Mikrocontrollers ein Personal Digital Assistant (PDA) zum Einsatz kommt, bei dem die Ein- und Ausgabe direkt über einen Touchscreen geschieht.

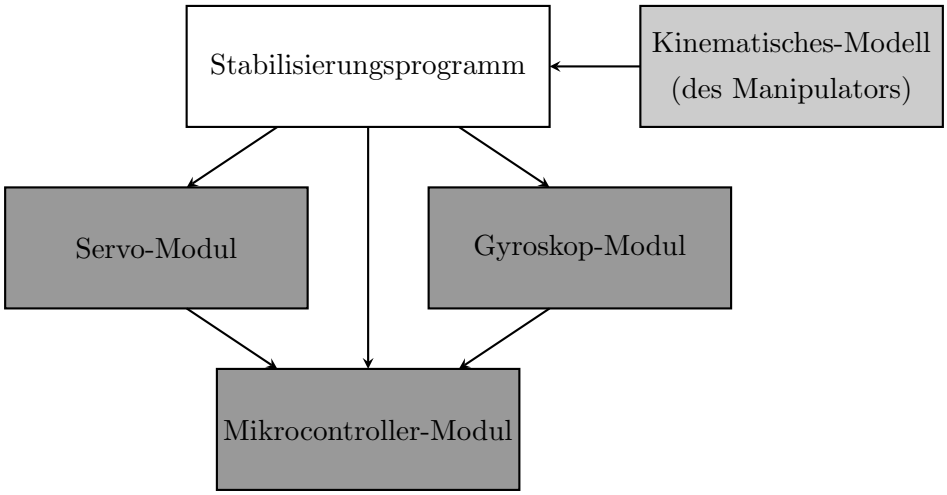

Abb. 6.6: Abhängigkeiten zwischen den Modulen

6.4 Zusammenspiel und Abhängigkeiten der einzelnen Module

Abbildung 6.6 zeigt die Abhängigkeiten zwischen den einzelnen Softwaremodulen. Zentral ist das Stabilisierungsprogramm, das durch das kinematische Modell die nötigen Informationen über den Aufbau des Manipulators und die Position der Sensoren erhält. Das Programm greift direkt auf Funktionen des Mikrocontrollers, wie zum Beispiel den Timer oder die Kommunikation zum Host-PC zu. Daneben werden Funktionen der Gyroskope, wie zum Beispiel das Auslesen und Initialisieren aufgerufen. Diese Funktionen aus dem Gyroskop-Modul setzen aber auch wieder Funktionen vom Mikrocontroller, wie das Auslesen der A/D-Wandler voraus. Die Ausgabe der Stabilisierungsfunktion sind die nötigen Gelenkwinkelstellungen und -geschwindigkeiten. Diese werden an das Servo-Modul in den entsprechenden Methoden übergeben. Von dort aus werden wiederum Funktionen im Mikrocontroller-Modul aufgerufen, um die Signale über eine Half-Duplex-UART-Schnittstelle an die Servos zu schicken.

Aus der Abbildung ist ersichtlich, dass jedes Modul wirklich nur die Einstellungen und den Quellcode enthält, der für eine Hardwarekomponente notwendig ist. Die Servo- und Gyroskopmodule enthalten zwar den nötigen Code, um Kommandos vom Stabilisierungsprogramm entgegenzunehmen. Sie dienen aber dazu, um diese Kommandos in geeignete Kommunikationsnachrichten zu übersetzen. Die Kommunikation zu den Servos und Gyroskopen geschieht dann schließlich über die spezifischen Kommunikationsfunktionen des Mikrocontrollers.

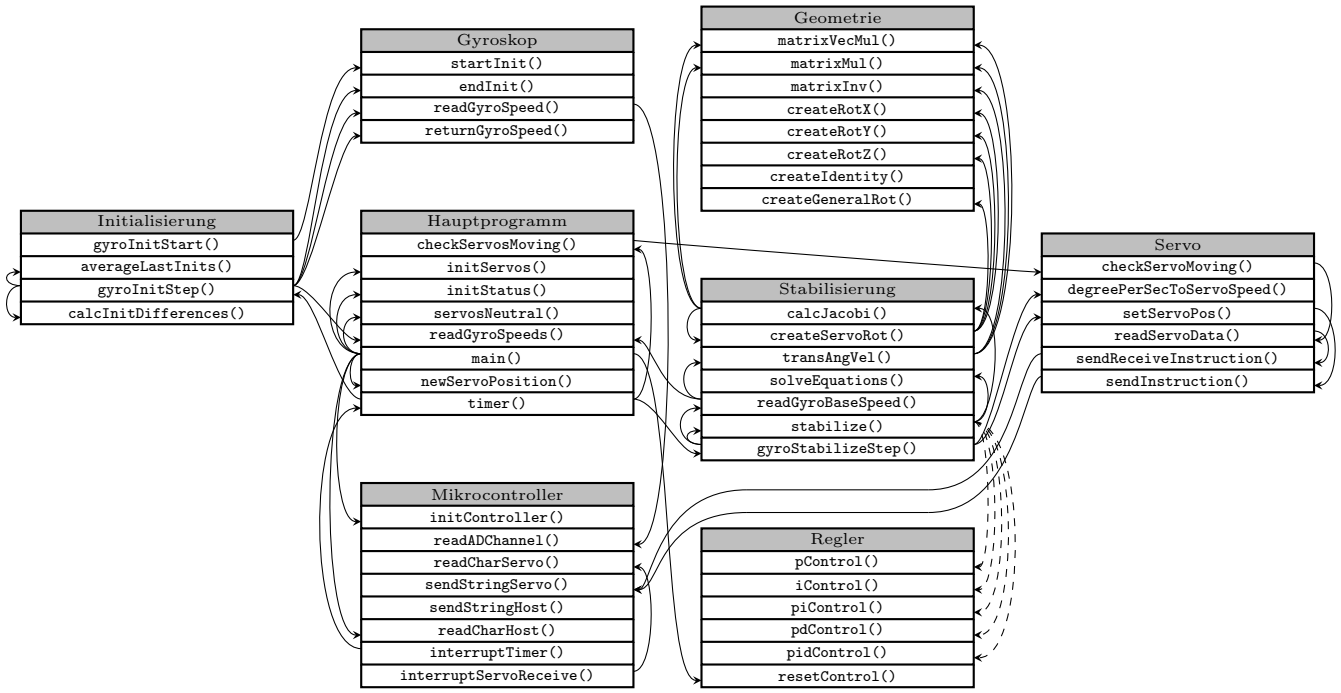

Abb. 6.7: Abhängigkeiten zwischen den Funktionen

6.5 Prozeduraler Aufbau des Stabilisierungsprogramms

In Schaubild 6.7 sind die Abhängigkeiten zwischen den einzelnen Funktionen und damit auch den einzelnen Modulen des Programms aufgeführt. Der Übersicht halber sind in diesem Diagramm kleine Hilfsfunktionen unberücksichtigt. Diese sind dann aber jeweils im zugehörigen Modul in der Quellcode-Datei enthalten. Wenn der jeweilige Header eingebunden wird, so stehen diese Hilfsfunktionen auch zur Verfügung.

Ein Pfeil von einer Funktion zu einer anderen bedeutet, dass diese Funktion die andere aufruft. Sie ist also von der Funktion abhängig, zu der der Pfeil zeigt. Um dies auf die Module auszuweiten, bedeutet es, dass ein Modul, bei dem eine Funktion eine Funktion aus einem anderen Modul aufruft, auch von diesem Modul abhängig ist. Im Quellcode ist dies erkennbar dadurch, dass die Header-Dateien in diesem Modul eingebunden wurden.

Die gestrichelten Pfeile von der Funktion `stabilize()` zu den aufgerufenen Reglerfunktionen sollen andeuten, dass hier wahlweise eine dieser Funktionen aufgerufen werden kann. Je nach Regelungstyp wird eine andere Funktion aufgerufen. In der vorliegenden Arbeit wurde ein Proportionalregler verwendet und entsprechend die Funktion `pControl()` aufgerufen.

6.6 Integration in ein bestehendes Gesamtsystem

Um die Stabilisierungsfunktion in ein bestehendes System zu integrieren, ist neben einigen Variableninitialisierungen und der Initialisierung der Gyroskope insbesondere der Aufruf der Funktion `stabilize()` nötig. Wenn in diesem System die Gyroskope auch anderweitig verwendet werden, so sollte diese Initialisierung bereits schon im System durchgeführt worden sein. Deswegen wird hier ausschließlich auf die Integration der Funktion `stabilize` und deren Abhängigkeiten eingegangen.

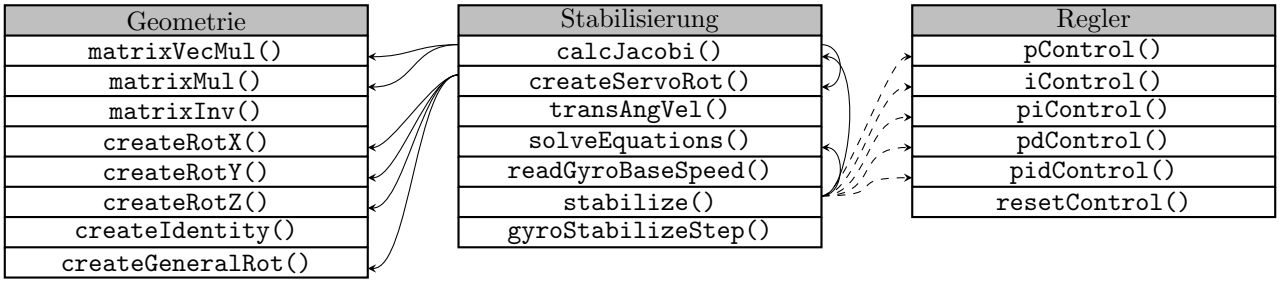

Abb. 6.8: Von der Funktion stabilize() benötigte Funktionen

In Abbildung 6.8 sind die nötigen Funktionen aufgeführt, um die Funktion `stabilize()` einzusetzen. Es handelt sich um einen Ausschnitt aus Abbildung 6.7, in dem alle nicht benötigten Module entfernt und die übrigen Module anders angeordnet wurden, um eine übersichtlichere Darstellung zu erhalten.

Die Funktion `stabilize()` selbst erhält als Parameter die Drehwinkelgeschwindigkeit im Basiskoordinatensystem als Pointer auf ein float-Array mit drei Einträgen. Als Ausgabe liefert sie dann die nötigen Geschwindigkeiten in den Gelenken, um eine bestmögliche Stabilisierung zu gewährleisten. Die Rückgabe geschieht auch wieder durch ein Array, das mit der Größe, wie die Anzahl der Aktuatoren, initialisiert wurde. Es wird ein Pointer auf dieses Array übergeben, und die Funktion trägt die nötigen Drehgeschwindigkeitswerte an die jeweiligen Positionen ein.

Wie aus Abbildung 6.8 ersichtlich, sind für die Funktion `stabilize()` fast alle Funktionen der Module *Regler* und *Geometrie* sowie einige Funktionen des Moduls *Stabilisierung* notwendig. Zusätzlich sollte vor Beginn der Stabilisierung aus dem Modul *Regler* die Funktion `resetControl()` aufgerufen worden sein.

Neben den Funktionen sind einige globale Variablen nötig, die vorher initialisiert werden oder bestimmte Werte enthalten müssen. Es handelt sich hierbei um die Variablen:

- `g_fpServoAxis[NUMBER_OF_SERVOS` $\times$ `3]`
  Orientierung der Drehachsen der einzelnen Servos. Im Regelfall sollten diese Achsen jeweils einem Einheitsvektor und damit einer der elementaren Achsen entsprechen, und im Falle von

Denavid-Hartenberg mit der z-Achse zusammenfallen. Es ist jedoch möglich, hier jede beliebige Drehachse vorzugeben.

- `g_fpServoPosition[NUMBER_OF_SERVOS]`
  Momentane Stellung aller Servos in Grad.
- `g_fpRotationFix[NUMBER_OF_SERVOS` $\times$ `9]`
  Feste Rotationsmatrizen zwischen den Gelenken. Jede Rotationsmatrix besteht aus neun Einträgen.

Der nötige Quellcode für die Initialisierung der Variablen `g_fpServoAxis[NUMBER_OF_SERVOS` $\times$ `3]` und `g_fpRotationFix[NUMBER_OF_SERVOS` $\times$ `9]` kann durch das Konfigurationswerkzeug automatisch generiert werden. Die Position der Servos im Array `g_fpServoPosition[NUMBER_OF_SERVOS]` muss während der Laufzeit des Programms berechnet werden.

6.7 Konfigurationswerkzeug

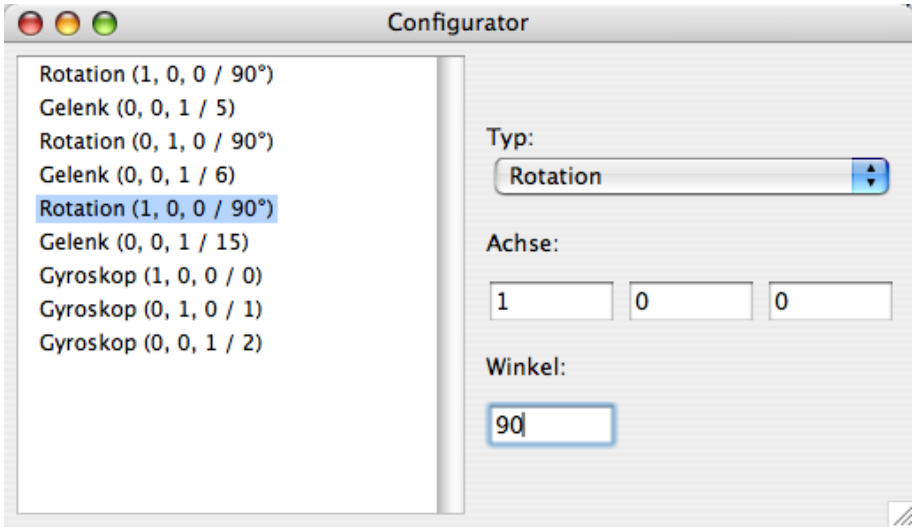

Abb. 6.9: Konfigurationswerkzeug zur Anpassung des kinematischen Modells

Damit vom Benutzer ohne große Vorkenntnis das kinematische Modell des Roboters verändert und gleichzeitig ein minimales Programm auf den Mikrocontroller geladen werden kann, wurde dafür ein Konfigurationswerkzeug entwickelt. Damit sollte es einem Benutzer, der die Grundlagen der Robotik beherrscht, möglich sein, das Stabilisierungsprogramm in soweit zu verändern, dass zum einen die kinematische Kette und zum anderen aber auch die Position und Lage der Sensoren verändert werden kann. Hierzu muss der Benutzer nicht in den Programmcode des Stabilisierungsprogramms eingreifen, sondern kann in einem Editor recht intuitiv die Einstellungen vornehmen. Nach der durchgeführten Anpassung kann er die Änderungen speichern oder direkt den nötigen Programmcode des Stabilisierungsprogramms für den Mikrocontroller automatisch generieren lassen. Dieser Code muss vor dem

Hochladen auf den Mikrocontroller lediglich kompiliert werden, da er in Form von C-Dateien generiert wird. Eine Installationsanleitung für das Konfigurationswerkzeug unter Windows findet sich in Anhang A.

6.7.1 Bedienung

Das Konfigurationswerkzeug besteht im Wesentlichen aus zwei Hauptteilen. Im linken Teil des Programmfensters ist eine Liste von verschiedenen Gliedertypen aufgeführt. Im Grunde entspricht diese Liste der kinematischen Kette des zu stabilisierenden Robotersystems. Zusätzlich zu den festen Transformationen zwischen den Gelenken und den Gelenken selbst sind hier jedoch auch die Sensoren aufgeführt. Beim Eingeben der Greifarmkonfiguration ist darauf zu achten, dass die Gyroskope in dieser Liste auch an der Position des richtigen Gliedes liegen.

Auf der rechten Hälfte des Programmfensters kann das jeweils markierte Listenelement auf der linken Seite verändert werden. Es kann hier der Elementtyp ausgewählt werden, der danach die möglichen Eingabefelder für den jeweiligen Typ freigibt. Das sind zum einen die Rotations- beziehungsweise die Drehachse, sowie die elementabhängigen Parameter, wie sie in der folgenden Sektion beschrieben werden.

6.7.2 Elementtypen

Als Dateneingabe stehen drei verschiedene Elementtypen zur Verfügung. Im Einzelnen sind dies:

- Rotation
  Die Rotation ist eine feste Transformation vom vorherigen zum nächsten Listenelement. Die Rotation ist durch eine Drehachse sowie einen festen Drehwinkel komplett beschrieben. Bei der Codeerzeugung wird überprüft, ob die Drehachse normiert ist, also die Länge 1 besitzt. Es wird während der Codegenerierung eine Warnmeldung ausgegeben und die Drehachse für den erstellten Code automatisch normiert und eine entsprechende allgemeine Rotationsmatrix (siehe 3.1.5) erstellt.
- Gelenk
  Beim Gelenk handelt es sich um ein Gelenk der kinematischen Kette, das im Manipulator durch einen Aktuator an dieser Stelle realisiert ist. Beschrieben wird ein solches Gelenk durch eine eindeutige ID, sowie eine Drehachse, um bei der laufenden Stabilisierung die Transformation, die durch das Gelenk entsteht, berechnen zu können. Zusätzlich bietet das Konfigurationstool sowie das Stabilisierungsprogramm die Möglichkeit, den Arbeitsbereich des Aktuators in Form einer minimalen und maximalen Auslenkung anzugeben. So kann vermieden werden, dass es zu mechanischen Anschlägen des Aktuators kommt und der Roboterarm beschädigt wird.

- Gyroskop
  Das Gyroskop als Sensor ist das einzige Element, das nicht zum Aufbau einer kinematischen Kette gehört. Dennoch können die Gyroskope als Elemente in der Liste beliebig positioniert werden, um eine maximale Flexibilität zu erreichen. Im Configurator werden die Gyroskope durch einen A/D-Kanal, sowie eine Drehachse beschrieben. Die Drehachse wird benötigt, um eine beliebige Achse vorzugeben, um die die Sensormessung erfolgt. Bei der Datenmessung wird dieser Vektor dann mit der gemessenen Drehwinkelgeschwindigkeit multipliziert.

6.7.3 Dateiformat

Um Dateien zu speichern bietet sich die Extensible Markup Language (XML) als Datenformat an. XML bietet die Möglichkeit einer strukturierten Szenebeschreibung in Form einer Baumstruktur. Als Szenenknoten gibt es den Knoten `inertial_stabilizer`. Darunter liegen alle Knoten auf einer Ebene. Die Knoten können hierbei folgende Typen, mit den jeweiligen Attributen sein:

- `rotation`
  - `x, y, z`: x-, y- und z-Komponente der Rotationsachse
  - `angle`: Drehwinkel
- `link`
  - `x, y, z`: x-, y- und z-Komponente der Drehachse
  - `id`: eindeutige ID des Aktuators
  - `min, max`: minimaler und maximaler Drehwinkel des Aktuators
- `gyroscope`
  - `x, y, z`: x-, y- und z-Komponente der Drehachse
  - `channel`: Kanal des A/D-Wandlers

6.7.4 Codegenerierung

Das Ergebnis des Konfigurationstools ist ein automatisch generierter Code nach den Einstellungen, die der Benutzer im Configurator eingegeben hat. Zum einen handelt es sich dabei um die Deklaration der entsprechenden Variablen und der Definition von statischen Präprozessoranweisungen. Auf der anderen Seite müssen aber auch die entsprechenden Variablen in dafür vorgesehenen Funktionen nach den Benutzervorgaben initialisiert werden.

Einen weiteren Geschwindigkeitsvorteil, den man beim Aufstellen der Rotationsmatrizen während der Laufzeit nutzen kann, ist die Tatsache, dass eine elementare Rotationsmatrix, wie sie in Kapitel 3.1.2

vorgestellt wird, wesentlich günstiger zu berechnen ist, als die allgemeine Rotationsmatrix aus Kapitel 3.1.5. Hierzu wird vom Konfigurationswerkzeug eine Funktion aufgestellt, die einen Drehwinkel und die Nummer des Gelenks übergeben bekommt, für das die Rotationsmatrix aufgestellt werden soll. Als Rückgabe gibt die Funktion dann die Rotationsmatrix aus. Beim Aufstellen der Matrix wird aber darauf geachtet, ob eine elementare Rotationsmatrix aufgestellt werden kann oder die allgemeine Rotationsmatrix aufgestellt werden muss. Entschieden wird dies über die Achse, die der Benutzer für das jeweilige Gelenk eingegeben hat. Handelt es sich um eine Achse der Form $(1,0,0)^T$, $(0,1,0)^T$ oder $(0,0,1)^T$, so kann eine elementare Rotationsmatrix erstellt und damit Rechenzeit gespart werden.

Neben dem automatisch generierten Code ist jedoch zusätzlich in der Header, sowie der C-Datei auch statischer Code, wie zum Beispiel die Hauptfunktion, weitere Funktionen, Variablendeklarationen oder Präprozessoranweisungen nötig. Dieser Teil ist nicht fest im Configurator vorkompiliert, sondern befindet sich in vier verschiedenen Dateien (top.h, foot.h, top.c und foot.c) im gleichen Verzeichnis wie das Konfigurationstool. Wenn der Code vom Programm erstellt wird, so geschieht dies in der folgenden Reihenfolge:

- top.h
- Präprozessoranweisungen
- foot.h

beziehungsweise

- top.c
- Globale Variablendeklarationen
- Funktionsdeklarationen
- foot.c

Präprozessoranweisungen

- `NUMBER_OF_GYROS`: Anzahl der definierten Gyroskope
- `NUMBER_OF_SERVOS`: Anzahl der definierten Aktuatoren

Variablendeklarationen

- `float g_fpRotationFix[NUMBER_OF_SERVOS × 9]`
- `float g_fpServoAxis[NUMBER_OF_SERVOS × 3]`
- `float g_fpServoMin[NUMBER_OF_SERVOS]`

- `float g_fpServoMax[NUMBER_OF_SERVOS]`
- `unsigned char g_ucpServoID[NUMBER_OF_SERVOS]`
- `float g_fpGyroAxis[NUMBER_OF_GYROS × 3]`
- `unsigned char g_ucpGyroChannel[NUMBER_OF_GYROS]`
- `unsigned char g_ucpGyroLinksBefore[NUMBER_OF_GYROS]`

Funktionsdeklarationen

Bevor die Funktionsdeklarationen in den generierten Quellcode eingefügt werden, müssen die einzelnen, vom Benutzer eingegebenen Drehachsen normiert werden. Sollte eine Drehachse nach dem euklidischen Maß eine größere Länge als 1 aufweisen, so wird dies zusätzlich dem Benutzer in einer Warnmeldung mitgeteilt.

- `void InitRotations()`
  In der Funktion `InitRotations()` werden die festen Rotationsmatrizen initialisiert. In Kapitel 5.2.3 ist der Aufbau der einzelnen Transformationen als eine Abfolge von statischen und online berechneten Rotationsmatrizen beschrieben. Nach dieser Beschreibung werden die einzelnen Rotationsmatrizen zusammengefasst. Wenn also der Benutzer nacheinander mehrere verschiedene statische Rotationen in die Liste eingetragen hat, so wird hier aus allen aufeinanderfolgenden Rotationen eine Gesamtrotationsmatrix berechnet, so dass die Abfolge aus Kapitel 5.2.3 gewährleistet ist. Das spart zur Laufzeit des Programms nochmals Rechenzeit.
- `void InitGyros()`
  In dieser Funktion werden die Drehachsen sowie die zugehörigen Kanäle der Gyroskope nach den Vorgaben der internen Datenstrukturen initialisiert. Sollten direkt vor den jeweiligen Gyroskopen feste Rotationsmatrizen eingegeben worden sein, so wird die eingegebene Drehachse vorher mit den festen Rotationsmatrizen multipliziert, um während der Laufzeit des Stabilisierungsprogramms Rechenzeit zu sparen.
- `void InitServos()`
  Innerhalb der Funktion `InitServos()` werden die Drehachsen, die minimalen sowie maximalen Drehwinkel und die entsprechende IDs der Servos initialisiert.
- `void genRotationmatrix(float angle, int link, float* rotMat)`
  Um die Tatsache auszunutzen, dass bei Aufstellung von elementaren Rotationsmatrizen Rechenzeit gespart werden kann, sollte bei der Generierung der Rotationsmatrizen für die Gelenke während der Laufzeit diese Funktion aufgerufen werden. Es wird dann automatisch jeweils die günstigste Alternative gewählt.

6.7.5 Kommandozeile

Neben dem Start des Konfigurationswerkzeugs mit grafischer Benutzeroberfläche ist es möglich, das Programm auch ohne solche von der Kommandozeile aus zu starten. Dies kann zum Beispiel nötig sein, wenn man die automatische Codegenerierung direkt aus einem Makefile heraus starten möchte.

Beim Start des Werkzeugs ohne Benutzeroberfläche wird eine XML-Datei und die darin enthaltenen Strukturen eingelesen und direkt der zugehörige Programmcode erstellt. Als Eingabe sind dann der Dateiname der zu öffnenden Datei und zusätzlich der Dateiname der zu erstellenden Code-Datei nötig. Bei der Erstellung wird eine möglicherweise bereits bestehende Code-Datei überschrieben. Es sollte zunächst überprüft werden, ob bereits eine Datei mit dem Namen existiert.

Der Aufruf des Programms ohne Benutzeroberfläche geschieht durch den Parameter `-f`, gefolgt von den beiden Dateinamen. Der Programmaufruf geschieht dann über das Kommando:

```
java Configurator -f %inputfilename %outputfilename
```

7 Experimentelle Erprobung

Um die Funktionsfähigkeit des Stabilisierungssystems zu verifizieren, soll eine Evaluierung des Systems durchgeführt werden. Im Wesentlichen handelt es sich hierbei um eine Messung der Rückstellgenauigkeit auf der einen und um die Geschwindigkeit bei der Stabilisierung auf der anderen Seite.

Die Messung wurde durch die Tatsache erschwert, dass die Gyroskope auf Dauer einen Drift aufweisen und somit die Messungen nicht direkt verwendet werden können. Stattdessen sollten Messreihen mit vielen Wiederholungen durchgeführt und die Ergebnisse gemittelt werden. Es ist davon auszugehen, dass der Drift, der durch die Gyroskope entsteht, prinzipiell eine statistische Streuung aufweist und im Mittel vieler Messwerte bei Null liegt.

Die Rückstellgenauigkeit sollte dann im besten Fall im Bereich des vorhandenen Drifts der Gyroskope liegen, da dieser Fehler immer im System vorhanden ist und hierbei berücksichtigt werden muss. Der Drift kann bei den Geschwindigkeitsmessungen der Stabilisierung vernachlässigt werden, da es sich hierbei immer nur um kurzzeitige Messungen handelt, in die der Drift nur minimal einfließen würde.

7.1 Erweitertes Testsytem

Die Tests mussten so durchgeführt werden, dass man sie zu jedem Zeitpunkt, gegebenenfalls mit anderen Parametern des Stabilisierungsprogramms, wiederholen kann, um Vergleiche zwischen verschiedenen Parametern ziehen zu können. Zusätzlich bieten die wiederholbaren Tests auch die Möglichkeit, den Drift zu berücksichtigen, so dass über mehrere Tests überprüft werden, ob sich der Fehler im Bereich des normalen Sensordrifts oder darüber hinaus befindet. Die Tests sollten sowohl hinsichtlich der abgefahrenen Positionen als auch mit der gefahrenen Geschwindigkeit jeweils gleich sein. Dadurch ist es ausgeschlossen, dass man zum Beispiel das System manuell an einen mechanischen Anschlag schiebt.

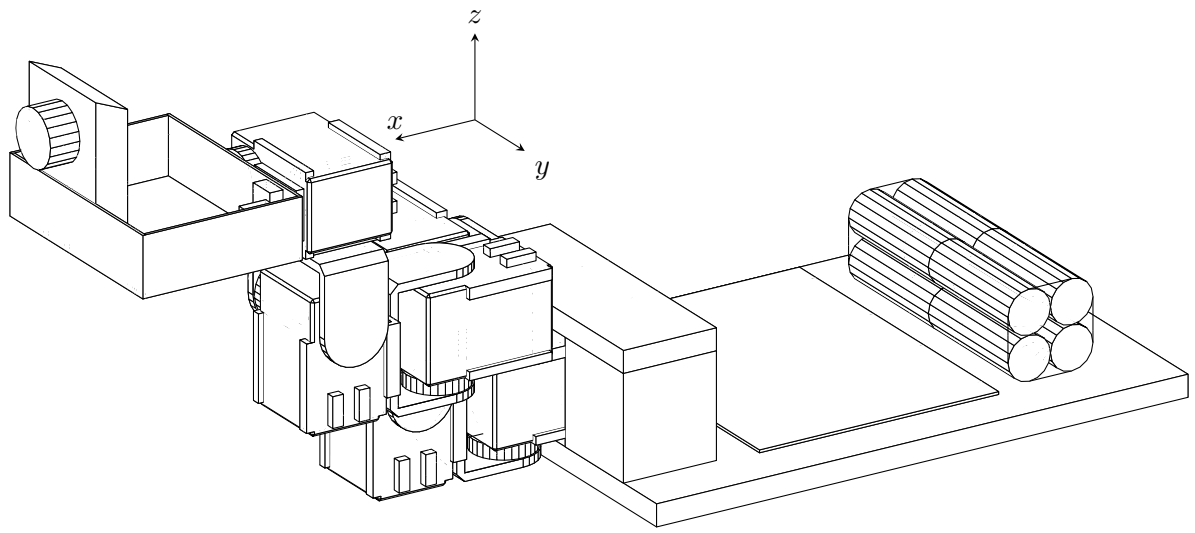

Abb. 7.1: Skizze des erweiterten Testsystems

Aus diesem Grund wurde das Testsystem dahingehend erweitert, dass neben den für die Stabilisierung nötigen Servos noch zusätzliche Testservos eingefügt wurden, um eine Umgebung mit wiederholbaren Tests zu gewährleisten. Die drei neuen Servos wurden am unteren Teil der kinematischen Kette und in der gleichen Konstellation bezüglich der Achsen eingefügt, wie die ursprüngliche kinematische Kette auch.

Diese Servos werden von den Stabilisierungsroutinen selbst nicht angesprochen. Jedoch wurde das Programm dahingehend erweitert, dass eine beliebige, vorher zu definierende Trajektorie, abgefahren wird. Hierzu können verschiedene Wegpunkte definiert werden, bei denen jeweils für alle Servos ein Sollwinkel und gleichzeitig auch eine Sollgeschwindigkeit vorgegeben wird. Die Geschwindigkeiten können sich von Wegpunkt zu Wegpunkt verändern, sind jedoch dazwischen jeweils konstant.

Dadurch, dass die Versuchstrajektorie und auch die Stabilisierung auf dem gleichen Controller ausgeführt wurden, musste die Frequenz der Timerfunktion auf $100Hz$ herabgesetzt werden, was die Testergebnisse eventuell minimal beeinflusst. Um die Tests komplett unter realen Bedingungen durchzuführen, wäre hier ein zweiter Mikrocontroller nötig, um die Versuchstrajektorie an die neuen Servos zu schicken.

7.2 Messung des Sensordrifts

Die Messung des Sensordrifts gestaltet sich relativ einfach. Hierbei wird das komplette System zum einen für die Initialisierung, aber auch danach in einer Ruhelage gehalten. Nachdem die Initialisierung der Gyroskope abgeschlossen ist, wird stetig anhand der kontinuierlich gemessenen Drehrate ω_i in

den Gyroskopen und dem zwischen den Messungen verstrichenen Zeitraum Δ_t der interne Drehwinkel $\alpha = \sum \Delta_t \cdot \omega_i$ jedes einzelnen Gyroskops berechnet. Danach wurde über fünf Minuten hinweg alle 30 Sekunden diese Position auf dem Hostrechner ausgegeben. Da die Gyroskope sich die ganze Zeit über in der Ruhelage befanden und damit die tatsächliche Postion jeweils null hätte sein müssen, ist die ausgegebene Position der aufgetretene Drift.

Während der Tests wurden verschiedene Initialisierungseinstellungen für die Gyroskope überprüft. Die Messergebnisse für die Intervalle von jeweils 60 Sekunden ist im Anhang B angegeben. Für die Bestimmung des Sensordrifts ist nur der Absolutbetrag der Abweichung von Interesse. Während der Tests gab es Abweichungen sowohl in die positive als auch in die negative Richtung. In den aufgeführten Tabellen ist nur noch der absolute Fehler eingetragen, und aus diesem errechnet sich auch der mittlere Fehler.

Während der Tests wurde in Intervallen von jeweils 30 Sekunden der gemessene Drehwinkel ausgegeben. Die Tests wurden mit drei verschiedenen Gyroskopen und jede Messreihe mit unterschiedlichen Initialisierungseinstellungen aufgenommen. Jede Messreihe besteht aus fünf identischen Versuchen. Als Vergleich zwischen den Initialisierungen können dann die Mittelwerte dieser Versuche verwendet werden. Somit fallen ausreißende Messwerte nicht so stark ins Gewicht.

In Abbildung 7.2 sind diese Tests als Graph dargestellt und die Mittelungen der Messwerte in den Zeitabständen von jeweils 30 Sekunden eingetragen. Hierbei wurden zum einen die fünf Versuche untereinander und zudem auch die Messwerte der drei verschiedenen Gyroskope gemittelt. Jeder Messpunkt besteht also aus dem Mittel von 15 verschiedenen Messwerten. Die Initialisierungseinstellungen unterschieden sich in der Anzahl der Sensorabfragen, die zur Bestimmung des Nullpunkts verwendet wurden und den Abständen, die zwischen den letzten drei Vorinitialisierungen liegen durften, so dass die komplette Initialisierung abgeschlossen wurde (siehe Kapitel 6.2.2).

Aus Graph 7.2 ist schnell ersichtlich, dass es sich bei dem gemessenen Fehler, wie zu erwarten, um einen linearen Fehler handelt. Hierbei wird bei der Initialisierung durch Rauschen eine begrenzte Auflösung der A/D-Wandler oder der internen Darstellung der Nullpunkte ein fehlerhafter Nullpunkt der Gyroskope erkannt. Bei der Berechnung des Integrals der Messwerte wird dieser Fehler linear über die Zeit aufsummiert. Des weiteren kann man aus dem Graphen sogar etwas Signalrauschen entnehmen, da die zeitlichen Messwerte nicht exakt auf einer gemeinsamen Geraden liegen, sondern durch das Signalrauschen etwas verschoben sind.

Wie zu erwarten, liefern die Initialisierungseinstellungen, bei denen mehr Messwerte in die Berechnung des Nullpunkts einfließen, eine genauere Berechnung dieses Nullpunkts. Die Verwendung von mehreren Abfragen für die Initialisierung entspricht einem Tiefpassfilter. Erhöht man die Anzahl dieser Abfragen, so entspricht das einem Tiefpassfilter, bei dem die Grenze der herauszufilternden hohen Frequenzen niedriger liegt. Das Signalrauschen mit hohen Frequenzen wird also eher rausgefiltert als mit einer kleineren Anzahl von Abfragen.

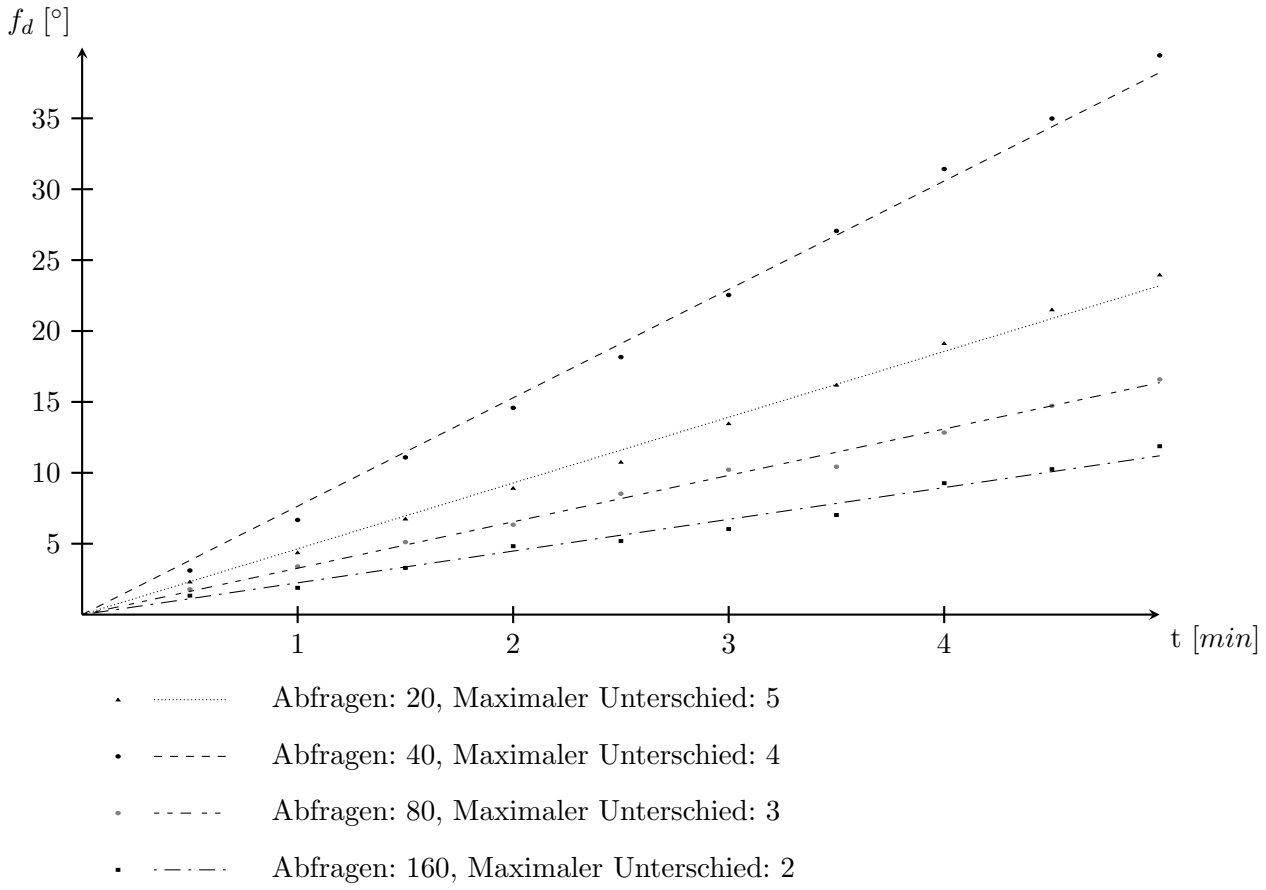

Abb. 7.2: Fehler durch Drift bei unterschiedlichen Initialisierungseinstellungen

Je größer die Anzahl der Abfragen, desto länger ist jedoch auch die Initialisierungszeit. Während der Evaluierung des Systems fällt diese Initialisierungszeit jedoch nicht so sehr ins Gewicht, so dass die weiteren Tests mit den hier idealsten Initialisierungseinstellungen von 160 Abfragen durchgeführt werden. Die Tests haben gezeigt, dass bei dieser Initialisierung mit einem Sensordrift von ca. $2,2\,°/s$ zu rechnen ist.

In den Abbildungen 7.3, 7.4 und 7.5 sind die gemessenen Driftfehler für alle drei Gyroskope aufgeführt. Die Graphen zeigen den gemessenen Fehler für die einzelnen Versuche. Die Initialisierungseinstellung war in jedem Fall mit 160 Abfragen für die Bestimmung des Nullpunkts der Gyroskope gleich. Zwischen den letzten drei Vorinitialisierungen durfte ein maximaler Unterschied von 2 liegen, damit die Initialisierung abgeschlossen wurde. Die Abbildungen zeigen die Streuung, die durch den Sensordrift hervorgerufen wird. Während im Zeitraum einer Minute noch Unterschiede von 5-10° zwischen den einzelnen Versuchen liegen, laufen die Positionen durch den Drift danach stark auseinander. Dies unterstreicht die Tatsache, dass die Gyroskope nicht für eine Langzeitstabilisierung eines Systems verwendet werden können. Für einen solchen Fall sollte ein weiteres Stabilisierungssystem übergeordnet werden.

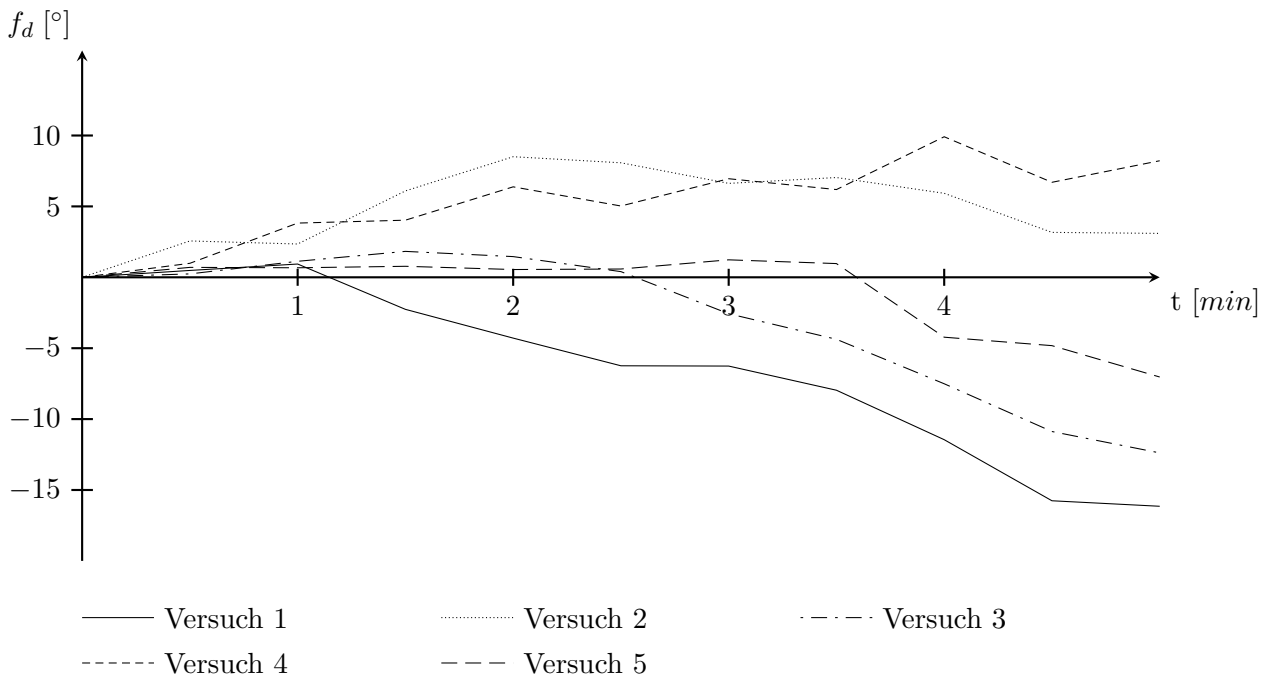

Abb. 7.3: Gemessener Driftfehler (Gyroskop 1)

7.3 Rückstellgenauigkeit

Um die Rückstellgenauigkeit zu messen, wird das erweiterte Testsystem verwendet. Um Versuchsreihen durchführen zu können, muss die Basis des Systems reproduzierbar bewegt werden. Für die Testservos wurde eine Trajektorie mit insgesamt drei Wegpunkten vorgegeben, die nach der Initialisierung kontinuierlich nacheinander abgefahren werden. Nach einer Minute wird der Programmzustand so gesetzt, dass die Trajektorie nur bis zum Endpunkt abgefahren wird, um die drei Testaktoren dort anzuhalten. Der Endpunkt wurde so gewählt, dass die Testaktoren sich, wie zu Beginn des Tests, in Neutrallage befanden.

Gemessen wurde nun ausschließlich die Position der Stabilisierungsservos nach dem Beenden der Testtrajektorie. Da sich diese Servos nach der Initialisierung in Neutralstellung und die Testservos zu diesem Zeitpunkt wieder in ihrer Ausgangsstellung befinden, müssten die Stabilisierungsservos nun ebenfalls wieder die ursprüngliche Neutralstellung einnehmen. Dies setzt voraus, dass die Stabilsierung als auch die Sensoren optimal arbeiten.

Als Testtrajektorie wurden zunächst zwei Wegpunkte gewählt. Der erste Wegpunkt fällt mit dem letzten Wegpunkt der Trajktorie zusammen. In dieser Stellung gibt es keine Rotationen der Testservos um die drei elementaren Raumachsen. Beim zweiten Wegpunkt wurde der erste Testservo um 30° bewegt. Dies entspricht einer Drehung um die z-Achse im Basiskoordinatensystem. Dieser Winkel wurde gewählt, da der Arbeitsbereich der Servos mechanisch bedingt auf ±45° eingeschränkt ist.

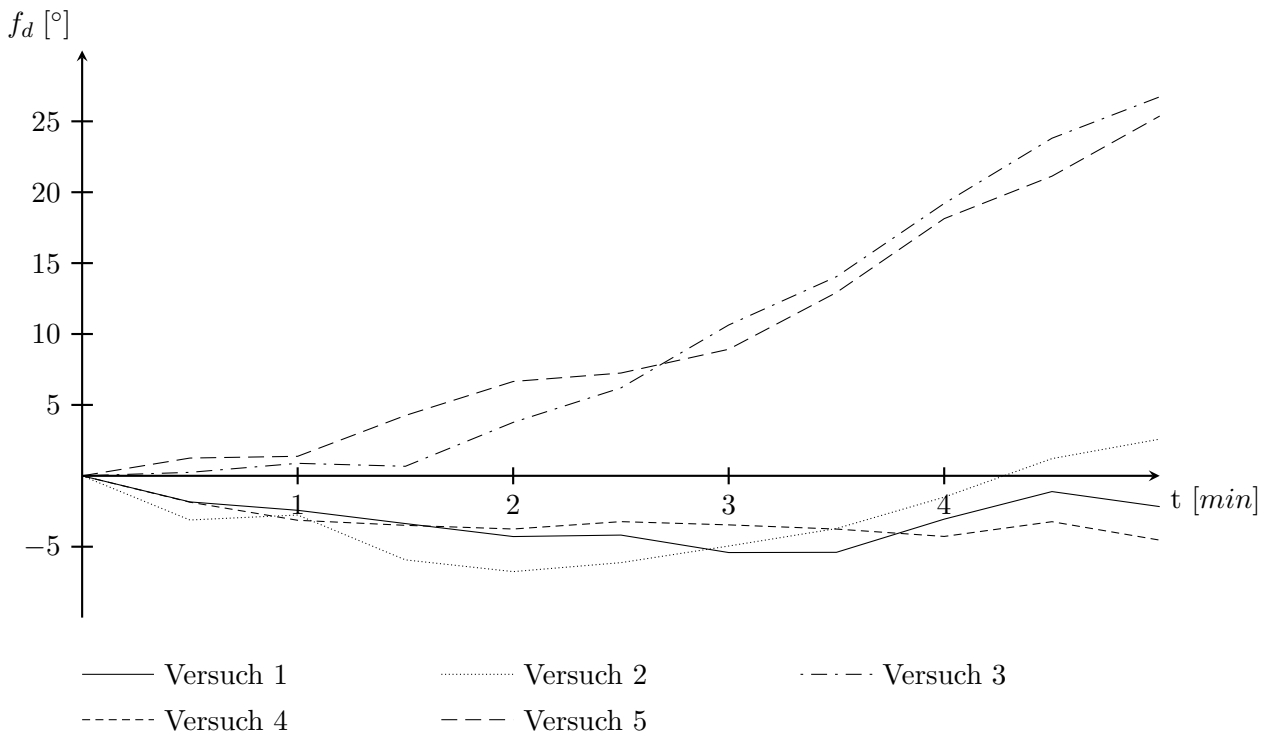

Abb. 7.4: Gemessener Driftfehler (Gyroskop 2)

7.3.1 Störungen durch laufende Servomotoren

Während der ersten Tests hat sich gezeigt, dass die Rückstellgenauigkeit der Stabilisierung scheinbar mit einem systematischen Fehler behaftet ist. Es wurde über den Zeitraum von 15 Sekunden kontinuierlich die Trajektorie abgefahren und gleichzeitig durch das Stabilisierungsprogramm die Orientierung des Endeffektors stabilisiert. Nach diesen 15 Sekunden wurde die Trajektorie noch bis zum Ende gefahren, damit die Testservos wieder in ihrer ursprünglichen Orientierung standen. Eine Messung der Position aller drei Stabiliiserungsservos hätte nun im Idealfall jeweils einen Winkel von 0° ausgeben müssen. Durch den Drift der Gyroskope war mit einem Winkel von ±1° zu rechnen. Jedoch stand die Position eines Servos im Bereich von $3 - 5°$. Eine Fehlersuche im Stabilisierungsprogramm ergab keinen weiteren Aufschluss über eine mögliche Fehlerquelle.

Im Vergleich zu den Driftmessungen wurden während des Tests der Rückstellgenauigkeit die Servos des Systems bewegt, so dass auch die Servos als mögliche Fehlerquelle in Frage kommen konnten. Diese These wurde überprüft, indem die Gyroskope vom Endeffektor des Systems abmontiert und stationär abgelegt wurden. Nun wurde durch die Testservos über eine Minute die vorgegebene Trajektorie abgefahren und gleichzeitig von den Gyroskopen die aufintegrierte Position ausgegeben. Das Ergebnis dieses Tests war, dass trotz der Ruhelage, in der sich die Gyroskope während dieser Zeit befanden an einem Gyroskop ein Drift von ca. 10° gemessen wurde. Nach weiteren Tests, bei denen der Motor eines

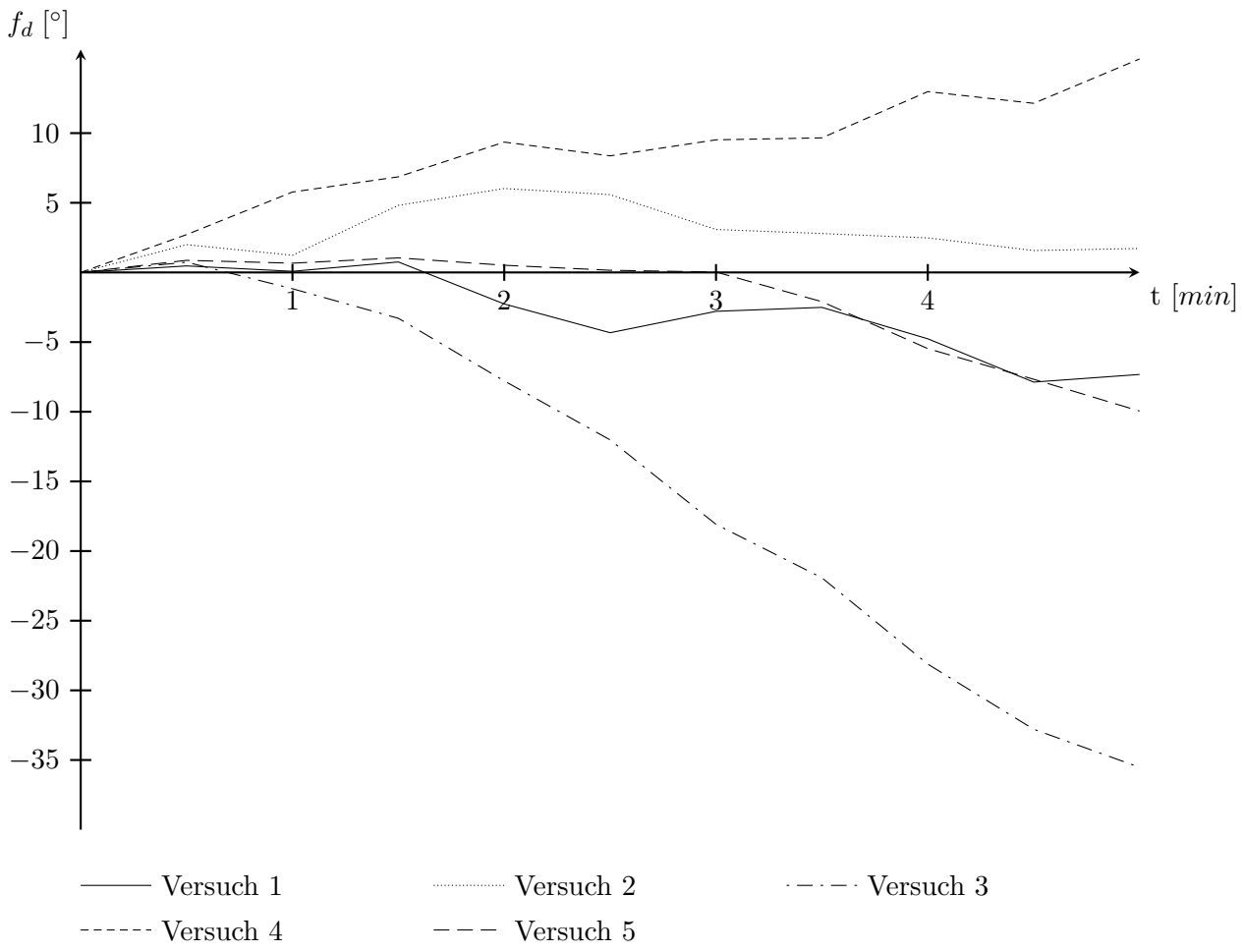

Abb. 7.5: Gemessener Driftfehler (Gyroskop 3)

Servos stromlos gemacht wurde und lediglich die Kommunikation über den Datenbus mit den Servos erfolgte, trat der gleiche Fehler auf.

Eine Überprüfung des Stabilisierungsprogramms ergab, dass direkt vor dem Auslesen der A/D-Wandler eine Statusmeldung von den Testservos abgefragt wurde, ob sich diese noch bewegen, oder ob sie den nächsten Wegpunkt erreicht haben. Diese Kommunikation wurde nun innerhalb des Programms an die Stelle nach der Abfrage der A/D-Wandler verschoben. Weitere Tests zeigten nun, dass sich der Sensordrift der Gyroskope gebessert hatte, während die Testtrajektorie abgefahren wurde.

Für die Bestimmung des Sensordrifts während einer Kommunikation mit den Servomotoren wurde eine weitere Messreihe gestartet. Hierbei wurden jeweils zehn Messungen mit und ohne Kommunikation zu den Servomotoren aufgenommen. Wie bei den Messungen des Sensordrifts wurden die Gyroskope dabei in einer Ruhelage gehalten. Die Messungen dauerten jeweils eine Minute, nach der die durch die Gyroskope gemessene Orientierung ausgegeben wurde. Da die Gyroskope dabei keiner Drehung

ausgesetzt waren, sollte diese gemessene Orientierung jeweils 0° betragen. Jede Abweichung von diesem Wert war auf den Sensordrift und etwaige Störungen zurückzuführen.

Die erreichten Messwerte sind in Anhang C aufgeführt. Zusätzlich sind die Messwerte in Abbildung 7.6 zu sehen. In dieser Abbildung sind für jedes Gyroskop die Driftfehler nach jeweils einer Minute durch kleine horizontale Linien aufgetraten. Der mittlere Driftfehler wird durch eine etwas größere und dick gedruckte Linie repräsentiert. Für jedes Gyroskop gibt es die Messungen sowohl ohne, als auch mit Kommunikation zu den Servos.

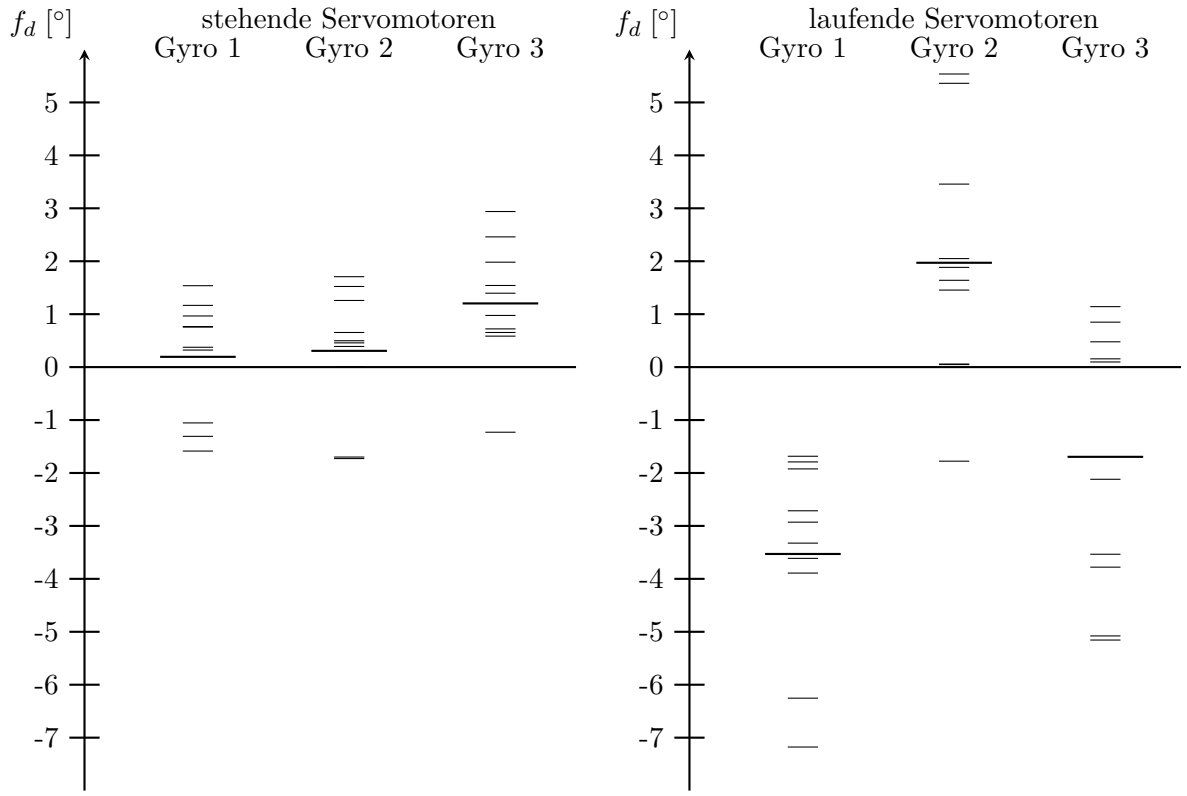

Abb. 7.6: Driftfehler bei laufenden Servos

Auffallend an dieser Abbildung ist, dass die Streuung der Driftfehler mit laufenden Servos wesentlich größer ausfällt. Desweiteren ist der mittlere Driftfehler wesentlich deutlicher von 0° entfernt als mit stehenden Servos. Bei Gyroskop 1 liegen sogar alle Driftfehler im negativen Bereich. Man kann hier davon ausgehen, dass es zu einer Verschiebung des Nullpunktes gekommen ist, oder dass es Störsignale durch die Servos gibt, die eine Verschiebung der gemessenen Orientierung bewirken.

In Kapitel 13.7.5 des Handbuchs über den Mikrocontroller SH7125 [33] schreibt die Firma Renesas, dass man bei der Konzipierung eines Boards für diesen Mikrocontroller darauf achten sollte, dass die Analogleitungen zu den A/D-Wandler möglichst weit von den restlichen Digitalleitungen entfernt liegen sollten, da eine mögliche Induktion die A/D-konvertierten Werte beeinflussen kann. Während der A/D-Wandlung der Sensorwerte findet im Programm keine Kommunikation zu den Servos statt. Den-

noch kommt es zu einer Beeinflussung der gewandelten Digitalwerte. Es konnte auch keine Veränderung des Signals von den Gyroskopen festgestellt werden, wenn die Kommunikation zu den Servomotoren gestartet wurde. Es ist davon aus zu gehen, dass die Kommunikation oder die laufenden Servomotoren die A/D-Wandler des Mikrocontrollers beeinträchtigen. Der hier beschriebene Fehler kann nicht weiter eingeschränkt werden, da der Mikrocontroller sowohl mit den Testservos als auch mit den Stabilisierungsservos kommunizieren muss, um die gewünschten Tests durchzuführen.

7.3.2 Nichtlinearität der Eingangssignale

Nachdem der auftretende Fehler durch die Kommunikation zu den Servomotoren etwas eingeschränkt werden konnte, wurden weitere Versuche durchgeführt. Hierbei wurde die Stabilisierung des Endeffektors abgeschaltet und ausschließlich die vorgegebene Trajektorie durch die Testservos abgefahren. Die Gyroskope wurden bei diesem Test wieder am Endeffektor befestigt. Sie folgen also der Fahrt der Testservos. Allerdings befinden sie sich am Ende des fünfzehnsekündigen Tests wieder in ihrer Ursprungsposition und sollten also wieder einen ähnlichen Wert haben, wie bei den vorigen Tests, bei denen sie sich die ganze Zeit über in einer Ruhelage befanden.

Die Ergebnisse dieses Tests sind für verschiedene Trajektorien in den Tabellen im Anhang C aufgeführt. Wie man dort sehen kann, sind für einzelne Trajektorien und Gyroskope Fehler von maximal 5° pro 15 Sekunden gemessen worden. Dies ist ein wesentlich höherer Wert als er bei den Gyroskopen in Ruhelage erreicht wurde.

Eine mögliche Ursache könnte ein Nichtlinearitätsfehler, wie er in Abbildung 13.5 des Handbuchs über den Mikrontroller SH7125 [33] beschrieben ist. Um diese Möglichkeit zu überprüfen, wurden am Eingang des A/D-Wandler verschiedene Spannungen angelegt und der zugehörige, gewandelte Digitalwert ausgelesen. Als Eingangsspannungen U_{in} wurde ein Intervall von $1 - 4V$ gewählt. Dies entspricht in etwa dem Bereich, der auch von den Gyroskopen bei den Drehbewegungen zurückgemeldet wird.

In Abbildung 7.7 ist ein Graph abgebildet, der die gemessenen Digitalwerte zu den Eingangsspannungen zeigt. Die dünne Gerade entspricht einer idealen A/D-Wandlungscharakteristik. Die Punkte entsprechen jeweils einem Messpunkt und die dicke Gerade der Charakteristik des verwendeten A/D-Wandlers. Man sieht, dass durch den A/D-Wandler keine Nichtlinearität entsteht. Die gemessene Gerade ist zwar etwas steiler als die ideale A/D-Charakteristik, aber für das hier beschriebene Problem kann der A/D-Wandler nicht verantwortlich sein.

Es ist davon aus zu gehen, dass sich die Störungen durch die Servos auf verschiedenen Spannungsbereichen unterschiedlich auswirken. Wenn höhere Spannungen durch die Störungen stärker gedämpft würden, dann wäre dies eine Erklärung für den hier festgestellten Fehler.

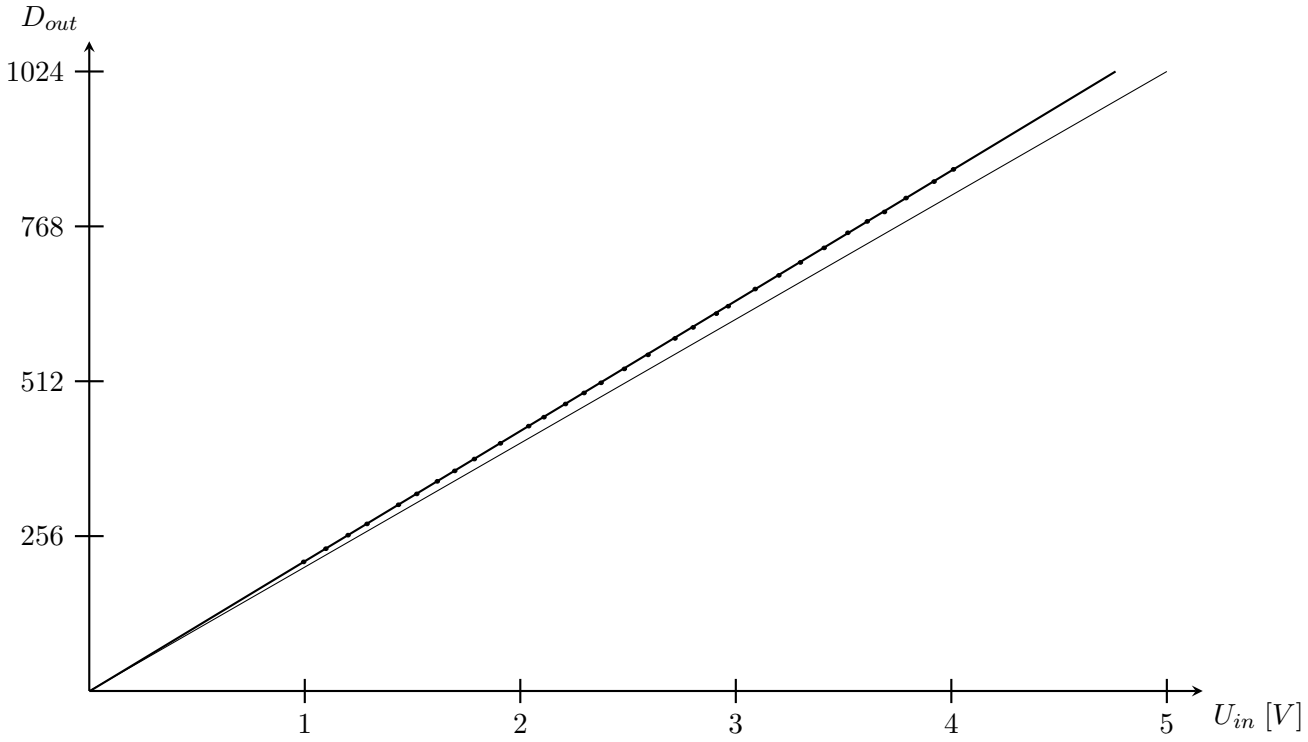

Abb. 7.7: Fehler des A/D-Wandlers

Auch beschränkt sich die maximale Abweichung nicht auf die Gyroskope in Bewegungsrichtung der Servos. Es sind auch Gyroskope von diesem Fehler betroffen, die während der Testbewegung nicht um ihre Messachse gedreht werden und somit eigentlich Driftfehler im Bereich der Nullpunktmessung des letzten Unterkapitels liefern sollten.

Unter den gegebenen Umständen ist es leider nur bedingt möglich, eine Aussage über die Rückstellgenauigkeit des Stabilisierungsprogramms zu liefern. Die Messungen sind für verschiedene Trajektorien in den Tabellen in Anhang C angegeben. Die Rückstellgenauigkeit befindet sich jeweils in etwa im Bereich der Abweichungen ohne Stabilisierung. Jedoch ist dies ohne Kenntnis über die Störquelle und deren Auswirkung auf die A/D-Wandlung nicht aussagekräftig.

Über eine Zeitdauer von 15 Sekunden konnte das System mit einem Rückstellfehler von maximal 5° stabilisiert werden. Das heisst, dass das entwickelte System in einem Intervall von 15 Sekunden über ein Langzeitstabilisierungssystem neue Servopositionen übermittelt bekommen muss, falls man einen maximalen Rückstellfehler von 5° zulassen kann. Bei diesem Fehler fließt jedoch ein systematischer Fehler durch die Servobewegung mit ein. Wenn es möglich ist diesen Störfaktor zu beheben, so ist damit zu rechnen, dass dieser Rückstellfehler innerhalb von 15 Sekunden etwa dem stets vorhanenden Sensordrift von 1-2° entspricht.

7.4 Messung der Rückstellgeschwindigkeit

Der Geschwindigkeitstest bedingt zum einen, dass ebenfalls wiederholbare Trajektorien mit dem System abgefahren werden. Zusätzlich muss nun aber auch während der Stabilisierung der Drehwinkel des Endeffektors gemessen werden. Dies kann auf zwei verschiedene Arten realisiert werden.

Zum einen kann in der Timerfunktion die Position der Test- und Stabilisierungsservos abgefragt und die beiden Rotationsmatrizen

$$R_{test} = R(z;\theta_{tz}) \cdot R(y;\theta_{ty}) \cdot R(x;\theta_{tx}) \tag{7.1}$$

und

$$R_{stab} = R(z;\theta_{sz}) \cdot R(y;\theta_{sy}) \cdot R(x;\theta_{sx}) \tag{7.2}$$

aus den Drehpositionen der Test- (θ_{tx}, θ_{ty} und θ_{tz}) und Stabilisierungsservos (θ_{sx}, θ_{sy} und θ_{sz}) errechnet werden. Multipliziert man nun beide Rotationsmatrizen miteinander, so erhält man die Gesamtrotationsmatrix

$$R_{ges} = R_{test} \cdot R_{stab} \tag{7.3}$$

des Endeffektors bezüglich des Basiskoordinatensystems. Aus ihr kann mit Hilfe der Formeln 3.16 und 3.15 sowohl die Drehachse, als auch der Drehwinkel des Endeffektors bestimmt werden. Dieses Verfahren setzt voraus, dass während der Timerfunktion viel Kommunikation mit den Servos stattfindet und die aktuellen Drehwinkel zwischengespeichert und später ausgelesen werden. Die Kommunikation mit den Servos und der Sensordrift der Gyroskope sollte bei diesem Test jedoch nur einen kleinen Teil des Fehlers aus machen, da die Zeitspanne des Tests nur wenige Sekunden beträgt.

Eine weitere Möglichkeit wäre die Orientierung des Endeffektors während der Stabilisierung zu ermiteln, eine Digitalkamera am Endeffektor der kinematischen Kette anzubringen und die Rotationsmatrix des Endeffektorkoordinatensystems bezüglich des Basiskoordinatensystems mit Hilfe geeigneter Bildverarbeitungsalgorithmen zu bestimmen. Aus dieser Rotationsmatrix kann dann, wie im anderen Fall die Drehachse und der Drehwinkel des Endeffektors errechnet werden. Die Digitalkamera muss an einem externen Rechner, wie zum Beispiel dem Host-PC, angeschlossen werden und dort einen Film aufzeichnen. Aus diesem Film müssen dann die Einzelbilder extrahiert werden, um sie mit Hilfe der Bildverarbeitung weiter zu verarbeiten. Mit der Software *Camera Calibration Toolbox for Matlab* kann man die Orientierung einer Digitalkamera zu einem schachbrettartigen Muster bestimmen. Jedoch hat die Software bei ersten Tests keine befriedigende Genauigkeit geliefert. Deswegen wird bei den folgenden Tests die Orientierung des Endeffektors über die einzelnen Gelenkwinkelstellungen der Aktoren bestimmt.

Bei den Tests ist zu berücksichtigen, dass die Servomotoren in Ruhelage nicht exakt in ihrer Nulllage stehen. Durch das Gewicht des Manipulators werden die Motoren teilweise minimal in eine Richtung

gedrückt. Dadurch, dass der innerhalb der Servos montierte Positionssensor abtriebseitig installiert ist, kann die exakte Position des Servos in Ruhelage gemessen werden. Mit Hilfe dieser Positionen errechnet sich dann die Drehmatrix der Orientierung des Endeffektors in Ruhelage

$$R_{idle} = R(z;\theta_{tz,idle}) \cdot R(y;\theta_{ty,idle}) \cdot R(x;\theta_{tx,idle}) \cdot R(z;\theta_{sz,idle}) \cdot R(y;\theta_{sy,idle}) \cdot R(x;\theta_{sx,idle}). \quad (7.4)$$

Um nun den auftretenden Fehler während der Stabilisierung zu bestimmten, werden die Positionen der Servos in Ruhelage sowie zu einem gewünschten Zeitschritt benötigt. Die Rotationsmatrix, die zur Bestimmung des Fehlers heran gezogen wird, ist

$$R_{fehler} = R_{idle}^{-1} \cdot R_{ges}. \quad (7.5)$$

R_{idle} entspricht der Orientierung des Endeffektors zu Beginn des Tests. Sie wird negativ genutzt, um die Orientierung des Endeffektors zum Messzeitpunkt zurück zu drehen. Diese Berechnung ist nötig, da zu Beginn des Tests als Orientierung des Endeffektors nicht die Einheitsmatrix berechnet wird, sondern die Orientierung R_{idle}. Würde man diese Rechnung nicht durchführen, so würde in jedem Fall die Orientierung des Endeffektors zu Beginn des Tests als Fehler ermittelt werden.

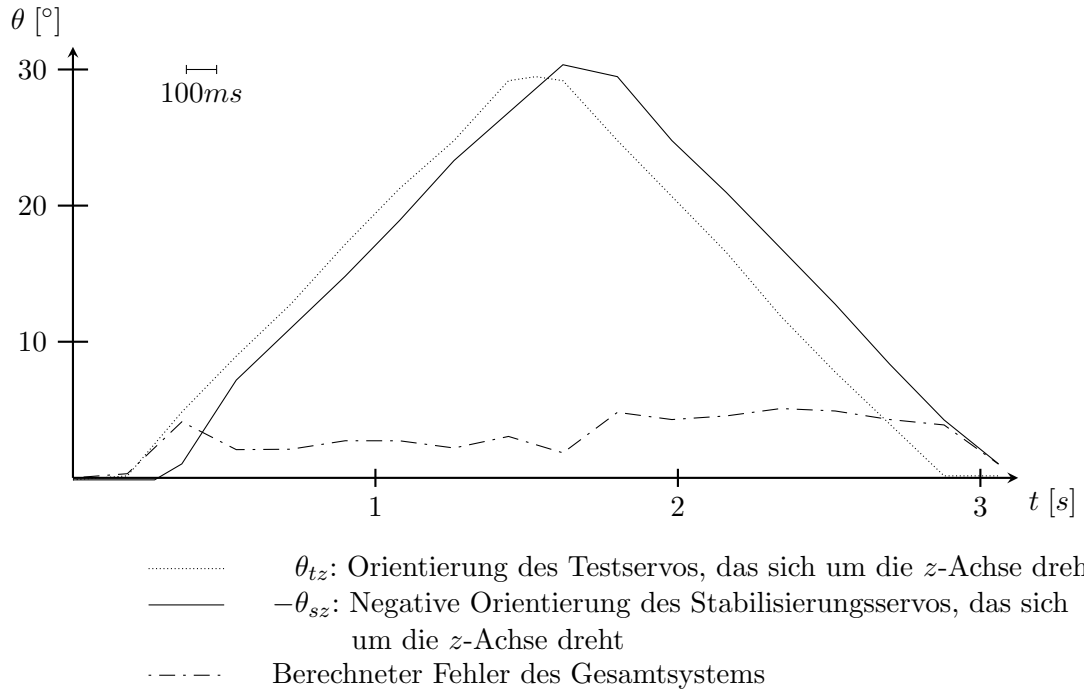

Abb. 7.8: Servobewegungen während der Stabilisierung, Weg: 0/0/0 → 30/0/0 → 0/0/0

In Anhang D sind für verschiedene Trajektorien die Stabilisierungsfehler in Zeitabständen von $180ms$ in unterschiedlichen Tabellen aufgeführt. Der mittlere Fehler beträgt bei einem bewegten Testservo insgesamt 3,115° im Durchschnitt, während der maximale aufgetretene Fehler 5,093° beträgt. Bei zwei gleichzeitig bewegten Testservos beträgt dieser Fehler im Schnitt 4,750° und maximal 7,193°.

Der Fehler durch die Kommunikation und den Sensordrift der Gyroskope, wie er in 7.3 beschrieben wurde kann innerhalb des gesamten Tests mit maximal ungefähr 1° wirken.

Bei den gewählten Trajektorien wurde für jeden Wegpunkt ein Tripel $(\theta_{tz}/\theta_{ty}/\theta_{tx})$ von Drehungen für die einzelnen Testservos vorgegeben. Die Reihenfolge der Drehungen um die folgt aus der Reihenfolge der Servos in der kinematischen Kette. So entspricht die Position 0/0/0 der Neutralstellung aller Servos und die Position 30/0/0 einer Drehung des Systems mit 30° um die z-Achse.

In diesem Unterkapitel sind vier Trajektorien als Abbildungen dargestellt. Zusätzlich zeigen die Abbildungen dann die Verzögerungszeit, mit der das Stabilisierungssystem die Stabilisierungsservos nachstellt. In den Abbildungen ist gleichzeitig auch die Verzögerungszeit von $100ms$ durch das Auslesen der Gyroskope, Berechnungen und Ansprechen der Stabilisierungsservos aufgezeichnet. Bei zwei dieser Trajektorien wurde nur ein Testservo bewegt, so dass die Stabilisierung auch nur auf einem Servo erfolgen musste. Im ersten Fall wurde hierbei eine Fahrt zu einem Wegpunkt und zurück zur Ausgangsposition durchgeführt, während im zweiten Fall nur eine einfache Fahrt zu einem Wegpunkt durchgeführt und dann dort die Position gehalten wurde.

In zwei weiteren Trajektorien wurden jeweils zwei Testservos gleichzeitig bewegt. Im ersten Fall wurde wieder nur ein Wegpunkt angefahren und die Testservos danach wieder in Neutralstellung bewegt. Im zweiten Fall wurden die Servos in schneller Folge gleichzeitig um jeweils 5° hin und her bewegt. Damit sollte ein mögliches Rütteln oder Vibrieren simuliert werden. In den Graphen sind dann die Gesamtdrehungen der Test- sowie der Stabilisierungsservos angegeben.

Abbildungen 7.8 und 7.9 zeigen für den Fall eines Bewegten Testservos über den Verlauf der Trajektorie die Positions des bewegten Testservos, sowie des korrespondierenden Stabilisierungsservos. Die Positionen des Stabilisierungsservos wurden hierbei negiert, um einen besseren Vergleich der beiden Servos ziehen zu können. Man sieht die Verzögerungszeit des Stabilisierungssystems anhand des zeitlichen Versatzes der gemessenen Positionen der beiden Servomotoren.

Während der Stabilisierung kommt es zwangsläufig zu einem Fehler. Dieser resultiert daraus, dass über einen Zeitraum von $90ms$ die Gyroskopgeschwindigkeit ausgelesen wird und danach innerhalb von $10ms$ das inverse Jacobi-Modell gelöst wird, um dann die Servos anzusteuern. Innerhalb von $100ms$ kommt es also durch die Umwelt zu einer Änderung der Orientierung des Gesamtsystems und nach dieser Zeit wird durch Steuerkommandos an die Servos entgegengewirkt. Diese Zeit kann reduziert werden, indem man den Zeitraum minimiert, in dem die Gyroskopgeschwindigkeit ausgelesen wird. Dies führt zu einer höheren Kommunikation zu den Servomotoren und einer Höheren Auswirkung des Signalrauschens auf die Stabilisierungsfunktion. Desweiteren wurde in dieser Arbeit vom gleichen Mikrocontroller auch das Testsystem angesteuert, wodurch das Zeitintervall für die Berechnungen erhöht werden musste.

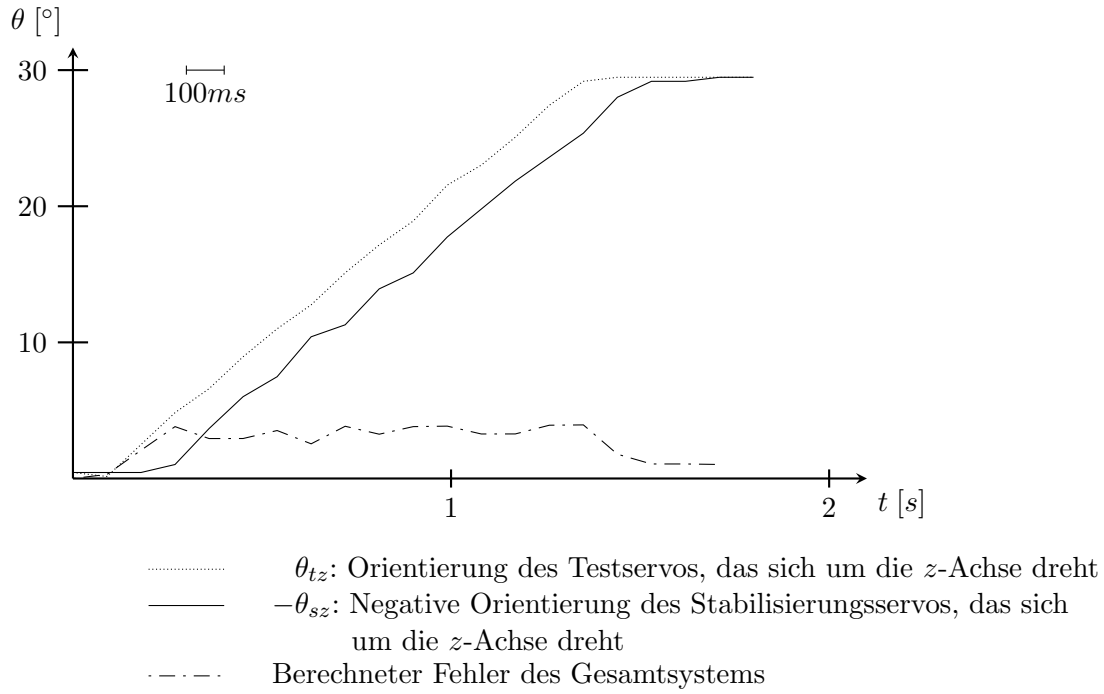

Abb. 7.9: Servobewegungen während der Stabilisierung, Weg: 0/0/0 → 30/0/0

Eventuell soll das Stabilisierungssystem auch in vibrierenden oder sich schnell bewegenden Gesamtsystemen eingesetzt werden. Abbildung 7.10 zeigt den Gesamtdrehwinkel und Gesamtfehler bei einer Trajektorie, bei der zwei Servos über fünf Sekunden kontinuierlich um 5° hin und her bewegt wurden.

Abbildung 7.11 zeigt den Verlauf des Gesamtdrehwinkels und den Gesamtfehler bei einer Stabilisierung. Während der Stabilisierung wurde von den Testservos eine Trajektorie abgefahren, bei der zwei Testservos bewegt wurden.

7.5 Aufnahmen eines stabilisierten externen Sensors

In den Abbildungen 7.12 und 7.13 sind zwei Fotoreihen gezeigt. Die Kamera hat sich jeweils im Endeffektor des Stabilisierungssystems befunden. Mit den Testservos wurde eine Trajektorie abgefahren, in der sich alle Servos zunächst in Neutralstellung befanden. Als nächsten Wegpunkt wurden die Testservos mit 30° um die z-Achse des Systems gedreht und danach mit 30° um die y-Achse. Im Anschluss wurden die Testservos wieder in Neutralstellung gefahren. Die Drehrate der einzelnen Servos während der Testtrajektorie betrug jeweils 20°/s. Der Gesamte Test lief über eine Zeitdauer von 4,59 Sekunden. Die erste Fotoreihe zeigt hierbei die Aufnahme der Kamera ohne Stabilisierung und die zweite Reihe zeigt die Aufnahme mit aktivierter Stabilisierung. In den Fotoreihen wurde jeweils jedes dritte Kamerabild bei einer Aufnahme von 10 Bildern pro Sekunde dargestellt.

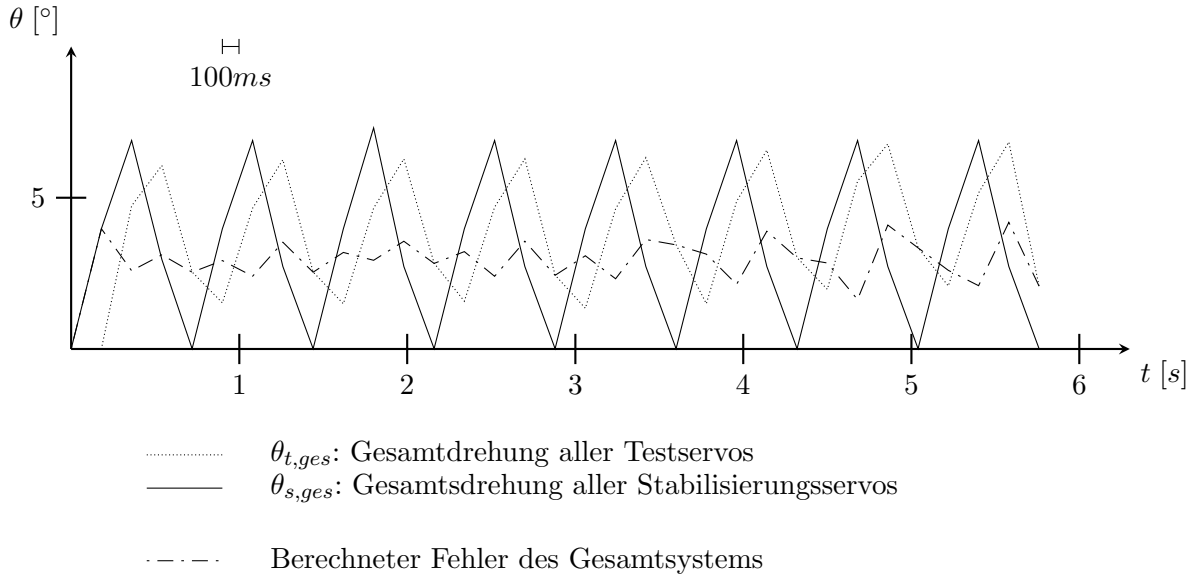

Abb. 7.10: Servobewegungen während der Stabilisierung, Weg: 0/0/0 → 5/0/5 → 0/0/0 → ...

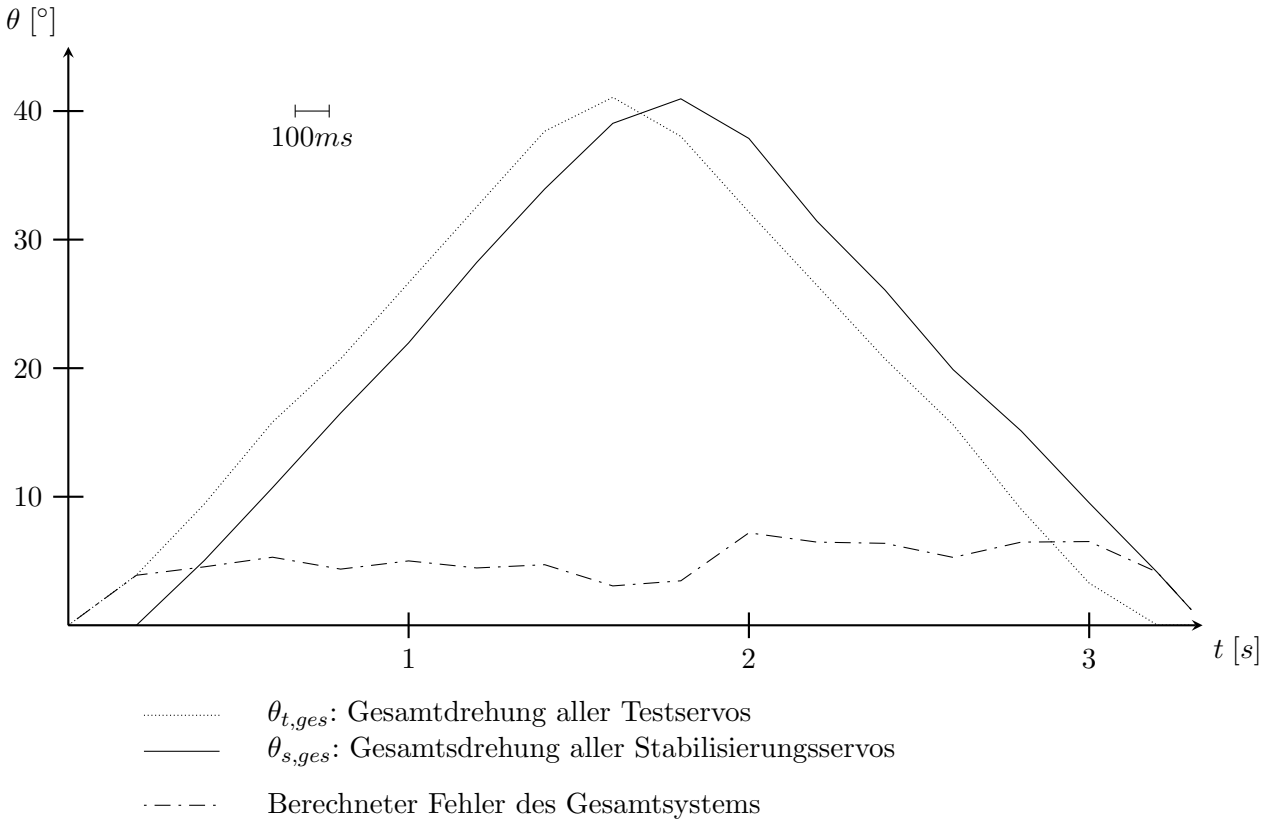

Abb. 7.11: Servobewegungen während der Stabilisierung, Weg: 0/0/0 → 30/0/30 → 0/0/0

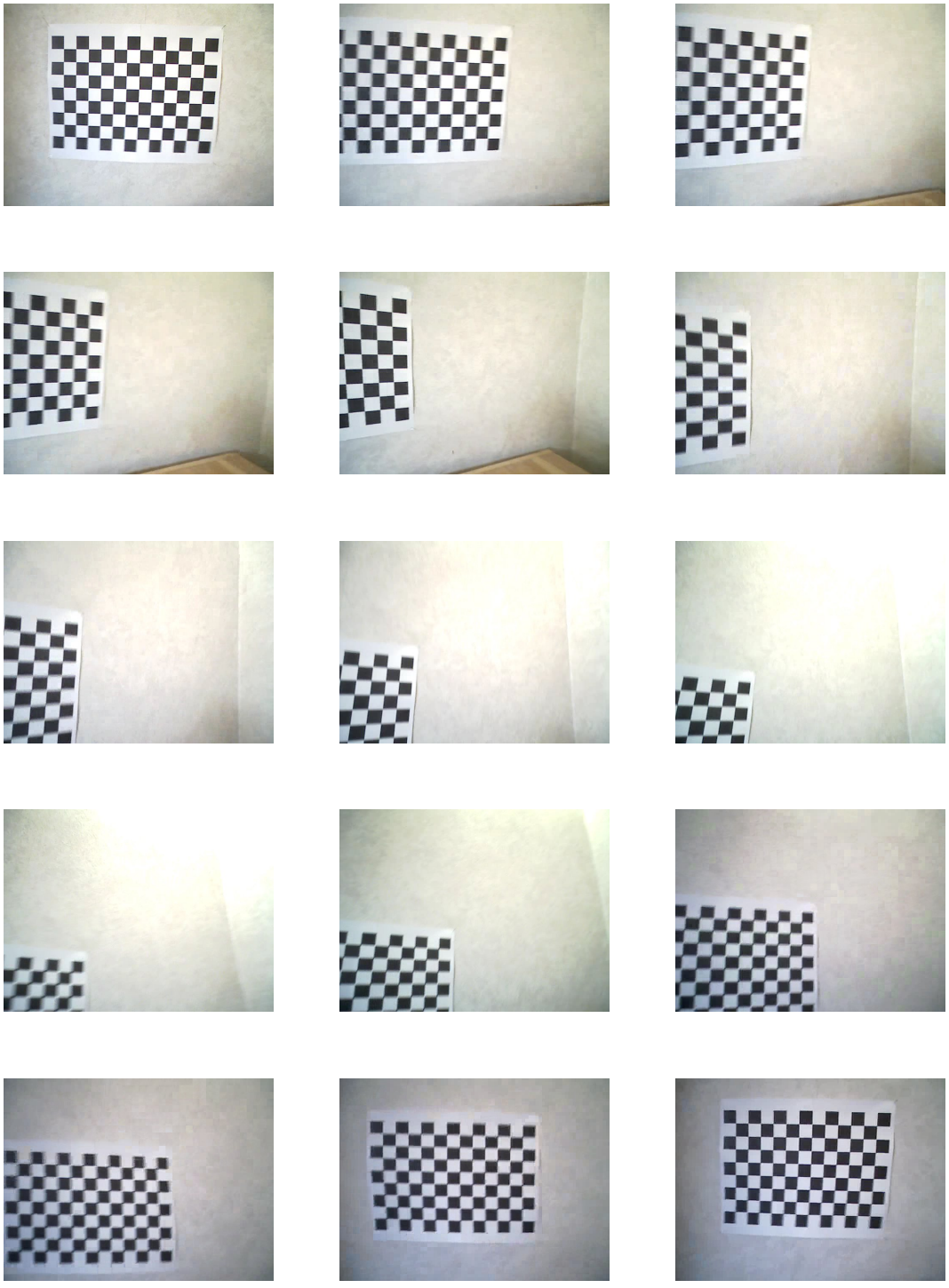

Abb. 7.12: Bilderreihe eines externen Sensors ohne Stabilisierung

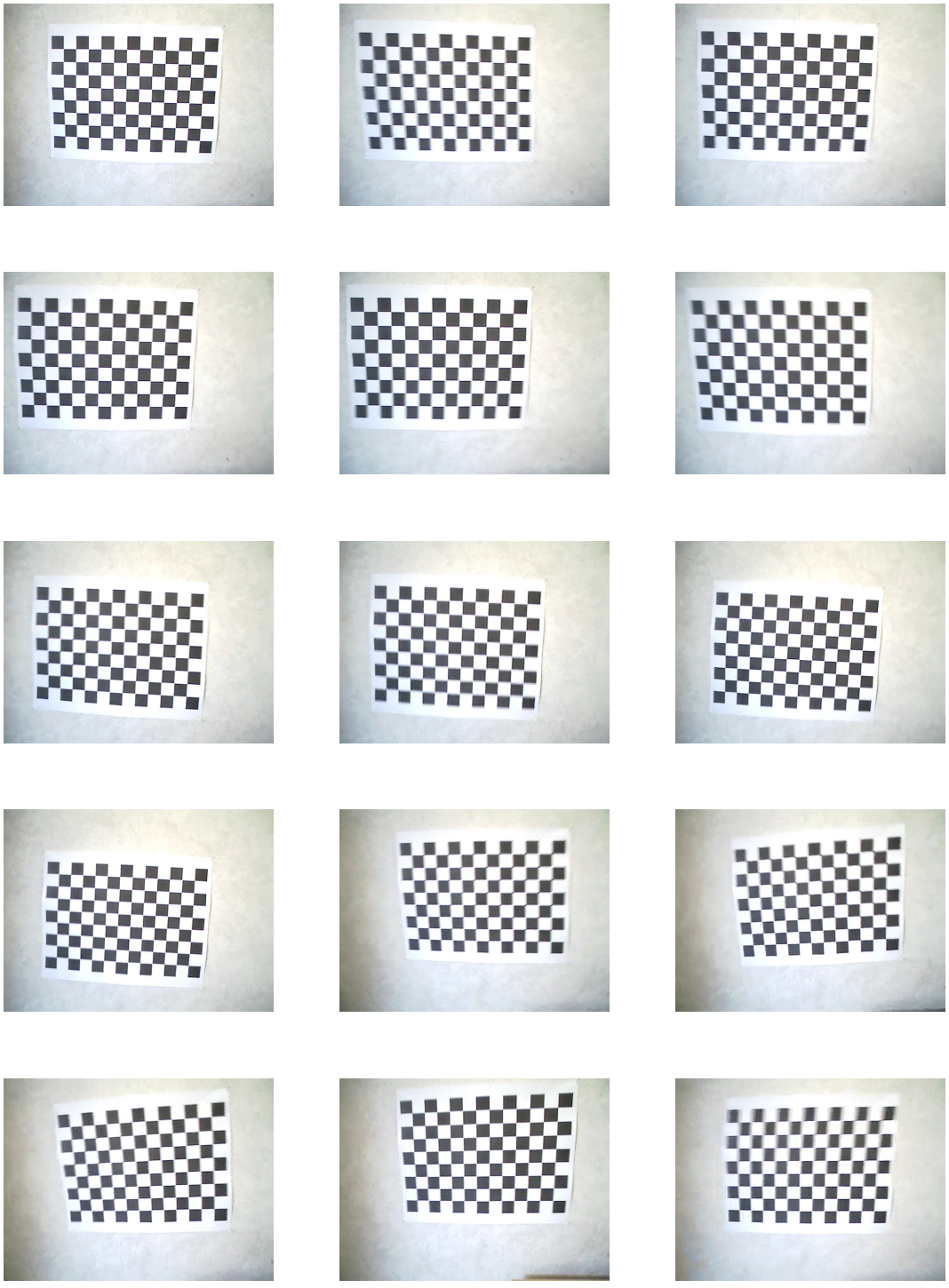

Abb. 7.13: Bilderreihe eines externen Sensors ohne Stabilisierung

8 Zusammenfassung und Ausblick

Beim Einsatz mobiler Robotersysteme sind externe Sensoren oft translatorischen und rotatorischen Bewegungen ausgesetzt. Diese Bewegungen können die Güte der aufgenommenen Sensordaten beeinflussen, indem zum Beispiel Kamerabilder verschwommen sind. Zusätzlich müssen die Daten in ein gemeinsames Koordinatensystem transformiert werden, um Messergebnisse zweier Zeitschritte miteinander vergleichen zu können.

8.1 Beschreibung der Entwicklung

Im Rahmen der vorliegenden Diplomarbeit wurde eine Software entwickelt, mit der es möglich ist, eine bestmögliche Orientierungsstabilisierung eines Endeffektors einer kinematischen Kette zu gewährleisten. Die Software kann als Basis eines Programmes für einen Mikrocontroller dienen. Die Stabilisierungsberechnungen können auf einem 32Bit Mikrocontroller mit 32MHz in $10ms$ durchgeführt werden. Als Sensoren dienen Gyroskope, die genau wie die Aktoren der kinematischen Kette an den Mikrocontroller angeschlossen sein sollten.

Die Software ist modular aufgebaut, so dass einzelne Hardwarekomponenten ausgewechselt werden können, ohne dass in der kompletten Software Änderungen vorgenommen werden müssen. Für jede Hardwarekomponente steht ein einzelnes Modul zur Verfügung, das anhand bestimmter Vorgaben erstellt werden muss und dann direkt in das Projekt eingebunden werden kann. Einige Softwaremodule wurden in der vorliegenden Arbeit bereits programmiert und stehen somit bereits zur Verfügung.

Um die Software bei anderen kinematischen Modellen einzusetzen, wurde ein Konfigurationswerkzeug entwickelt. Darin kann das kinematische Modell im Hinblick auf die Rotationen und die Position der Sensoren in der kinematischen Kette eingegeben werden. Danach kann direkt aus der Benutzeroberfläche oder später per Kommandozeile der nötige Quellcode für den Mikrocontroller erstellt werden. Der nicht vom Konfigurationswerkzeug erstellte Code ist nicht direkt im Werkzeug selbst hinterlegt, sondern wird dynamisch aus vorliegenden Dateien ausgelesen. Dieser Code kann also auch im Nachhinein noch direkt geändert werden, ohne den generierten Code selbst verändern zu müssen.

8.2 Evaluierung

Für die Entwicklung der Software und den späteren Test der Leistungsfähigkeit des Programms wurde ein Testsystem entwickelt. Dieses System besteht aus mehreren Servomotoren, die zu einer kinematischen Kette verbunden wurden. Sie bieten die Möglichkeit, in alle drei Raumrichtungen zu rotieren. Zusätzlich befinden sich im Endeffektor drei Einachsgyroskope, die Drehraten um alle drei Raumrichtungen messen. Für die Evaluierung wurde die kinematische Kette um drei Testservos erweitert, um mit ihnen eine Bahn abzufahren, während die drei Stabilisierungsservos weiterhin die Orientierungsstabilisierung des Endeffektors durchführen.

Das gesamte System wurde dann genutzt, um verschiedene Tests durchzuführen. Zuerst wurde der zu erwartende Sensordrift der Gyroskope bestimmt. Hierzu wurden verschiedene Initialisierungseinstellungen genutzt. Wie zu erwarten, lieferten die Einstellungen, bei den am meisten Sensorwerte miteinander gemittelt wurden, die besten Ergebnisse. Diese Einstellung wurde dann für alle weiteren Tests verwendet.

Anschließend wurde die Rückstellgenauigkeit des Systems bestimmt. Hierzu wurde mit den Testservos kontinuierlich eine Trajektorie abgefahren, während das System und die Stabilisierungsservos den Endeffektor stabilisierten. Nach 15 Sekunden wurde die Trajektorie noch zu Ende gefahren, bis die Testservos wieder in ihrem Ursprungszustand angelangt waren. Sobald die Stabilisierungsservos danach auch wieder eine Ruheposition erreicht hatten, wurde die Position der Stabilisierungsservos als Rückstellfehler ausgelesen, da diese Servos sich vor dem Test in Neutrallage befanden. Nach den 15 Sekunden ist ein maximaler Rückstellfehler von 5° gemessen worden.

Nachdem die Tests der Rückstellgenauigkeit abgeschlossen waren, wurde die Rückstellgeschwindigkeit gemessen. Hierzu wurde von den Testservos einmalig eine vorgegebene Trajektorie abgefahren. Während der Fahrt wurden kontinuiertlich die Positionen der Test- und Stabilisierungsservos gemessen. Aus diesen Positionen wurde die Orientierung des Endeffektors in Form einer Rotationsmatrix bestimmt. Aus dieser Matrix wurde als Fehler dann der Drehwinkel ermittelt. Da die Gyroskope über einen Zeitraum von $90ms$ ausgelesen und danach in $10ms$ die für die Stabilisierung nötigen Geschwindigkeiten in den Gelenken berechnet werden, kommt es zwangsläufig zu einem Fehler während der Stabilisierung. Der gemessene Fehler lag etwa im zu erwartenden Rahmen von maximal 7° bei gleichzeitiger Drehung um zwei Achsen mit einer Geschwindigkeit von jeweils 20°/s. Der Fehler kommt durch den zeitlichen Versatz der durch das Auslesen der Gyroskope, die Stabilisierungsberechnungen und das Ansteuern der Servos entsteht. Wenn diese Zeit, beispielsweise durch einen schnelleren Mikrocontroller, verkleinert werden kann, so verkleinert sich linear auch dieser Fehler, der während der Stabilisierung auftritt.

Während der Messungen der Rückstellgenauigkeit ist ein systematischer Fehler gemessen worden. Untersuchungen haben ergeben, dass die laufenden Servomotoren die Messungen der Gyroskope beeinträchtigen. Bei der Entwicklung von zukünftigen Boards für die Mikrocontroller sollte darauf geachtet

werden, dass die Analogleitungen zu den A/D-Wandler möglichst getrennt von den Digitalleitungen laufen. Dadurch kann es nicht so stark zu Einstreuungen und damit zu Störungen des Analogsignals kommen. Der auftretende Fehler bei der Rückstellgenauigkeit sollte dadurch minimiert werden. Grundsätzlich kann man sagen, dass ein Stabilisierungssystem nur die Qualität des schlechtesten Bauteils erreichen kann. Sollten Servos, A/D-Wandler oder Gyroskope nicht einwandfrei funktionieren, so wirkt sich dies auf das komplette Stabilisierungssystem aus.

8.3 Ausblick

8.3.1 Langzeitstabilisierung

Ein Problem bei der Orientierungsstabilisierung mit Hilfe von Inertialsensoren ist der auftretende Drift. Durch den Drift kann keine Langzeitstabilisierung des Systems erreicht werden. Denkbar wäre ein System mit zwei Stabilisierungsstufen. In der unteren Stufe würde das vorliegende Stabilisierungssystem arbeiten und in einer darüberliegenden Stufe ein System, das mit Hilfe von externen Sensoren eine Langzeitstabilisierung gewährleistet. Dieses System könnte dann auch in der Rechenzeit aufwändiger sein, da zwischenzeitlich das untere System zum Einsatz kommt. Eventuell bestünde auch die Möglichkeit, bei jeder gesendeten Positionsänderung des übergeordneten Systems eine mögliche Verschiebung des Nullpunkts der Inertialsensoren zu berechnen und automatisch eine Nullpunktkorrektur durchzuführen.

Das vorliegende System bietet die Option, zu jedem beliebigen Zeitpunkt eine Richtungsänderung des Endeffektors vorzugeben. Es ist also bereits die Möglichkeit gegeben, um mit Hilfe eines weiteren externen Systems eine Langzeitstabilisierung durchzuführen.

8.3.2 Lagestabilisierung

In dieser Arbeit wird ausschließlich die Orientierung des Endeffektors eines Manipulators stabilisiert. Es ist aber auch denkbar, dass man neben der Orientierung auch zusätzlich die Position des Manipulators stabilisieren möchte. Diese Aufgabe kann auch mit Hilfe von Inertialsensoren durchgeführt werden. Allerdings sind dann zusätzlich zu den Drehwinkelsensoren auch Beschleunigungssensoren nötig. Die Daten der Beschleunigungssensoren müssen dann zu linearen Geschwindigkeiten umgerechnet werden. Alle Berechnungen müssten in homogenen Koordinaten stattfinden und es müsste das komplette inverse Jacobi-Modell gelöst werden.

A Installation Konfigurationstool (Windows)

Das Konfigurationstool wurde in der Programmiersprache Java geschrieben. Das macht es zwar unter allen gängigen Betriebssystemen verfügbar, jedoch lässt es sich auf manchen Betriebssystemen, wie zum Beispiel Microsoft Windows nicht als eigenständige und ausführbare Datei aufrufen. Hier wird ein wenig Vorarbeit nötig, die im Folgenden beschrieben wird.

Zunächst wird für zum Ausführen von Javaprogrammen die JRE selbst benötigt. Diese kann unter www.java.com/getjava heruntergeladen werden. Nach der Installation der Laufzeitumgebung für das verwendete Betriebssystem können prinzipiell schon Programme gestartet werden. Zum Ausführen des Konfigurationstools sind dennoch zwei Schritte nötig.

Das Konfigurationstool wurde mit Klassen des Standard Widget Toolkit (SWT) entwickelt. Hierbeit handelt es sich um eine Bibliothek von IBM, die bei der Darstellung die nativen grafischen Bedienelemente des Betriebssystems nutzt. Die damit erstellten Programme sehen unter jedem Betriebssystem so aus, als wären sie direkt für das jeweilige System entwickelt worden. Damit das SWT unter Windows genutzt werden kann, muss die Datei swt-win32-3236.dll entweder ins Windows-System-Verzeichnis (in der Regel C:\Windows\System) kopiert werden oder sich im gleichen Verzeichnis befinden, aus dem der Programmaufruf erfolgt.

Als letzten Schritt muss die CLASSPATH-Variable für Java gesetzt werden. Dies ist eine Umgebungsvariable, in der Java nach eventuell nötigen Klassendateien sucht. Um die Variable unter Windows zu editieren, muss unter Start→Systemsteuerung→Leistung und Wartung→System das Fenster mit den Systemeigenschaften geöffnet werden. Im Fenster muss nun der Reiter Erweitert gewählt werden und dort der Knopf Umgebungsvariablen gedrückt werden. Nun öffnet sich ein weiteres Fenster, in dem unter den Systemvariablen die Variable CLASSPATH aufgeführt sein sollte.

Wenn die Variable schon vorhanden ist, so sollte sie ausgewählt und danach auf Bearbeiten geklickt werden. Ansonsten sollte der Knopf Neu gewählt und in dem neu geöffneten Fenster unter Variable der Wert CLASSPATH eingegeben werden. In einem letzten Schritt sollte unter Wert der Variablen durch Semikolon getrennt
%Pfad_zum_Configurator%\Configurator.jar und

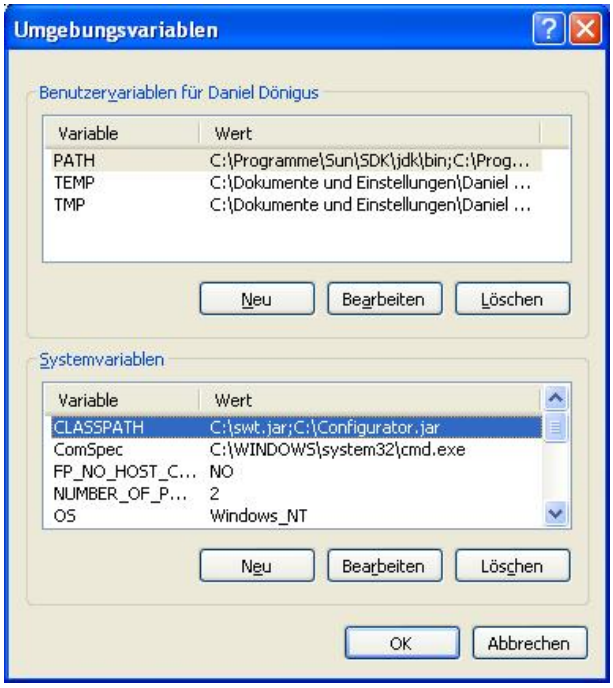

Abb. A.1: Dialog “Umgebungsvariablen”

%Pfad_zu_SWT%\org.eclipse.swt.win32.win32.x86_3.2.2.v3236.jar
eingegeben werden. Bei der Eingabe ist darauf zu achten, dass zwischen den Dateien und in den Dateinamen keine Leerzeichen enthalten sein dürfen.

Abb. A.2: Dialog zur Änderung einer Umgebungsvariablen

B Driftmessungen

Für die Driftmessungen wurde das System auch nach der Initialisierung in einer Ruhelage gehalten und nach bestimmten Zeitabständen von jeweils 30 Sekunden wurde die aktuelle Position der Gyroskope ausgegeben. Da diese jedoch die Orientierung nicht geändert haben, ist der ausgegebene Wert der Drift, der in der jeweiligen Zeitspanne aufgetreten ist. In den hier aufgeführten Tabellen sind jeweils die Positionen in Intervallen von jeweils 60 Sekunden aufgelistet. Zusätzlich wurden jeweils die minimalen, maximalen und Durchschnittswerte berechnet.

Da davon auszugehen ist, dass die Initialisierung einen großen Einfluss auf die Bestimmung des Nullpunkts und damit auch auf den Sensordrift hat, wurden die Messungen mit verschiedenen Einstellungen bei der Initialisierung durchgeführt. Es wurde immer zwischen den letzten drei Werten verglichen, ob die Initialisierung fertig ist. Im einzelnen wurden folgende Einstellungen verändert:

- **Abfragen**
  Anzahl der zeitlichen Samples, die herangezogen werden, um eine Initialisierungsmessung der Gyroskope zu erhalten. Dies entspricht einem Tiefpassfilter und ist nötig, damit das Signalrauschen kompensiert werden kann. Je größer dieser Wert ist, desto mehr Einzelmessungen werden zusammengefasst. Daraus resultiert ein genaueres Messergebnis aber auch eine längere Initialisierungszeit.
- **Max Unterschied**
  Maximaler Unterschied, der zwischen den letzten drei Messergebnissen aufgetreten sein darf. Wenn zwischen dem minimalen und maximalen Wert dieser Messergebnisse maximal dieser Unterschied ergibt, so gilt die Initialisierung als erfolgreich und als Nullpunkt wird das Mittel der letzten drei Messergebnisse genommen.
- **A/D-Abfragen**
  Bei den letzten Messergebnissen wurde direkt beim Auslesen der A/D-Wandler schon die Anzahl der A/D-Abfragen verdoppelt. Prinzipiell handelt es sich dabei um einen Tiefpassfilter direkt bei der Abfrage der A/D-Wandler.

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	9,467	3,203	4,433
Versuch 2	3,398	0,615	3,886
Versuch 3	9,487	9,873	1,328
Versuch 4	3,466	4,941	3,476
Versuch 5	1,469	3,466	3,212
Minimum	1,469	0,615	1,328
Maximum	9,487	9,873	4,433
Mittel	5,457	4,420	3,267

Abfragen: 20, Max.Unterschied: 5,
Zeit: 60 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	19,238	5,864	10,795
Versuch 2	9,433	0,014	9,716
Versuch 3	14,692	16,508	0,019
Versuch 4	6,796	12,602	9,135
Versuch 5	5,571	9,653	3,896
Minimum	5,571	0,014	0,019
Maximum	19,238	16,508	10,795
Mittel	11,146	8,928	6,712

Abfragen: 20, Max.Unterschied: 5,
Zeit: 120 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	28,945	6,533	16,083
Versuch 2	13,012	0,634	14,746
Versuch 3	22,807	24,296	0,927
Versuch 4	11,865	21,777	15,975
Versuch 5	5,864	13,027	5,781
Minimum	5,864	0,634	0,927
Maximum	28,945	24,296	16,083
Mittel	16,499	13,253	10,702

Abfragen: 20, Max.Unterschied: 5,
Zeit: 180 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	39,233	6,953	19,726
Versuch 2	18,295	2,421	22,373
Versuch 3	30,234	32,646	1,445
Versuch 4	18,974	32,626	24,287
Versuch 5	11,484	20,869	5,498
Minimum	11,484	2,421	1,445
Maximum	39,233	32,646	24,287
Mittel	23,644	19,103	14,666

Abfragen: 20, Max.Unterschied: 5,
Zeit: 240 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	48,491	7,114	23,471
Versuch 2	24,985	6,489	28,095
Versuch 3	35,209	40,493	2,421
Versuch 4	21,499	38,715	29,956
Versuch 5	18,452	28,178	5,986
Minimum	18,452	6,489	2,421
Maximum	48,491	40,493	29,956
Mittel	29,727	24,198	17,986

Abfragen: 20, Max.Unterschied: 5,
Zeit: 300 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	2,592	7,597	3,281
Versuch 2	7,275	2,314	7,255
Versuch 3	10,947	11,572	8,515
Versuch 4	3,676	4,365	3,261
Versuch 5	7,182	12,343	7,958
Minimum	2,592	2,314	3,261
Maximum	10,947	12.343	8,515
Mittel	6,334	7,638	6,054

Abfragen: 40, Max.Unterschied: 4,
Zeit: 60 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	7,802	18,872	9,677
Versuch 2	14,257	0,581	13,764
Versuch 3	21,572	21,030	14,790
Versuch 4	13,696	15,766	12,612
Versuch 5	15,405	25,639	13,237
Minimum	7,802	0,581	9,677
Maximum	21,572	25,639	14,790
Mittel	14,546	16,378	12,816

Abfragen: 40, Max.Unterschied: 4,
Zeit: 120 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	13,974	34,179	15,937
Versuch 2	21,679	2,021	19,960
Versuch 3	31,992	29,667	20,439
Versuch 4	19,814	25,507	21,767
Versuch 5	24,248	37,041	19,882
Minimum	13,974	2,021	15,937
Maximum	31,992	37,041	21,767
Mittel	22,341	25,683	19,597

Abfragen: 40, Max.Unterschied: 4,
Zeit: 180 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	18,457	48,671	22,158
Versuch 2	28,129	5,859	26,240
Versuch 3	44,008	39,140	28,291
Versuch 4	30,922	36,914	34,453
Versuch 5	33,608	48,173	26,328
Minimum	18,457	5,859	22,158
Maximum	44,008	48,671	34,453
Mittel	31,025	35,751	27,494

Abfragen: 40, Max.Unterschied: 4,
Zeit: 240 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	24,130	65,083	28,701
Versuch 2	36,616	9,360	34,594
Versuch 3	57,036	47,495	34,370
Versuch 4	41,914	45,747	44,125
Versuch 5	39,433	54,926	28,002
Minimum	24,130	9,360	28,002
Maximum	57,036	65,083	44,125
Mittel	39,826	44,522	33,958

Abfragen: 40, Max.Unterschied: 4,
Zeit: 300 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	6,342	4,042	7,255
Versuch 2	1,918	1,386	0,498
Versuch 3	1,655	2,382	1,386
Versuch 4	6,171	5,107	5,634
Versuch 5	1,474	3,251	2,451
Minimum	1,474	1,386	0,498
Maximum	6,342	5,107	7,255
Mittel	3,512	3,234	3,445

Abfragen: 80, Max.Unterschied: 3,
Zeit: 60 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	12,358	9,204	12,861
Versuch 2	0,844	3,291	2,216
Versuch 3	3,334	5,058	1,870
Versuch 4	12,617	10,922	12,231
Versuch 5	0,952	3,901	3,432
Minimum	0,844	3,291	1,870
Maximum	12,617	10,922	12,8611
Mittel	6,021	6,475	6,522

Abfragen: 80, Max.Unterschied: 3,
Zeit: 120 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	22,739	18,974	22,539
Versuch 2	2,973	5,712	4,765
Versuch 3	4,453	7,050	2,695
Versuch 4	17,861	15,566	16,865
Versuch 5	0,747	5,742	4,570
Minimum	0,747	5,712	2,695
Maximum	22,739	18,974	22,539
Mittel	9,755	10,609	10,287

Abfragen: 80, Max.Unterschied: 3,
Zeit: 180 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	30,415	26,181	30,458
Versuch 2	4,355	7,089	5,087
Versuch 3	4,082	8,749	3,281
Versuch 4	20,136	16,210	16,826
Versuch 5	2,050	9,970	7,656
Minimum	2,050	7,089	3,281
Maximum	30,415	26,181	30,458
Mittel	12,208	13,640	12,662

Abfragen: 80, Max.Unterschied: 3,
Zeit: 240 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	39,804	37,475	39,990
Versuch 2	6,757	11,201	8,642
Versuch 3	4,531	10,244	4,790
Versuch 4	25,214	19,545	19,301
Versuch 5	0,126	11,752	9,467
Minimum	0,126	10,244	4,790
Maximum	39,804	37,475	39,990
Mittel	15,286	18,043	16,438

Abfragen: 80, Max.Unterschied: 3,
Zeit: 300 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	0,937	2,431	0,078
Versuch 2	2,348	2,763	1,230
Versuch 3	1,127	0,878	1,171
Versuch 4	3,825	3,134	5,771
Versuch 5	0,673	1,376	0,664
Minimum	0,673	0,878	0,078
Maximum	3,825	3,134	5,771
Mittel	1,782	2,116	1,783

Abfragen: 160, Max.Unterschied: 2,
Zeit: 60 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	4,287	4,282	2,265
Versuch 2	8,505	6,757	6,025
Versuch 3	1,459	3,769	7,788
Versuch 4	6,380	3,753	9,355
Versuch 5	0,546	6,650	0,517
Minimum	0,546	3,753	0,517
Maximum	8,505	6,757	9,355
Mittel	4,235	5,042	5,190

Abfragen: 160, Max.Unterschied: 2,
Zeit: 120 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	6,259	5,410	2,792
Versuch 2	6,625	4,951	3,085
Versuch 3	2,568	10,634	18,085
Versuch 4	6,950	3,460	9,521
Versuch 5	1,230	8,925	0,029
Minimum	1,230	3,460	0,029
Maximum	6,950	10,634	18,085
Mittel	4,726	6,676	6,702

Abfragen: 160, Max.Unterschied: 2,
Zeit: 180 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	11,450	3,056	4,765
Versuch 2	5,922	1,503	2,480
Versuch 3	7,509	19,208	28,115
Versuch 4	9,912	4,276	12,978
Versuch 5	4,223	18,134	5,468
Minimum	4,223	1,503	2,480
Maximum	11,450	19,208	28,115
Mittel	7,803	9,235	10,761

Abfragen: 160, Max.Unterschied: 2,
Zeit: 240 sec

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	16,147	2,172	7,324
Versuch 2	3,095	2,587	1,718
Versuch 3	12,382	26,728	35,522
Versuch 4	8,218	4,537	15,322
Versuch 5	7,031	25,371	9,951
Minimum	3,095	2,587	1,718
Maximum	16,147	26,728	35,522
Mittel	9,375	12,279	13,967

Abfragen: 160, Max.Unterschied: 2,
Zeit: 300 sec

C Messungen der Rückstellgenauigkeit

Die folgenden zwei Tabellen enthalten die Messwerte einer weiteren Driftmessung der Gyroskope. Während der Versuchsdauer von einer Minute wurden die Gyroskope in Ruhelage gehalten und nach dieser Zeit die berechnete Position der Gyroskope ausgegeben, bei der lediglich die Sensorsignale der Gyroskope aufaddiert wurden. In der linken Tabelle wurden nur die Sensorsignale aufaddiert, während in der rechten Tabelle zusätzlich die Servomotoren angesteuert wurden. Die Motoren wurden so angesteuert, dass eine kontinuierliche Trajektorie mit zwei Wegpunkten abgefahren wurde. Alle $10ms$ wurden die Motoren abgefragt, ob sie am gewünschten Wegpunkt angekommen waren. Sobald ein Wegpunkt erreicht wurde, wurde der nächste Wegpunkt als Sollposition an den Aktuator geschickt.

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	0,766	0,009	-1,230
Versuch 2	1,166	-1,699	1,982
Versuch 3	1,538	0,498	0,976
Versuch 4	0,375	0,390	0,722
Versuch 5	-1,308	0,654	2,460
Versuch 6	0,966	1,259	1,542
Versuch 7	0,322	1,523	0,654
Versuch 8	-1,054	0,458	0,585
Versuch 9	0,756	1,708	1,396
Versuch 10	-1,586	-1,728	2,939
Minimum	-1,586	-1,728	-1,230
Maximum	1,538	1,708	2,939
Mittel	0,194	0,307	1,203
Abs.Mittel	0.984	0,993	1,449

Nullpunktmessung ohne Ansteuerung der Servomotoren

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	-3,891	1,884	0,849
Versuch 2	-1,791	-1,777	0,156
Versuch 3	-7,177	5,537	-3,779
Versuch 4	-1,923	0,058	-3,535
Versuch 5	-3,325	2,050	-5,078
Versuch 6	-2,714	1,640	-5,156
Versuch 7	-3,613	3,457	1,142
Versuch 8	-2,929	5,361	0,097
Versuch 9	-1,684	1,455	-2,119
Versuch 10	-6,254	0,048	0,478
Minimum	-7,177	-1,777	-5,156
Maximum	-1,684	5,537	1,142
Mittel	-3,530	1,971	-1,695
Abs.Mittel	3,530	2,327	2,239

Nullpunktmessung mit laufenden Servomotoren

Für die folgenden Messergebnisse wurden die Gyroskope wieder am Endeffektor des Manipulators befestigt. Während der Tests wurde von den Testservos eine Trajektorie abgefahren. Die Wegpunkte der Trajektorie sind in den Tabellenbezeichnungen angegeben. Hierbei bedeutet ein Tripel $\theta_{tz}/\theta_{ty}/\theta_{tx}$ die Winkelstellung der drei Testservos bei einem Wegpunkt.

Da bei den Messungen unerklärliche Abweichungen bei der Rückstellgenauigkeit aufgetreten sind, wurden die Tests jeweils mit aktivierter und deaktivierter Stabilisierung durchgeführt. Die Trajektorie wurde jeweils über den Zeitraum von 15 Sekunden abgefahren. Nach diesen 15 Sekunden wurde die Trajektorie noch zum Ursprungswegpunkt 0/0/0 zu Ende gefahren. Danach wurde die Position der Gyroskope im Falle der deaktivierten Stabilisierung und die Position der Servos bei der aktivierten Stabilisierung gemessen. Bei einer deaktivierten Stabilisierung müsste die Position der Gyroskope nach Beenden des Test wieder jeweils den Ursprungszustand von jeweils 0° anzeigen.

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	-1,054	1,191	1,220
Versuch 2	-1,821	3,085	-1,083
Versuch 3	-1,396	2,236	0,776
Versuch 4	-1,147	1,489	0,981
Versuch 5	-1,157	1,918	-1,762
Versuch 6	-1,362	1,171	0,439
Versuch 7	-3,159	2,470	0,527
Versuch 8	-2,861	4,394	-2,724
Versuch 9	-1,494	0,634	0,400
Versuch 10	0,595	1,669	-1,416
Minimum	-2,861	0,634	-2,724
Maximum	0,595	4,394	1,220
Mittel	-1,486	2,026	-0,266
Abs.Mittel	1,605	2,026	1,133

Keine Stabilisierung,
Weg: 0/0/0 → 30/0/0 → 0/0/0

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	3,228	3,347	0,410
Versuch 2	1,846	3,924	0,560
Versuch 3	1,792	1,840	-1,064
Versuch 4	2,740	2,849	0,365
Versuch 5	2,918	3,122	-1,163
Versuch 6	1,931	2,620	-2,768
Versuch 7	1,814	1,123	0,363
Versuch 8	1,672	3,413	0,628
Versuch 9	2,451	2,815	-1,386
Versuch 10	1,691	1,925	0,568
Minimum	1,672	1,123	-2,768
Maximum	3,228	3,924	0,628
Mittel	2,208	2,698	-0,349
Abs.Mittel	2,208	2,698	0,928

Mit Stabilisierung,
Weg: 0/0/0 → 30/0/0 → 0/0/0

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	-1,152	1,357	-2,080
Versuch 2	0,102	1,240	-1,875
Versuch 3	-1,982	4,565	-5,795
Versuch 4	-1,914	2,070	-2,631
Versuch 5	-1,098	1,933	-1,230
Versuch 6	0,747	1,899	-2,226
Versuch 7	0,292	1,874	-1,171
Versuch 8	-1,958	2,827	-4,008
Versuch 9	-2,812	3,413	-3,818
Versuch 10	-2,255	3,403	-3,945
Minimum	-2,812	1,240	-5,795
Maximum	0,747	4,565	-1,171
Mittel	-1,203	2,458	-2,878
Abs.Mittel	1,431	2,458	2,878

Keine Stabilisierung,
Weg: 0/0/0 → 0/30/0 → 0/0/0

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	2,597	1,078	0,033
Versuch 2	3,576	0,445	-1,965
Versuch 3	1,943	0,100	0,515
Versuch 4	1,545	-1,350	0,272
Versuch 5	3,891	0,874	0,840
Versuch 6	2,678	1,241	0,584
Versuch 7	3,125	0,703	0,258
Versuch 8	3,190	0,447	0,543
Versuch 9	2,231	2,996	-2,165
Versuch 10	1,724	0,642	0,547
Minimum	1,545	-1,350	-2,165
Maximum	3,891	2,996	0,840
Mittel	2,650	0,718	-0,054
Abs.Mittel	2,650	0,988	0,772

Mit Stabilisierung,
Weg: 0/0/0 → 0/30/0 → 0/0/0

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	-1,450	0,788	-1,367
Versuch 2	0,512	0,161	-1,621
Versuch 3	0,029	1,381	0,258
Versuch 4	-1,328	0,317	-1,513
Versuch 5	-3,027	3,544	-3,525
Versuch 6	-1,220	1,210	-1,098
Versuch 7	0,615	0,610	-1,411
Versuch 8	0,502	1,083	0,258
Versuch 9	-2,343	2,050	-2,509
Versuch 10	-1,249	2,084	0,502
Minimum	-3,027	0,161	-3,525
Maximum	0,615	3,511	0,502
Mittel	-0,896	1,323	-1,203
Abs.Mittel	1,228	1,323	1,406

Keine Stabilisierung,
Weg: 0/0/0 → 30/30/0 → 0/0/0

[°]	Gyro 1	Gyro 2	Gyro 3
Versuch 1	0,892	-5,330	2,005
Versuch 2	1,956	-2,669	0,742
Versuch 3	0,770	-2,063	2,035
Versuch 4	0,010	-3,500	1,478
Versuch 5	0,905	-2,500	2,137
Versuch 6	1,683	-4,779	3,814
Versuch 7	2,974	-4,316	0,465
Versuch 8	1,198	-1,955	2,160
Versuch 9	0,722	-4,159	3,178
Versuch 10	1,602	-4,964	2,630
Minimum	0,010	-5,330	0,465
Maximum	2,974	-1,955	3,814
Mittel	1,271	-3,624	2,064
Abs.Mittel	1,271	3,624	2,064

Mit Stabilisierung,
Weg: 0/0/0 → 30/30/0 → 0/0/0

D Messungen während der Stabilisierung

Die Tabellen der folgenden Seiten zeigen Messungen, die während der Stabilisierung durchgefürt wurden. Hierbei wurde durch die Testservos eine Testtrajektorie abgefahren. Die Trajektorie ist in der Tabellenbezeichnung angegeben. Die Tripel $\theta_{tz}/\theta_{ty}/\theta_{tx}$ bedeuten hierbei die Positionen der Testservos bei einem Wegpunkt.

Die Trajektorie wurde einmal abgefahren und in vorgegebenen Zeitabständen von jeweils $180ms$ wurden die Positionen der Test- und Stabilisierungsservos gemessen. Aus diesen Positionen wurde, wie in Kapitel 7.4 beschrieben, eine Rotationsmatrix aufgestellt. Aus dieser Matrix wurde der Drehwinkel, wie in Formel 3.15 angegeben, berechnet. Dieser Winkel geht als Orientierungsfehler für jeden Messschritt ein.

[°]	Servo TZ	ServoTY	Servo TX	Servo SZ	Servo SY	Servo SX	Gesamtfehler
0,00s	-0,146	-1,026	0,733	-0,146	-0,146	0,146	—
0,18s	0,146	-1,026	0,733	-0,146	-0,146	0,146	0,306
0,36s	4,838	-1,024	0,733	-1,026	-0,146	0,146	4,119
0,54s	8,944	-1,026	0,733	-7,184	-0,146	0,146	2,074
0,72s	12,756	-1,026	0,733	-10,997	-0,146	-0,439	2,102
0,90s	17,155	-1,026	0,733	-14,809	-0,146	-0,733	2,726
1,08s	21,260	-1,026	0,733	-18,914	-0,146	-0,733	2,722
1,26s	24,780	-1,026	0,733	-23,313	0,146	-1,319	2,192
1,44s	29,178	-1,026	0,733	-26,832	0,439	-1,319	3,042
1,62s	29,178	-0,733	0,733	-30,351	0,439	-1,026	1,837
1,80s	24,780	-1,026	0,733	-29,472	0,439	-1,906	4,812
1,98s	20,674	-1,026	0,733	-24,780	0,439	-1,906	4,283
2,16s	16,568	-1,026	0,733	-20,967	0,439	-1,906	4,549
2,34s	11,876	-1,026	0,733	-16,862	0,439	-1,906	5,093
2,52s	7,771	-1,026	0,733	-12,756	0,439	-1,319	4,914
2,70s	3,958	-1,026	0,733	-8,357	0,439	-1,026	4,285
2,88s	0,146	-1,026	0,733	-4,252	0,439	-0,439	3,887
3,06s	0,146	-1,026	0,733	-1,026	0,439	-0,439	1,005
Minimum	—	—	—	—	—	—	0,306
Maximum	—	—	—	—	—	—	5,093
Mittel	—	—	—	—	—	—	3,173

Rückstellgeschwindigkeitstest, Weg: 0/0/0 → 30/0/0 → 0/0/0

[°]	Servo TZ	ServoTY	Servo TX	Servo SZ	Servo SY	Servo SX	Gesamtfehler
0,00s	-0,146	-1,026	0,733	-0,146	-0,146	0,146	—
0,18s	0,146	-0,146	0,733	-0,146	-0,146	0,146	0,931
0,36s	0,146	3,958	0,733	-0,146	-1,026	0,146	4,114
0,54s	-0,146	8,357	0,733	-0,439	-6,304	0,146	3,248
0,72s	0,146	12,170	0,733	-0,733	-11,583	0,146	1,821
0,90s	-0,146	16,568	0,733	-1,026	-14,809	0,146	3,127
1,08s	-0,146	20,674	0,733	-1,319	-18,621	0,439	3,500
1,26s	-0,146	25,366	0,733	-1,319	-23,607	0,733	3,230
1,44s	0,146	29,178	0,733	-1,612	-28,005	0,733	2,709
1,62s	-0,146	29,178	0,733	-2,492	-30,351	1,026	2,407
1,80s	-0,146	24,486	0,733	-2,492	-30,058	1,319	4,993
1,98s	0,146	19,794	0,733	-2,492	-25,073	1,319	4,616
2,16s	-0,146	16,568	0,733	-2,199	-20,381	1,319	3,426
2,34s	0,146	12,170	0,733	-1,319	-15,689	1,319	2,680
2,52s	-0,146	8,064	0,733	-1,026	-12,756	1,319	3,777
2,70s	0,146	3,665	0,733	-0,733	-8,651	1,319	3,982
2,88s	0,146	-0,439	0,733	-0,733	-3,372	1,026	2,791
3,06s	0,146	-0,439	0,733	-0,439	-0,146	0,733	0,592
Minimum	—	—	—	—	—	—	0,592
Maximum	—	—	—	—	—	—	4,993
Mittel	—	—	—	—	—	—	3.056

Rückstellgeschwindigkeitstest, Weg: 0/0/0 → 0/30/0 → 0/0/0

[°]	Servo TZ	ServoTY	Servo TX	Servo SZ	Servo SY	Servo SX	Gesamtfehler
0,00s	-0,146	-0,733	-0,146	-0,146	-0,439	-0,146	—
0,18s	0,146	-0,439	-0,146	-0,146	-0,439	-0,146	0,412
0,36s	4,838	3,958	-0,146	-1,026	-1,026	-0,146	5,801
0,54s	8,944	8,357	-0,146	-7,184	-6,011	-0,146	4,163
0,72s	12,756	12,463	-0,146	-12,170	-10,410	-0,146	4,671
0,90s	17,155	16,568	-0,146	-16,568	-13,929	1,026	4,888
1,08s	21,260	20,967	-0,146	-20,674	-18,035	3,372	5,070
1,26s	25,073	25,366	-0,146	-25,659	-21,554	5,131	6,253
1,44s	29,178	29,178	-0,146	-30,351	-25,366	8,064	6,619
1,62s	29,178	29,178	-0,146	-33,870	-26,832	11,290	4,395
1,80s	24,780	24,486	-0,146	-32,991	-25,366	14,222	5,590
1,98s	20,674	20,674	-0,146	-27,126	-20,967	13,636	6,209
2,16s	16,568	16,568	-0,146	-21,847	-18,621	8,064	4,725
2,34s	12,170	12,170	-0,146	-17,155	-15,102	5,718	5,151
2,52s	7,771	8,064	-0,146	-13,343	-12,170	3,372	6,075
2,70s	3,665	3,665	-0,146	-8,651	-8,357	1,906	5,970
2,88s	0,146	-0,146	-0,146	-3,958	-3,372	0,439	4,253
3,06s	-0,146	-0,733	-0,146	-0,733	-0,733	0,146	0,510
Minimum	—	—	—	—	—	—	0,412
Maximum	—	—	—	—	—	—	6,619
Mittel	—	—	—	—	—	—	4,750

Rückstellgeschwindigkeitstest, Weg: 0/0/0 → 30/30/0 → 0/0/0

[°]	Servo TZ	ServoTY	Servo TX	Servo SZ	Servo SY	Servo SX	Gesamtfehler
0,00s	0,439	-1,026	0,439	-0,439	-0,146	-0,439	—
0,09s	0,146	-1,026	0,439	-0,439	-0,146	-0,439	0,296
0,18s	2,492	-0,733	0,439	-0,439	-0,146	-0,439	2,073
0,27s	4,838	-1,026	0,439	-1,026	-0,146	-0,439	3,810
0,36s	6,598	-1,026	0,439	-3,665	-0,146	-0,439	2,933
0,45s	8,944	-1,026	0,439	-6,011	-0,146	-0,439	2,935
0,54s	10,997	-1,026	0,439	-7,478	-0,146	-0,439	3,522
0,63s	12,756	-0,439	0,439	-10,410	0,146	-0,733	2,545
0,72s	15,107	-1,026	0,439	-11,290	0,146	-0,733	3,837
0,81s	17,155	-1,026	0,439	-13,929	0,146	-0,733	3,254
0,90s	18,914	-1,026	0,439	-15,102	-0,146	-0,733	3,810
0,99s	21,554	-1,026	0,439	-17,741	0,146	-1,026	3,848
1,08s	23,020	-1,026	0,439	-19,794	0,146	-1,026	3,271
1,17s	25,073	-1,026	0,439	-21,847	0,146	-1,026	3,273
1,26s	27,419	-1,026	0,439	-23,607	0,439	-1,026	3,912
1,35s	29,178	-1,026	0,439	-25,366	0,439	-1,319	3,938
1,44s	29,472	-1,026	0,439	-28,005	0,439	-1,319	1,783
1,53s	29,472	-1,026	0,439	-29,178	0,439	-1,319	1,067
1,62s	29,472	-1,026	0,439	-29,178	0,439	-1,319	1,067
1,71s	29,472	-1,026	0,439	-29,472	0,439	-1,319	1,029
Minimum	—	—	—	—	—	—	0,296
Maximum	—	—	—	—	—	—	3,938
Mittel	—	—	—	—	—	—	2,748

Rückstellgeschwindigkeitstest, Weg: 0/0/0 → 30/0/0

[°]	θ_{tz}	θ_{ty}	θ_{tx}	θ_{sz}	θ_{sy}	θ_{sx}	$\theta_{t,ges}$	$\theta_{s,ges}$	Ges.Fehler
0,00s	-0,146	-1,026	0,733	-0,439	0,146	-0,146	0	0	—
0,18s	2,492	-1,026	3,665	-0,439	0,146	-0,146	3,902	0,002	3,892
0,36s	6,598	-1,026	7,478	-4,545	0,146	2,785	9,442	5,051	4,548
0,54s	10,997	-1,026	12,170	-8,651	1,026	6,598	15,804	10,705	5,296
0,72s	15,102	-1,026	15,089	-12,756	2,199	10,410	20,717	16,500	4,369
0,90s	18,914	-1,026	19,794	-16,568	4,252	13,636	26,643	21,968	5,009
1,08s	23,313	-1,026	23,900	-20,087	7,184	17,741	32,551	28,222	4,460
1,26s	27,419	-1,026	28,299	-23,313	9,824	21,260	38,434	33,940	4,713
1,44s	28,885	-1,026	30,645	-26,539	12,170	23,900	41,067	39,043	3,063
1,62s	26,832	-1,026	28,299	-27,419	14,516	24,193	38,031	40,952	3,461
1,80s	22,727	-1,026	23,900	-25,659	15,102	20,967	32,144	37,868	7,193
1,98s	18,621	-1,026	19,794	-21,554	11,876	18,035	26,440	31,435	6,470
2,16s	14,516	-1,026	15,689	-18,328	8,944	15,395	20,719	26,077	6,375
2,34s	10,703	-1,026	12,170	-14,809	6,011	11,583	15,603	19,889	5,281
2,52s	6,304	-1,026	7,184	-12,170	3,665	8,357	9,030	15,121	6,470
2,70s	2,199	-1,026	3,079	-8,064	3,079	4,545	3,280	9,525	6,514
2,88s	-0,146	-1,026	0,733	-3,665	2,199	1,319	0,014	4,119	4,133
3,06s	-0,146	-1,026	0,733	-1,022	1,026	0,439	0,014	1,213	1,222
Min	—	—	—	—	—	—	—	—	1,222
Max	—	—	—	—	—	—	—	—	7,193
Mittel	—	—	—	—	—	—	—	—	4,851

Rückstellgeschwindigkeitstest, Weg: 0/0/0 → 30/0/30 → 0/0/0

E Nötige Funktionsdeklarationen für die Schnittstellen

Name	Rückgabe	Übergabe
InitController	void	void
readADChannel	int	unsigned char ucChannel unsigned short usFilterSamples
sendStringServo	void	unsigned char *ucpMessage
readCharServo	unsigned char	void
sendStringHost	void	unsigned char *ucpMessage
readCharHost	unsigned char	void
interruptServos	void	void
interruptTimer	void	void

Tab. E.1: Funktionen des Mikrocontroller-Moduls

Beschreibung der Funktionen aus Tabelle E.1:

- `initController`
  Initialisierungsfunktion für alle Teile des Mikrocontrollers. Hier sollten die Interrupts, I/O-Ports, Timer, Serielle Schnittstellen oder der A/D-Wandler initialisiert werden. Dies kann auch wieder durch Unterfunktionen geschehen, die von dieser Funktion aufgerufen werden.
- `readADChannel`
  Liest einen Kanal des A/D-Wandler aus und liefert dessen Wert zurück. Übergen wird der gewünschte Kanal, sowie die Anzahl der Abfragen für eine erste Tiefpassfilterung.
- `sendStringServo`
  Sendet einen Kommandostring auf die UART-Schnittstelle zu den Servos.
- `readCharServo`
  Liest ein Zeichen auf der UART-Schnittstelle von den Servos. In dieser Funktion wird nicht explizit auf den Empfang eines Zeichens gewartet.
- `sendStringHost`
  Sendet einen String an den Hostrechner.

- `readCharHost`
  Wartet auf den Empfang eines Zeichens vom Host-PC und gibt dieses dann zurück.
- `interruptServos`
  Interruptfunktion, die aufgerufen wird, wenn ein ein Zeichen von den Servos wurde. Hier sollte dann auch direkt die komplette Statusmeldung fertig ausgelesen werden.
- `interruptTimer`
  Interruptfunktion, die nach einer bestimmten Zeitspanne aufgerufen wird. Sie dient hier lediglich dazu die Timerfunktion im Hauptprogramm aufzurufen.

Name	Rückgabe	Übergabe
isServoMoving	bool	unsigned char ucID
degreePerSecToServoSpeed	unsigned short	float fSpeed
degreeToServoPosition	unsigned short	float fPosition
setServoPos	void	unsigned short usPosition unsigned char ucID unsigned short usSpeed
readServoData	void	unsigned char ucID unsigned char ucAdress unsigned char ucLength

Tab. E.2: Funktionen des Servo-Moduls

Beschreibung der Funktionen aus Tabelle E.2:

- `isServoMoving`
  Gibt binären Zustand zurück, ob sich das Servo mit der gegebenen ID sich bewegt oder nicht.
- `degreePerSecToServonSpeed`
  Rechnet die Geschwindigkeit von $[°/s]$ in die nötige Geschwindigkeit für das Servo um.
- `degreeToServoPosition`
  Rechnet die Position von $[°]$ in die Servoposition um.
- `setServoPos`
  Lässt das Servo mit der übergebenen ID an eine neue Position fahren. Neben der ID wird auch die gewünschte Sollposition und die Geschwindigkeit übergeben, mit der das Servo an diese Position fahren soll.
- `readServoDatat`
  Liest einen Datenbereich aus der Zustandstabelle des Servos aus. Es wird die ID des Servos, sowie die Anfangsposition der Daten und die Länge in Byte der auszulesenden Daten übergeben.

Name	Rückgabe	Übergabe
startInit	void	unsigned char ucGyroID
endInit	void	unsigned char ucGyroID
readGyroSpeed	void	unsigned char ucGyroID unsigned char ucChannel
returnSpeed	int	unsigned char ucGyroID

Tab. E.3: Funktionen des Gyroskop-Moduls

Beschreibung der Funktionen aus Tabelle E.3:

- `startInit`
  Startet die Initialisierung. Hier sollten die aktuellen Nullpunkte und die Variable mit den aufsummierten Messwerten gelöscht werden. Übergeben wird die interne ID des Gyroskops, für das die Initialisierung gestartet werden soll.
- `endInit`
  Beendet die Initialisierung und berechnet aus den aufsummierten Messwerten den Nullpunkt des Gyroskops. Übergeben wird die interne ID des Gyroskops, für das die Initialisierung beendet werden soll.
- `readGyroSpeed`
  Liest die den A/D-Wandler mit der übergebenen Kanalnummer aus und addiert diesen Wert zu den Messwerten für das Gyroskop mit der ebenfalls übergebenen ID.
- `returnSpeed`
  Gibt die aufsummierten Messwerte für das Gyroskop mit der übergebenen ID aus und setzt diese zurück.

F Skizzen und Fotos des Stabilisierungssystems

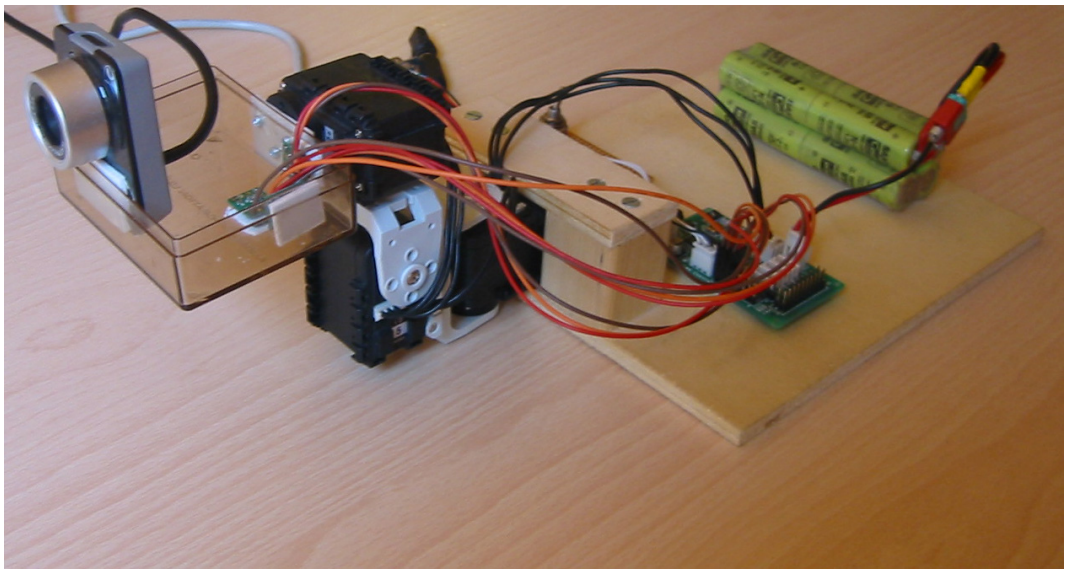

Abb. F.1: Foto des Testsystems

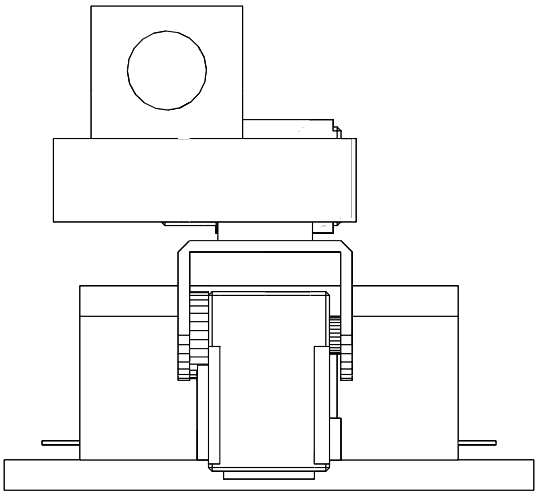

Abb. F.2: Frontansicht des Testsystems

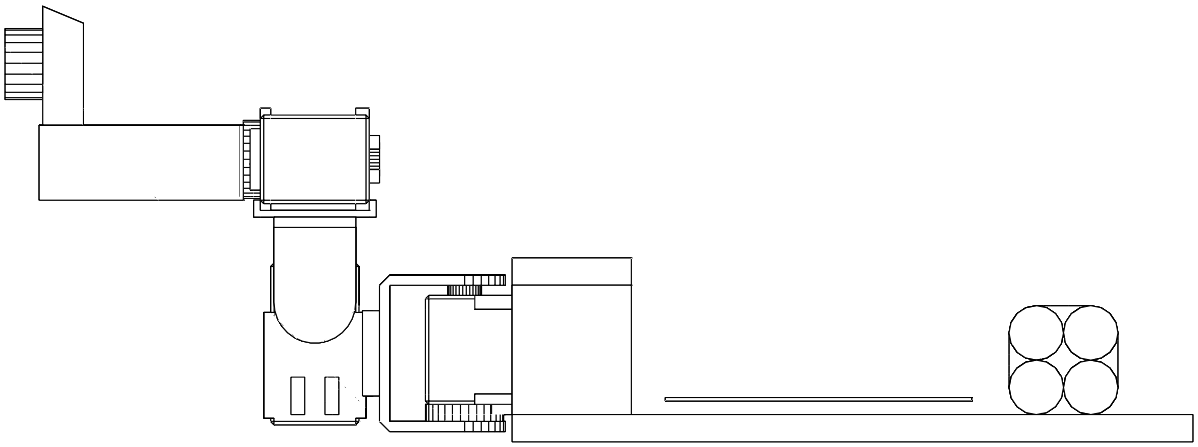

Abb. F.3: Seitenansicht des Testsystems

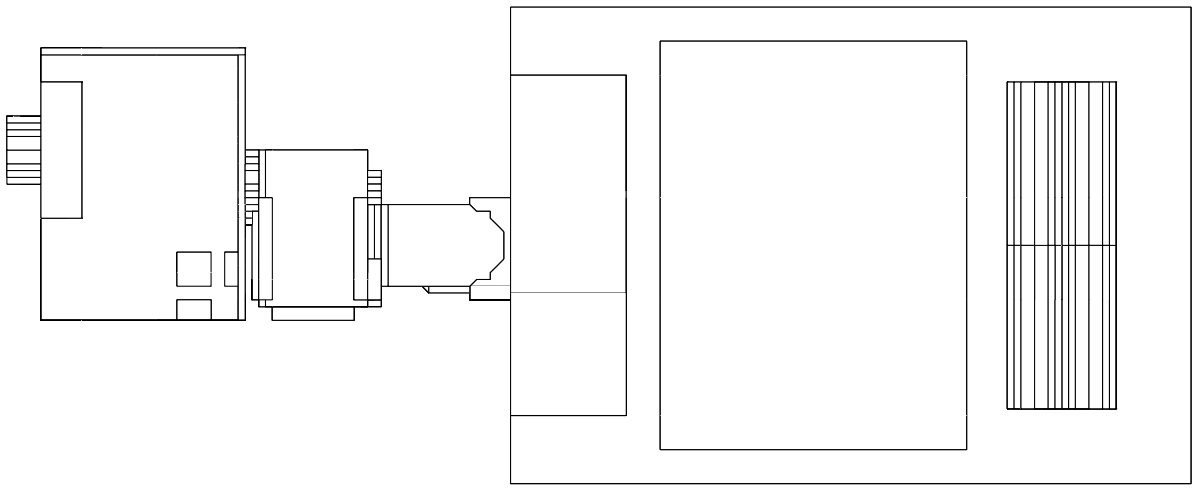

Abb. F.4: Draufsicht des Testsystems

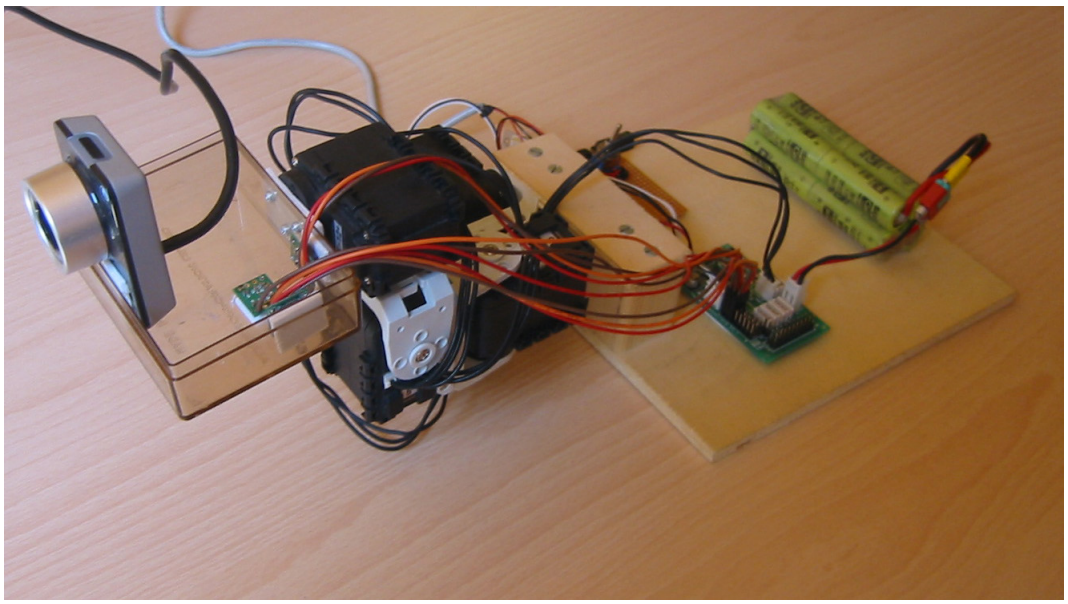

Abb. F.5: Foto des erweiterten Testsystems

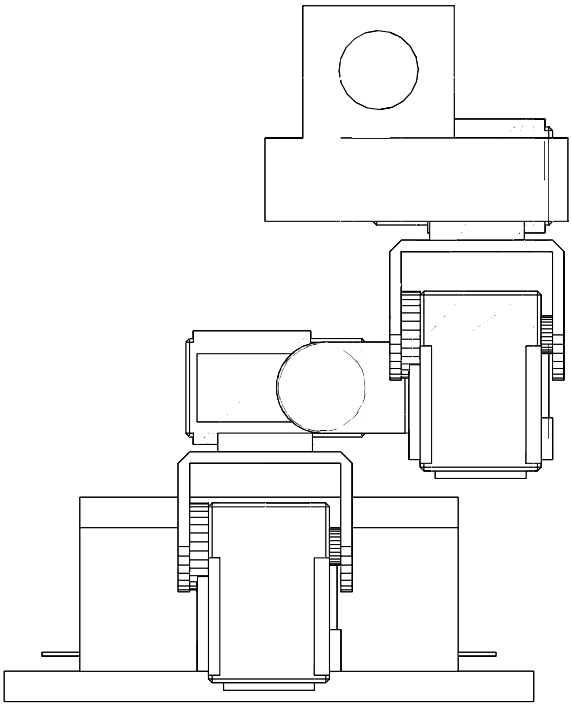

Abb. F.6: Seitenansicht des erweiterten Testsystems

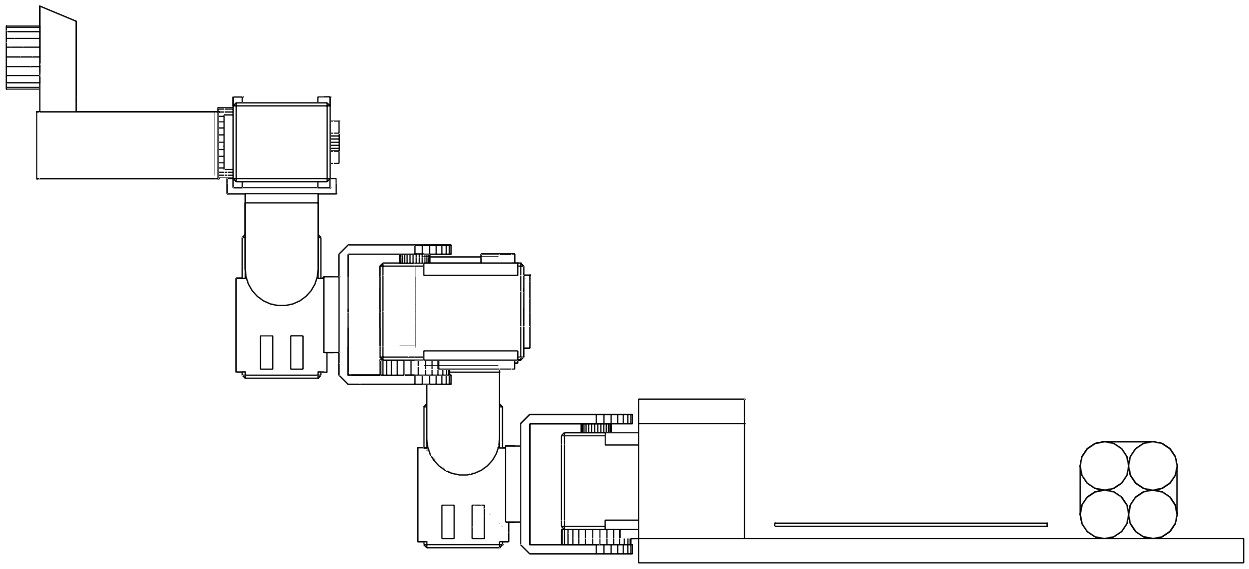

Abb. F.7: Seitenansicht des erweiterten Testsystems

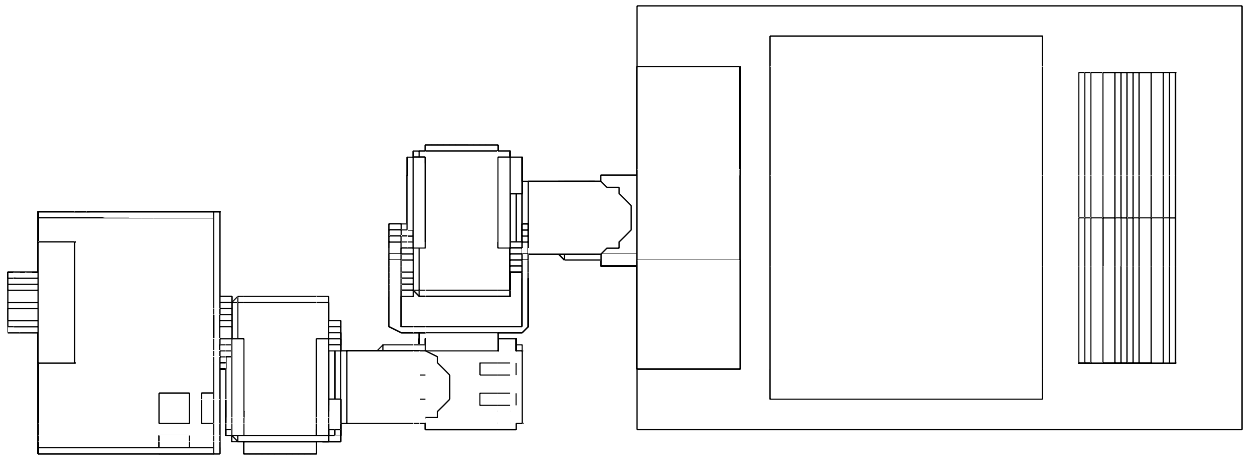

Abb. F.8: Draufsicht des erweiterten Testsystems

Abkürzungsverzeichnis

A/D-Wandler

Analog-Digital-Wandler 4, 28, 31, 35, 36, 49–51, 53, 54, 56, 59, 61, 73, 77–79, 90, 95, 111, 113

ANSI

American National Standards Institute 61

CCD-Sensor

Charge-Coupled-Device-Sensor 11

DSP

Digital Signal Processor 12

FIFO

First In - First Out 32

GKmM

Graduiertenkolleg Cooperative , Adaptive and Responsive Monitoring in Mixed Mode Environments 7

GPS

Global Positioning System 13

ID

Identifikationsnummer 49, 52, 66, 67, 69

IMU

Inertial Measuerement Unit 8, 29

JRE

Java Runtime Environment 43, 93

μC

Mikrocontroller 1, 35, 48, 51

MEMS

mikro-elektro-mechanische Systeme 27, 28, 30

PC

Personal Computer 2, 8, 32, 50–52, 81

PDA

Personal Digital Assistant 61

RS-232

Radio Sector 232 32

SIM

Fachgebiet Simulation, Systemoptimierug und Robotik 1, 2, 4, 7

SWT

Standard Widget Toolkit 93

UART

Universal Asynchronous Receiver Transmitter 32, 49, 50, 52

VOR

Vestibuokulärer Reflex 9

WLAN

Wireless Local Area Network 9

XML

Extensible Markup Language 67, 70

Glossar

Aktor

Ein Aktor wandelt im allgemeinen eine Eingangsgröße in eine andersartige Ausgangsgröße um. In der Vorliegenden Arbeit wird von den Aktoren elektrische Energie in eine Drehbewegung umgewandelt. Sie dienen hier auch zusätzlich als Drehgelenke. (siehe auch Servo) 2

Drift

Unerwünschte Verschiebung des Nullpunkts eines Sensors. Dies kann auf Dauer zu Fehlern in Langzeitmessungen führen. 3

Duplex

Bezeichnet das gleichzeitige Senden und Empfangen zweier Systeme über ein Übertragungsmedium. Bei einer Halbduplex- oder Simplexkommunikation kann nur ein System senden, während das andere System die Daten empfängt. 32

Externer Sensor

Sensor, der für ein System Merkmale aus dessen Umwelt misst. 3

Gyro

Umgangssprache für Gyroskop (siehe Gyroskop) 35

Gyroskop

Sensor zum Messen von Drehwinkelgeschwindigkeiten um eine Achse. Mechanische Gyroskope werden mit ihrer Schwungmasse teilweise auch direkt als Stabilisatoren eingesetzt. 2

Host-Rechner

Ein weiterer Rechner der seine Dienste über eine Datenverbindung zur Verfügung stellt.. 3

Inertialsensoren

Sensoren, die aufgrund von Trägheiten zum Beispiel Drehraten und lineare Beschleunigungen ohne Bezug zur äußeren Umwelt messen. 2

kinematische Kette

System aus starren gliedern, die durch Gelenke verbunden sind. Die Gelenke können in der Regel durch Aktoren bewegt werden. 3

kinematisches Modell

Aufbau einer kinematischen Kette. Anordnung der einzelnen Glieder und Gelenke.. 3

Mikrocontroller

Ein Chip, der Prozessor und Peripheriefunktionen vereint. Teilweise befinden sich zusätzlich auf dem Chip der Arbeits- und Programmspeicher 2

Modul

Baustein, der zusammen mit anderen Elementen ein Gesamtsystem bildet. 3

Quadrocopter

Eine Hubschrauberart, die über vier Rotoren verfügt, die senkrecht nach unten ausgerichtet sind. Mit Hilfe dieser vier Rotoren, wovon jeweils zwei in unterschiedliche Richtungen laufen, wird zum einen der Auftrieb gewährleistet aber auch die Steuerungsfunktionen durchgeführt. 7

Redundanz

Überschneidung von Informationen. Im Falle der Messung von überschneidenden Informationen mit verschiedenen Sensoren ist dies hier ein unerwünschter Zustand, der eine Nachberechnung notwendig macht. 40

Servo

Servos sind komplette Einheiten aus Ansteuerung und Antrieb. Die hier benutzten Servos können verschiedene Winkelstellungen anfahren und diese Position halten. (siehe auch Aktor) 3

Tiefpassfilter

Filter, der bei einem Signal die hohen Frequenzen herausfiltert. 36

Zeilensprungverfahren

Verfahren zur Aufnahme und Wiedergabe von Kamerabildern mit einer möglichst niedrigen Bildwiederholfrequenz. Es werden in den einzelnen Wiedergabebildern abwechselnd nur die geraden beziehungsweise die ungeraden Zeilen aktualisiert. 2

Literaturverzeichnis

[1] Dana H. Ballard. Animate vision. *Artif. Intell.*, 48(1):57–86, 1991.

[2] E. Grosso and D.H. Ballard. Head-centered orientation strategies in animate vision. *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 395–402, 11-14 May 1993.

[3] P. Wagner; W. Gunthner; H. Ulbrich. A motion device for a stabilized vehicle camera system with control parameter optimization. *Industrial Electronics, 2006 IEEE International Symposium on*, 4:3038–3043, 9-13 July 2006.

[4] B. Marsh, C. Brown, T. LeBlanc, M. Scott, T. Becker, C. Quiroz, P. Das, and J. Karlsson. The rochester checkers player: multimodel parallel programming for animate vision. *Computer*, 25(2):12–19, Feb 1992.

[5] A.J. Shelley and N.L. Seed. Animate vision demonstrator utilising reconfigurable system designs. *Image Processing and its Applications, 1995., Fifth International Conference on*, pages 500–504, 4-6 Jul 1995.

[6] C. Breazeal, A. Edsinger, P. Fitzpatrick, B. Scassellati, and P. Varchavskaia. Social constraints on animate vision. *Intelligent Systems and Their Applications, IEEE [see also IEEE Intelligent Systems]*, 15(4):32–37, Jul/Aug 2000.

[7] Jorge Lobo and Jorge Dias. Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(12):1597–1608, 2003.

[8] E.R. Kandel; J.H. Schwartz; T.M. Jessell. *Principles of Neural Science*. Elsevier, 1981.

[9] K.J. Quinn; N. Schmajuk; J.F. Baker; B.W. Peterson. Modeling the three neuron vestibulo-ocular reflex arc: contribution to eye movement computation. *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 699–704 vol.2, 17-21 Jun 1990.

[10] R. Wagner; H.L. Galiana. Unifying vestibulo-ocular reflexes. *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, 1:849–852 vol.1, 2001.

[11] F. Patane; C. Laschi; H. Miwa; E. Guglielmelli; P. Dario; A. Takanishi. Design and development of a biologically-inspired artificial vestibular system for robot heads. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2:1317–1322 vol.2, 28 Sept.-2 Oct. 2004.

[12] Jian Peng; A. Peters; Xinyu Ao; A. Srikaew. Grasping a waving object for a humanoid robot using a biologically-inspired active vision system. *Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on*, pages 115–120, 31 Oct.-2 Nov. 2003.

[13] B. Schweber. Image stabilization shows diversity of engineering approaches. *Electronics Design, Strategy, News*, pages 32–37, Oct 2000.

[14] Lenses: Image stabilisation. cpn.canon-eurpe.com Infobank.

[15] Nikon vr. http://nikonimaging.com/global/technology/vr/index.htm.

[16] Sigma-lenses-overview. http://www.sigmaphoto.com/lenses/lenses.asp.

[17] Tamron europe-vibration compensation. http://www.tamron.de/Vibration-Compensation-VC.575.0.html.

[18] Panasonic megaois explained. http://www2.panasonic.com.

[19] Sony-glossar-deutschland. http://www.sony.de/glossary/.

[20] Pentax-shake reduction fact sheet. SHAKEREDUCTIONFACTSHEET.pdf.

[21] Olympus-image stabilizer. http://www.olympus-europa.com/consumer/dslr_16742.htm.

[22] Juan J. De La Cierva. Gyroscopic image motion compensator for a motion picture camera, Jan 1974.

[23] Alan C. Brooks. Real-time digital image stabilization. *EE 420 Image Processing Computer Project*, Mar 2003.

[24] Prof. Dr. Oskar von Stryk. *Robotik 1.* TU Darmstadt - Fachgebiet Simulation und Systemoptimierung, 2004.

[25] E.A. Maxwell. *The Methods of Plane Projective Geometry Based on the Use of General Homogeneous Coordinates.* Cambridge, England: Cambridge Univ. Press, 1946.

[26] E.A. Maxwell. *General Homogeneous Coordinates in Space of Three Dimensions.* Cambridge, England: Cambridge Univ. Press, 1951.

[27] John J. Craig. *Introduction to Robotics 2nd Edition.* Addison-Wesley Publishing Company Inc, 1989.

[28] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans ASME J. Appl. Mech*, 23:215–221, 1955.

[29] Richard A. Tapia; Cynthia Lanius. Computational science: Tools for a changing world. page Chapter 3.3.4, 2001.

[30] Prof. Dr.-Ing. J. Adamy. *Robotik und Künstliche Intelligenz.* TU Darmstadt - Institut für Automatisierungstechnik, 2003.

[31] Analog Devices, Inc. *ADXRS300*, 2004. Rev. B.

[32] Atmel. *ATmega128*, Nov 2004. Rev. 2467M-AVR-11/04.

[33] Renesas. *SH7125 Group; SH7124 Group Hardware Manual*, Sep 2006. Rev. 2.00.

[34] Maxim. *MAX3311E/MAX3313E*, Jan 2001.

[35] Maxim. *MAX3221/MAX3223/MAX3243*, Oct 2002.

[36] Dynamixel. *AX-12*, Jun 2006.