

**Friedrich-Alexander-Universität Erlangen-Nürnberg**



**Lehrstuhl für Informationstechnik  
mit dem Schwerpunkt Kommunikationselektronik**



Professor Dr.-Ing. Jörn Thielecke

**Diplomarbeit**

**Thema:**

Horizontale Geschwindigkeitsregelung eines Quadrocopter mit Hilfe von  
Laserdaten

Bearbeiter: B.Eng. Matthias Welter

Betreuer: Dipl.-Inf. Manuel Stahl  
Dipl.-Ing. Christian Strobel

Beginn: 01. August 2014

Ende: 31. Januar 2015

---

## Bestätigung

---

Erklärung:

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und, dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 31.01.2015

---

Matthias Welter

---

## Thema und Aufgabenstellung

---

**Thema:**

Horizontale Geschwindigkeitsregelung eines Quadrocopter mit Hilfe von Laserdaten

**Aufgabenstellung:**

Um das manuelle sowie automatisierte Navigieren eines Quadrocopters in der horizontalen Ebene zu vereinfachen ist es von Vorteil, die Bewegung ausschließlich in Form von Geschwindigkeiten in x- und y-Richtung vorzugeben. Manuell soll die Vorgabe über die Fernsteuerung erfolgen. Für das automatisierte Navigieren ist eine Schnittstelle zum Übergeben der Sollwerte vorzusehen. Die Geschwindigkeit ist anhand der vom Laserscanner erfassten Daten zu ermitteln.

***Ziel ist es eine Regelung zu entwerfen, welche die horizontale Geschwindigkeit des Quadrocopters auf den Sollwert einregelt.***

Optional kann eine automatisierte relative Positionsverschiebung des Quadrocopters implementiert werden.

Die Arbeitsschritte sind:

- Literaturrecherche
- Auswahl und Integration einer geeigneten Methode zur Bestimmung der relativen Position aus den Laserdaten
- Bestimmung der Geschwindigkeit in der x-y-Ebene
- Entwurf und Implementierung einer Geschwindigkeitsregelung
- Optional: Integration einer automatisierten relativen Positionsverschiebung

**Klassifikation:**

Robotik, Regelungstechnik, Informatik, Elektrotechnik, Sensorik

---

## Kurzzusammenfassung

---

*Hier soll eine kurze Zusammenfassung der Arbeit eingefügt werden, in der grob umrissen wird, um welches Thema es sich bei der Arbeit dreht und die Ergebnisse, die erzielt worden sind. Die Kurzzusammenfassung soll nur eine halbe bis dreiviertel Seite lang sein, auf keinen Fall länger als eine Seite!*

---

## Abstract

---

*Die englische Version der Kurzzusammenfassung. Für die Länge gelten die Gleichen Vorgaben wie für die deutsche Version.*

---

## Vorwort

---

*Hier können allgemeine Hinweise zur Arbeit gegeben werden, bspw. wie man mit englischen Begriffen, Abkürzungen und Codeabschnitten umgeht. Der nachfolgende Text kann als Beispiel gesehen werden, ist aber keinesfalls verpflichtend und sollte der eigenen Konvention angepasst werden!*

---

Diese Arbeit behandelt ein aktuelles technisches Thema, die Verwendung von englischen Begriffen ist unumgänglich. Soweit sinnvoll findet eine deutsche Übersetzung Verwendung. Nicht übersetzbare Begriffe, die eine wichtige Bedeutung für diese Arbeit haben, werden in einer Fußnote erklärt. Ausdrücke und Bezeichnungen aus Standards werden allgemein nicht übersetzt. Englische Begriffe sind im Text kursiv geschrieben. Wörter, die im deutschen Sprachgebrauch alltägliche Anwendung finden, wie beispielsweise Computer, Software, Internet etc., sind nicht kursiv geschrieben.

Bei erstmaliger Verwendung von Abkürzungen wird die volle Bezeichnung ausgeschrieben und das Kürzel dahinter in Klammern gesetzt. In der Folge wird nur die Abkürzung benutzt.

Quelltexte von Programmen sowie programmiertechnische Bezeichnungen und Schlüsselwörter werden durch die Verwendung von Schreibmaschinenschrift hervorgehoben.

Am Anfang der Arbeit befindet sich ein Abkürzungsverzeichnis, in dem alle in dieser Arbeit genannten Abkürzungen und deren ausgeschriebenen Formen enthalten sind. Im Anschluss an den Ausblick werden die wichtigsten Begriffe im Glossar zusätzlich kurz erläutert.

---

## Abkürzungsverzeichnis

---

AscTec ASCENDING TECHNOLOGIES

ENU East-North-Up

FCU Flight Control Unit

FOAW First-Order Adaptive Windowing

HLP High Level Processor

I<sup>2</sup>C Inter Integrated Circuit

ICP Iterative Closest Point

IIR Infinite Impulse Response

IMU Inertial Measurement Unit

IP Internet Protocol

LLP Low Level Processor

NED North-East-Down

PC Personal Computer

ROS Robot Operation System

SPI Serial Peripheral Interface

UART Universal Asynchronous Receiver/Transmitter

VLSAM Visual Simultaneous Localization and Mapping

---

# Inhaltsverzeichnis

---

Abkürzungsverzeichnis	i
1 Einleitung	1
2 Systemarchitektur des Quadrocopters	2
2.1 Grundlegende Funktionsweise eines Quadrocopters . . . . .	2
2.2 Hardwareaufbau . . . . .	4
2.3 Softwarestruktur . . . . .	6
3 Grundlagen	9
3.1 Das Robot Operation System <i>Robot Operation System (ROS)</i> . . . . .	9
3.2 Einführung in die Koordinatensysteme und Koordinatentransformationen . .	11
3.2.1 Koordinatensysteme . . . . .	11
3.2.2 Koordinatentransformationen . . . . .	14
4 2D Positionsbestimmung	18
4.1 Orthogonale Laserprojektion . . . . .	19
4.2 Positionsbestimmung über Scanmatching . . . . .	22
5 Positionsregelung	29
5.1 Struktur der Positionsregelung . . . . .	30
5.2 Modellbildung . . . . .	33
5.3 Exakte Zustandslinearisierung . . . . .	37
5.4 Vorsteuerung/Referenzmodell . . . . .	42
5.5 Folgeregler . . . . .	44
5.5.1 Einstellung der Dynamik mittels Polvorgabe . . . . .	49

5.5.2	Automatische Optimierung der Reglerparameter . . . . .	52
5.6	Zustandsschätzung . . . . .	52
5.6.1	Geschwindigkeitsbestimmung über die Inertialsenorik . . . . .	53
5.6.2	Geschwindigkeit als Ableitung der Position . . . . .	54
5.6.3	Geschwindigkeitsbestimmung über die Methode First-Order Adaptive Windowing . . . . .	56
5.6.4	Bestimmung der Zustände mittels eines Fusionsfilter . . . . .	61
6	Systemarchitektur des Quadrocopters	68
6.1	Grundlegende Funktionsweise eines Quadrocopters . . . . .	68
6.2	Hardwareaufbau . . . . .	70
6.3	Softwarerestruktur . . . . .	73
	Glossar	76
	Literaturverzeichnis	77
	Abbildungsverzeichnis	79
	Tabellenverzeichnis	81
A	Datenblätter	82
B	Definition flacher Mehrgrößensysteme	89
C	Simulinkmodell	92

# KAPITEL 1

---

## Einleitung

---

Anmerkungen die in der Einleitung auftauchen sollen.

Positionsregelung wird zur Geschwindigkeitsregelung missbraucht. Soll Position wird einfach aufintegriert.

# KAPITEL 2

---

## Systemarchitektur des Quadrocopters

---

Zu Beginn wird in diesem Kapitel die Grundlegende Funktionsweise des Quadrocopters erläutert. Anschließend die bei dieser Arbeit zum Einsatz kommende Hardware dargelegt und wie die einzelne Komponenten miteinander verknüpft sind. Dabei geht darum aufzuzeigen, an welchen Stellen Software bereits fest implementiert ist und wo eigene Algorithmen integriert werden können.

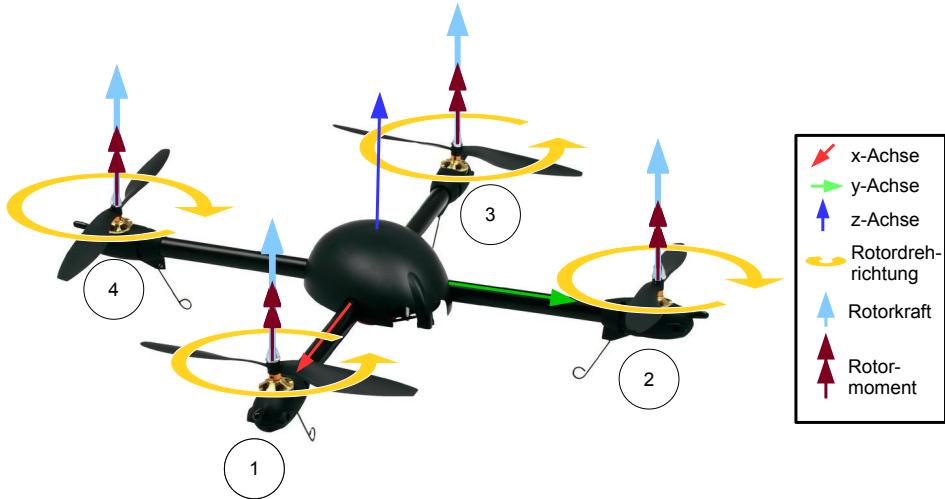
### 2.1 Grundlegende Funktionsweise eines Quadrocopters

Ziel dieses Kapitel ist es ein Verständnis dafür zu erlangen, wie über die gezielte Ansteuerungen der vier Rotoren eine Bewegung des Quadrocopters hervorgerufen werden kann. Dabei wird auf die wirkende Kräfte und Momente eingegangen, ohne tief in Physik einzusteigen.

Anhand der Drehzahl  $n_i$  der Rotorblätter lässt sich individuell der Schub der Rotoren einstellen. Somit lässt sich die Kraft  $F_i$  jedes Quadrocopterarme vorgeben. Die Gesamtkraft aller Rotoren ergibt den Schubvektor  $S^b$ .

$$S^b = \begin{bmatrix} S_x^b \\ S_y^b \\ S_z^b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} \quad (2.1)$$

Damit der Qudrocopter eine Bewegung im Raum vollziehen kann muss dieser Vektor aus der Vertikalen ausgelenkt werden. Dies wird durch eine Änderung der Lage realisiert. Reduziert man zum Beispiel die Drehzahl  $n_1$  und erhöht gleichzeitig die Drehzahl  $n_3$ , hat das



**Abbildung 2.1:** Momente und Kräfte an einem Quadrocopter

resultierende Kräfteungleichgewicht ein positives Moment um die  $y^b$ -Achse zur Folge. Der Quadrocopter dreht sich um die  $y^b$ -Achse. Der Pitch-Winkel ändert sich. Der Quadrocopter erfährt in der horizontalen Ebene des Raums eine Beschleunigung. Das gleiche Prinzip gilt auch für den Roll-Winkel, sprich Rotation um die  $x^b$ -Achse. Hier ist allerdings der Drehzahldifferenz zwischen  $n_2$  und  $n_4$  verantwortlich für die Rotation.

Eine Änderung der Orientierung um die Hochachse  $z$ , sprich Änderung des Yaw-Winkels ist ebenfalls über Variation der Rotordrehzahlen hervorgerufen. Dabei kommt der Effekt zum Tragen, dass die umgebende Luft entgegen der Drehrichtung der Motoren eine Kraft auf die Rotorblätter erzeugt und somit eine Moment auf den Quadrocopter. Diese Momente die an Armen des Quadrocopters angreifen lassen sich zur Vereinfachung in den Schwerpunkt verschieben. Damit bei gleicher Drehzahl aller Rotorblätter, einen Momentengleichgewicht herrscht drehen sich die Motoren eins und drei gegen, die Motoren zwei und vier mit dem Uhrzeigersinn. Um nun die gewünschte Rotation zu erzielen erhöht man die Drehzahl  $n_1$  und  $n_3$ , reduziert dabei gleichzeitig  $n_2$  und  $n_4$ . Das Ergebnis wäre in diesem Fall eine Rotation in positiver Richtung.

Zusammenfassen lassen sich die für die Rotation um die Quadrocopter-Achsen verantwortlichen Momente  $M_{x,y,z}^b$  in einem Vektor  $M^b$ .

$$M^b = \begin{bmatrix} M_x^b \\ M_y^b \\ M_z^b \end{bmatrix} = \begin{bmatrix} I(F_3 - F_1) \\ I(F_2 - F_4) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} \quad (2.2)$$

Aufzuklären ist warum mir einer Erhöhung der Drehzahl immer auch eine Reduzierung des Gegenparts verknüpft ist. Die Begründung ist, dass der Schubvektor  $S^b$  durch eine Rotation möglichst wenig beeinflusst werden soll. Damit er durch die Gesamtschubvorgabe ganze einfach bestimmt werden.

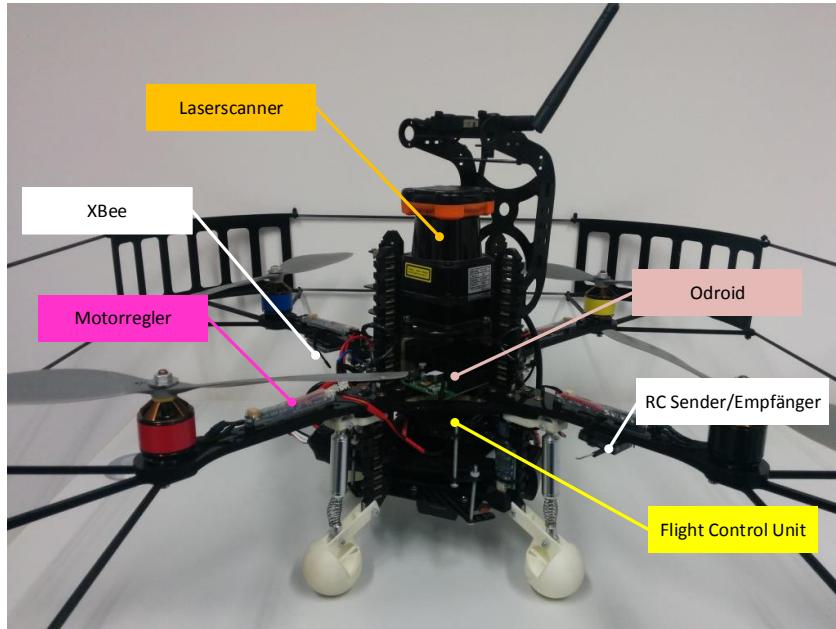
## 2.2 Hardwareaufbau

Zum Einsatz kommt der AscTec Pelican der Firma *ASCENDING TECHNOLOGIES (AscTec)*. Dieser Quadrocopter ist speziell für die Forschung entworfen worden. Seine Turmstruktur ermöglicht eine einfache Integration zusätzlicher Sensoren und Nutzlasten. Durch diese Flexibilität im Aufbau ist es Ziel dieses Teilkapitels einen Überblick zu geben, wo die einzelnen Komponenten positioniert sind. Begleitend zum Text ist der Aufbau in Abbildung 6.2 sowie etwas ausführlicher, mit den Daten der Komponenten, im Anhang dargestellt.

Für jeder der vier mit einem Propellor verbundenen Elektromotoren, ist ein separater Motorcontroller zuständig. Diese sorgen dafür, dass sich die von der *Flight Control Unit (FCU)* angeforderten Drehzahlen einstellen.

Die *FCU* ist die zentrale Steuer- und Regeleinheit des Quadrocopters. Sie besitzt zwei ARM7 Prozessoren, einen *Low Level Processor (LLP)* und einen *High Level Processor (HLP)*. Außerdem verschiedene Kommunikationsschnittstellen (vgl. Kapitel 6.4). Zusätzlich dient sie als inertiale Messeinheit (engl. *Inertial Measurement Unit (IMU)*). Diese Einheit wird zur Bewegungsdetektion sowie zur Bestimmung der Lage und Ausrichtung benötigt. Sie ist nicht zur Positionsbestimmung in einem ortsfesten Koordinatensystem (Koordinaten- systeme siehe Kapitel HIER MUSS EINE REF hin) geeignet. Bestandteile der IMU sind ein 3D-Beschleunigungssensor, drei Drehratensensoren(Gyros), einem Kompass sowie einem Drucksensor zur Ermittlung der Flughöhe anhand des Luftdrucks. Verbaut sind die Sensoren mit Ausnahme des Kompass direkt auf der Platine (siehe Abbildung 6.3).

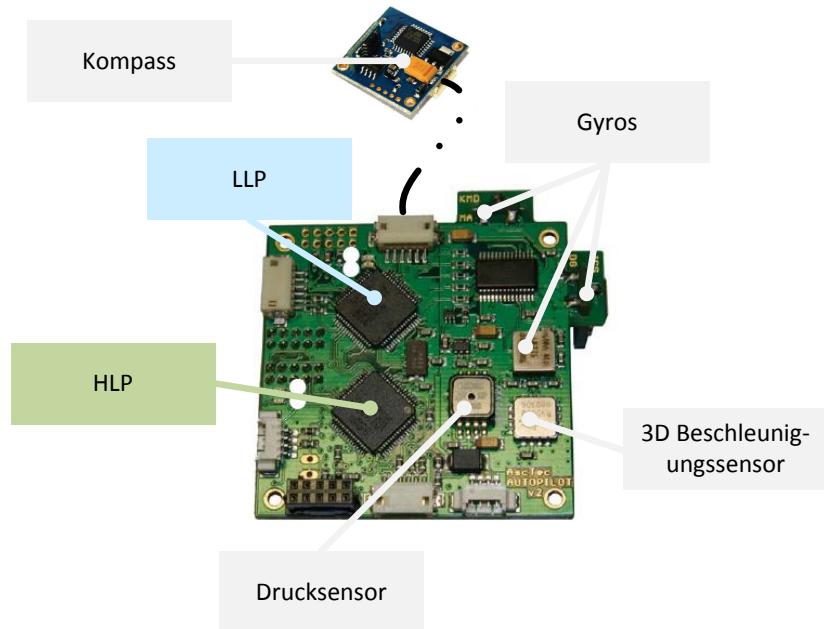
Da der Einsatzbereich im Indoorbereich liegt, ist Drucksensor ist zur Höhenbestimmung in geschlossenen Räumen nicht eignet, da er erst ab einer Höhe von 5m zuverlässige Werte



**Abbildung 2.2:** Hardwareaufbau des Quadrocopters...DIESE GRAFIK IST EIN PLATZHALTER GRAFIK NUR MIT NAMEN DER KOMPONENTEN

liefert. Daher wurde in einer vorangegangen Arbeit von Jan Kallwies (LITERATURVERWIES JAN) die Hardware um ein Modul zur Messung der Höhe im Indoorbereich erweitert. Auf diesem Modul befinden sich ein zwei Infrarotsensoren für den Nahbereich. Beide zusammen decken einen Messbereich von Bereich von 4 cm bis 142 ab. Erweitert wird der Messbereich durch einen Ultraschallsensor für Entfernung von bis zu 5 m. Aus diesen drei Sensordaten wird über einen Extended-Kalman-Filter die Flughöhe bestimmt. Eine genaue Beschreibung dieses Fusionsfilters kann in der Arbeit von Jan Kallwies [LITERATURVERZEICHNIS] nachgelesen werden. Da in dieser Arbeit die Navigation in der horizontale Ebene den Schwerpunkt darstellt, wird dieses Modul nicht weiter behandelt.

Um allerdings in der Horizontalen navigieren zu können, muss die Position des Flugkörpers in der x-y Ebene (VGL koORDINATENSYSTEME) bekannt sein. Da dies, wie schon beschrieben, nicht mit der Inertialsenorik möglich ist, wurde in die Turmstruktur der Laserscanner UTM-30LX der Firma Hokuyo integriert. Dieser Scanner hat eine maximale Reichweite von 30 m und Abtastbereich von 270°. Die Umlaufdauer beträgt dabei 40 Hz, d.h. alle 25 ms steht ein neuer Umgebungsscan zur Verfügung.



**Abbildung 2.3:** Platine der FCU

Damit zur Berechnung der Position sowie Implementierung weiterer Algorithmen und Funktionen ausreichend Rechenleistung vorhanden ist, befindet sich auf dem Quadrocopter ein zusätzlicher Odroid-X Mikrocomputer mit einem Quad Core Prozessor mit 1.4 Ghz und einen 1024MB LP-DDR2 Arbeitsspeicher. Außerdem besitzt diese Entwicklungsplattform sechs USB-Schnittstellen sowie ein 10/100Mbps Ethernet-Anschluss.

Nun sollte man einen Überblick über die im Quadrocopter verbauten Komponenten besitzen. Wie die Einheiten untereinander vernetzt sind, darauf wird im folgenden Kapitel 6.3 eingegangen.

## 2.3 Softwarestruktur

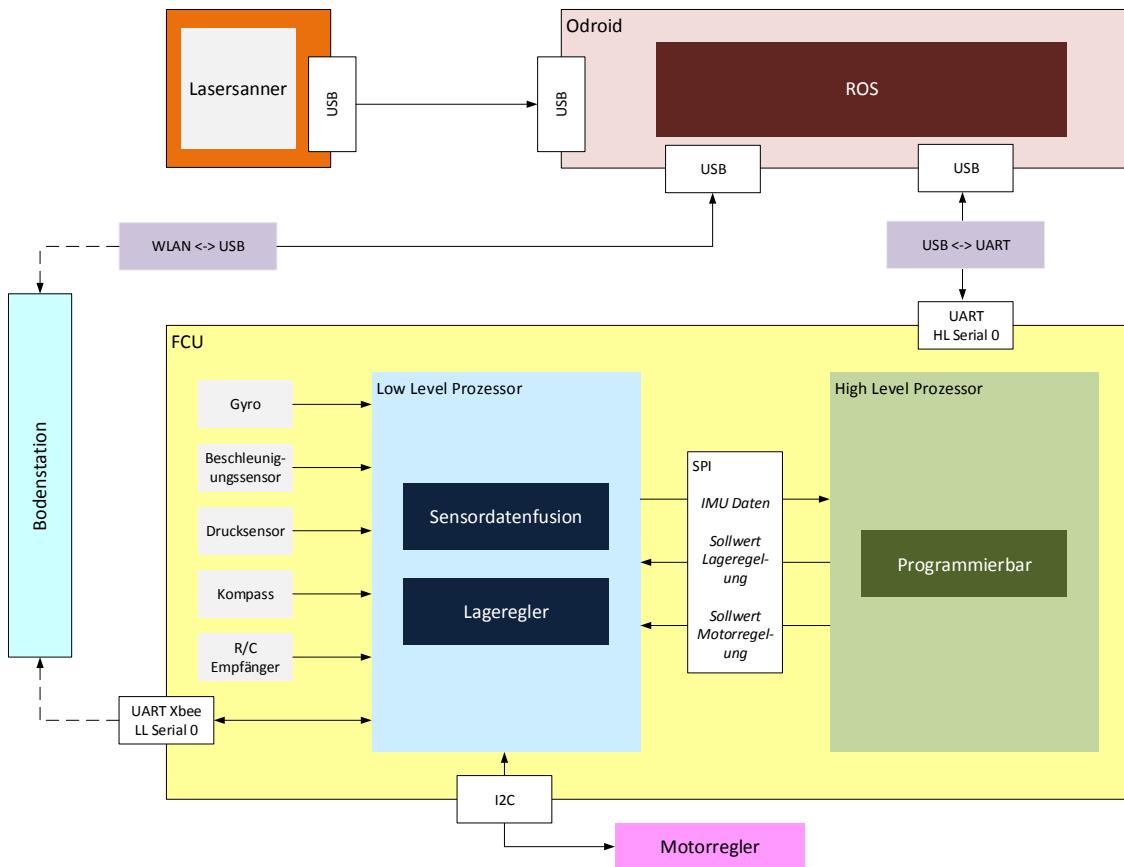
Nachdem im vorhergegangen Kapitel 6.2 die verbaute Hardware vorgestellt wurde, geht es in diesem Abschnitt um die Softwarestruktur (Abbildung 6.4). Es wird aufgezeigt welche Software bereits fest implementiert ist und wo adaptive Applikationen integriert werden können. Des weiteren wird die Kommunikationsstruktur dargelegt, wie und über welche

Protokolle die einzelnen Komponenten mit einander kommunizieren.

Beginnend mit der *FCU*, deren beiden Prozessoren *LLP* und *HLP* die mit einer Frequenz 1kHz getaktet und über einem *Serial Peripheral Interface (SPI)* Bussystem verknüpft sind, wird zunächst der *LLP* betrachtet. Auf dem Low Level Prozessor befinden sich die Sensordatenfusion der *IMU*-Sensorik zur Lagebestimmung des Quadrocopters. Aufbauend darauf stabilisiert die implementierte Lageregelung das Flugverhalten. Dabei werden die geforderten Sollwinkel bzw. Solllage, die dem *LLP* über die Fernbedienung oder den *HLP* übergeben wird, eingestellt. Kombiniert mit der Schubvorgabe werden den Motorreglern die jeweiligen Solldrehzahlen der Rotoren über einen *Inter Integrated Circuit (I<sup>2</sup>C)*-Bus, serieller synchroner Zweidraht-Bus, übergeben. Diese Algorithmen sind fest eingepflegt. *AscTec* stellt hier dem Benutzer eine Art White-Box zur Verfügung, d.h es ist bekannt was integriert ist, allerdings nicht wie es umgesetzt ist. Überwachen lässt sich der *LLP* über einen externen *Personal Computer (PC)*, in Abbildung 6.3 als Bodenstation bezeichnet. Zur Kommunikation benötige werden zwei XBee Funkmodule. Eines ist am *Universal Asynchronous Receiver/Transmitter (UART)* LL-Serial0 Port der *FCU* angeschlossen, das andere am USB Port der Bodenstation. Mit der AutoPilot Software lassen sich so unter anderem der Akkustand, die *IMU*-Daten sowie die Stellgrößen der Fernsteuerung betrachten. Außerdem ist es möglich Parameter der Sensorfusion und Lageregelung auszulesen und zu verändern.

Mit dem *HLP* stellt *AscTec* eine Entwicklungsumgebung zur Implementierung eigener Algorithmen auf der *FCU* zur Verfügung. Hier können erweiternde Programmteile integriert werden die den Lageregler des *LLP* ansprechen oder die direkt den Motorcontroller mit Solldrehzahlen speisen. Die zweite Möglichkeit ist der Grund warum keine Änderungen, abgesehen von den Parametern, am *LLP* vorgenommen werden können. So gibt es bei Experimentalflügen immer eine sichere Rückfallebene. Möglich ist dies, da der *HLP* über die Fernsteuerung aktiviert und deaktiviert werden kann.

Wie schon in Kapitel 6.2 beschrieben, befindet sich auf dem Quadrocopter zur Erhöhung der Rechenleistung der Odroid-X. Anders wie bei den auf der *FCU* befindlichen Prozessoren, besitzt das Odroid Bord ein Betriebssystem. Dabei diesem handelt es sich um das Open-source Betriebssystem Ubuntu 13.04. Dieses wurde ausgewählt, da es die Installation eines weiteren Opensource Betriebssystems ermöglicht, dem *ROS*. Einem Software Framework für Roboteranwendungen(siehe Kapitel 3.1). Zum Einsatz kommt der Odroid-X bei der Implementierung der Positionsbestimmung(Kapitel VERWEIS). Verbunden ist es zum einen über



**Abbildung 2.4:** Kommunikationsstruktur des Quadrocopters Kompass auf deutsch ROS auch Programmierbar Namen der UART Ports einfügen

einen USB-Port mit dem Laserscanner, zum anderen ist mit einem weiteren USB-Anschluss über den HL-Serial0 Port mit dem HLP verknüpft. Von der Bodenstation kann über WLAN eine SSH Verbindung aufgebaut werden, und somit die Entwicklungsplattform bedient werden.

Nun ist bekannt wie die einzelnen Komponenten untereinander vernetzt sind. Somit lässt sich im weiteren Verlauf der Arbeit nachvollziehen, an welchen Stellen die Anwendungen implementiert werden und über welche Verbindungen sie untereinander kommunizieren.

# KAPITEL 3

---

## Grundlagen

---

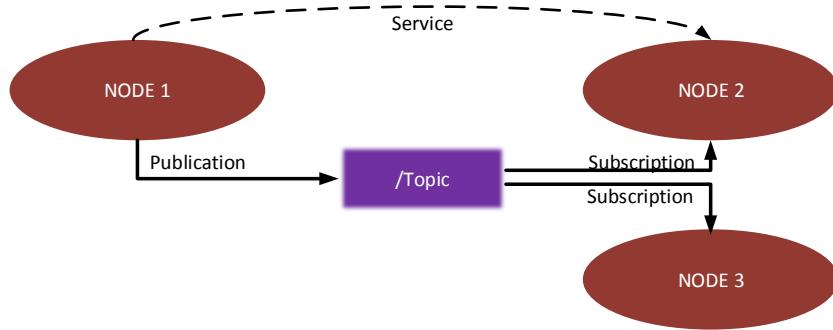
Das Kapitel Grundlagen behandelt die Themen, die in mehreren Abschnitten dieser Arbeit relevant sind. Dabei handelt es sich um das Robot Operation System, die verwendeten Koordinatensysteme und die Transformation zwischen ihnen.

### 3.1 Das Robot Operation System *ROS*

Ziel dieses Unterkapitel ist es das Opensource Betriebssystem *ROS* vorzustellen. Wie es aufgebaut ist und welche Vorteile es besitzt.

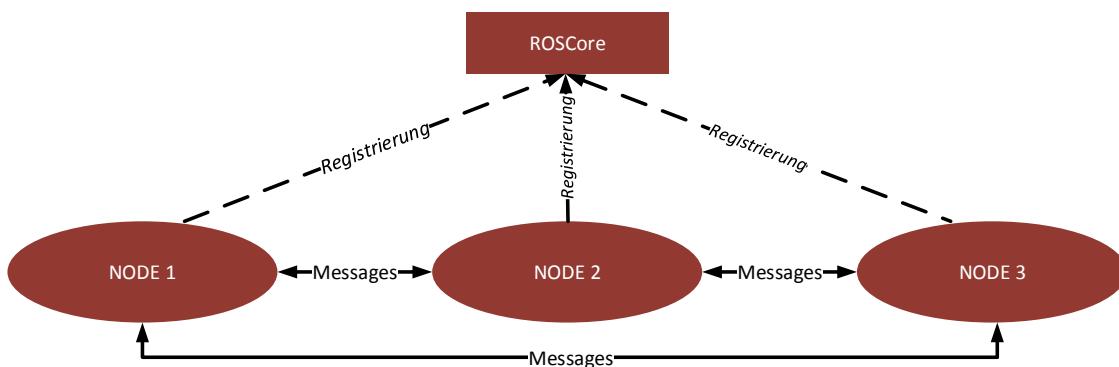
*ROS* stellt dem Softwareentwickler Bibliotheken und Werkzeuge zur Verfügung, die helfen Roboteranwendungen zu erstellen. Das auf einem *Internet Protocol (IP)*-basierende modulare Kommunikationsframework ermöglicht die Verknüpfung von Anwendungssoftware, Sensoren und Aktoren sogar unter mehreren Robotern. Die Grundlage dafür ist die sogenannte Hardwareabstraktion. Dabei wird durch hardwarespezifische Module erreicht, dass Komponenten unterschiedlicher Hersteller miteinander verbunden werden können. In unserem Fall Hokuyo Laserscanner und *AscTec FCU*. Außerdem ermöglicht es eine hardwareunabhängige Programmierung, die in den Programmiersprachen C/C++ oder in Python erfolgen kann. Jede Hardwareabstraktion oder Anwendung wird als Node, bzw. Konten bezeichnet und läuft als eigener Prozess.

Der Austausch von Daten zwischen den Nodes erfolgt über so genannte Topics (Abbildung 3.1]. Dabei werden von den Knoten Nachrichten (engl. Messages) in Topics gepostet und somit veröffentlicht (publication). Benötigt ein weiterer Knoten den Inhalt dieses Topic


**Abbildung 3.1:** Kommunikation von Nodes über Topics und Services

kann er es abonnieren (subscription). Sobald die Nachricht im Knoten aktualisiert wurde, wird sie den abonnierenden Knoten übertragen. Dabei sind Knoten nicht auf ein Topic beschränkt, es können beliebig viele Topics beschrieben oder empfangen werden. Alternative zu dieser Art der asynchronen Datenübertragung, biete *ROS* die Möglichkeit einer Synchrone Kommunikation zwischen zwei Nodes über Services. Dabei wird auf einem Knoten ein Service gestartet. Dieser dient als Server und agiert nach dem Anfrage-Antwort-Prinzip. Schickt ein anderer Knoten eine Anfrage, wird ihm die geforderte Nachricht zu gesendet.

Anzumerken ist, das durch das verwendete *IP*-Protokoll keine deterministische Versendung der Nachrichten nicht gewährleistet ist, da es sein kann, das Nachrichten gleichen Types in Paketen zusammengefasst werden. Bei der Programmierung empfiehlt es sich daher auf Topics mit einem Zeitstempel (engl. timestamp) zurückzugreifen. Die Echtzeitfähigkeit des *ROS* ist durch allerdings nicht gefährdet.


**Abbildung 3.2:** Registrierung der Knoten

Der wohl größte Vorteil von *ROS* ist die ständig wachsende Community. So stellen Forscher aus der ganzen Welt ihre Algorithmen und Hardwareabstraktionen zur Verfügung. Dadurch ist es möglich bei der Erstellung einer Roboteranwendung auf Bausteine zurück zugreifen, die ohne diese Plattform selbst zu implementieren wären. Abgesehen davon stellt *ROS* eine Vielzahl von Hilfsmitteln wie zum Beispiel die Transferfunktion (*/tf*) bereit. Hier lassen sich Koordinatensysteme definieren. Die Transformation der Daten wird dann automatisch von *ROS* durchgeführt.

## 3.2 Einführung in die Koordinatensysteme und Koordinatentransformationen

Anhand von Koordinatensystemen und Transformationen lässt sich die Lage von Punkten und Objekten in einen Raum mathematisch beschreiben. Die Grundvoraussetzung zur Bestimmung der Position des Quadrocopters im 2D-Raum (siehe Kapitel HIER MUSS NOCH EINE REFERENZ HIN). Außerdem ermöglicht die Einführung von Koordinatensystemen die mathematisch/physikalische Beschreibung des Quadrocopters und stellt somit die Grundlage zur Modellbildung und Reglerentwurf (siehe Kapitel so und so).

### 3.2.1 Koordinatensysteme

Über ein Koordinatensystem lässt sich ein Vektor oder die Position eines Punktes bezogen auf den Koordinatenursprung in einer zweidimensionalen Ebene, bzw in einem dreidimensionalen Raum beschreiben. Ziel dieser Teilabschnittes ist die Erläuterung der in dieser Arbeit eingeführten Koordinatensysteme.

Zu Beginn werden nun zwei Konventionen bezüglich der Bezugssysteme vorgestellt. Nummer eins, die in Abbildung 3.3a dargestellte *East-North-Up (ENU)* Konvention. Diese wird in vor allem bei der Landnavigation eingesetzt. Hier zeigt die z-Achse nach oben. Bei der zweiten Konvention, hauptsächlich in der Wasser-, Luft- und Raumfahrt eingesetzt, handelt es sich um das *North-East-Down (NED)* Bezugssystem (Abbildung 3.3b). Die z-Achse zeigt nach unten. Anzumerken ist, dass in dieser Arbeit die Ausrichtung der x- und y -Achse nicht wie in Abbildung 3.3 und auch der Namensgebung entsprechend den

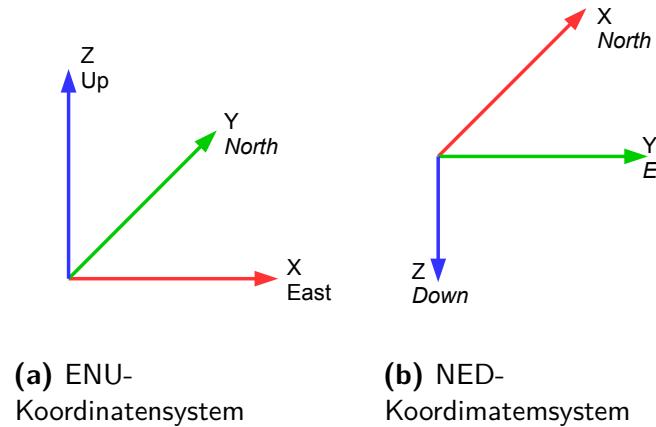


Abbildung 3.3: test

Himmelsrichtungen entspricht. Die Begriffe *ENU* und *NED* dienen hier zur Beschreibung der Ausrichtung der Koordinatenachsen in Abhängigkeit der positiven z-Achse.

Bei der nun folgenden Einführung der Koordinatensysteme (Abbildung 3.4) handelt es ausschließlich um kartesische, das heißt orthogonale Koordinatensysteme, die nach der *ENU* Konvention ausgerichtet sind. Dies steht erstmal im Widerspruch mit dem Abschnitt zuvor, dort ist das *NED* als Koordinatensystem für Flugkörper eingeführt worden. Es ist allerdings so, dass die *ROS* Koordinatensysteme auf *ENU* basieren. Deshalb die Wahl von Bezugssystemen mit positiver z-Achse nach oben.

Wie aus Abbildung 3.4 zu entnehmen sind vier xyz-Koordinatensysteme definiert.

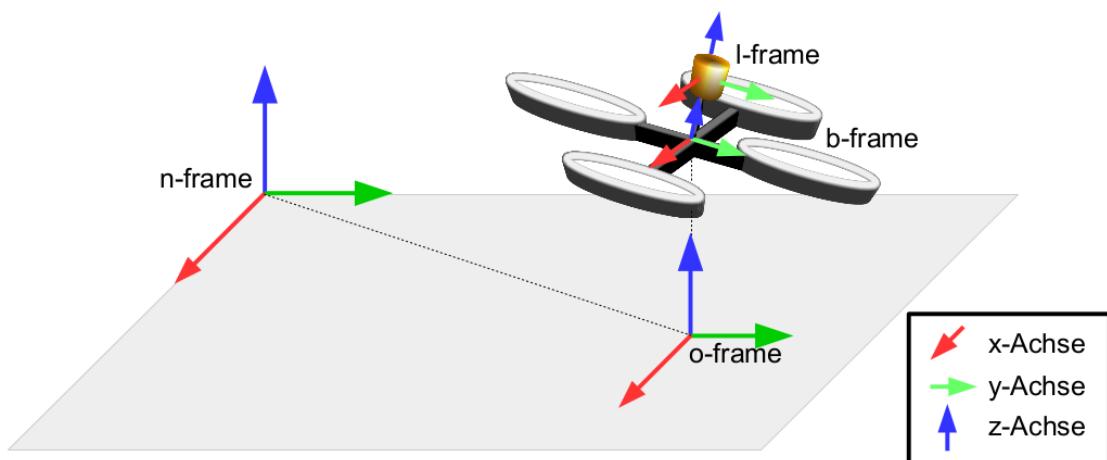


Abbildung 3.4: In der Arbeit angewandte Koordinatensysteme

- **n-frame(Lokaler Navigationsframe):** Ortsfestes Koordinatensystem zur Beschreibung der Position im Raum. Da es in dieser Arbeit um die horizontale Positionsregelung geht, ist hier ausschließlich die xy-Ebene von Interesse. Der Ursprung des Koordinatensystems wird bei jedem Systemstart neu initialisiert. Zu beachten ist dies beim vollautonomen Flug in Räumen, dabei beziehen sich die Sollpositionen nicht auf ein Raumkoordinatensystem mit festem Ursprung, sondern auf den beim Systemstart initialisierten Bezugspunkt. Keinen Einfluss hat diese Tatsache auf die Geschwindigkeitsregelung per Fernsteuerung, da hier die relative Bewegung von Interesse ist.

Es sei darauf hingewiesen, dass die Verwendung eines xyz Navigationsframes die Krümmung der Erdoberfläche vernachlässigt. Diese ist legitim, da die Drohne in Gebäuden zum Einsatz kommt. Möchte man jedoch Weltweit navigieren, benötigt man ein rotationselliptische Koordinatensysteme[THIELECKE LITVERWEIS]

- **b-frame(Bodyframe):** Dieses Koordinatensystem ist fest mit dem Rahmen des Quadrocopters verbunden. Man spricht dabei von einen körperfesten Koordinatensystem. Dabei befindet sich der Ursprung des Systems im Schwerpunkt, die x-Achse zeigt in die als Vorne definierte Richtung. Die y- und z-Achse sind abhängig davon nach der *ENU* Konvention angeordnet. Informationen die sich auf dieses Referenzsystem beziehen sind unter anderen die *IMU*-Daten. Außerdem lässt sich mit diesem System die Lage des Quadrocopters im n-frame über die Position des Nullpunkts und Drehwinkel beschreiben.
- **l-frame(laserframe):** Ebenfalls ein körperfestes Koordinatensystem. In die dem Entfernungsmessungen des Lasers aufgetragen werden. Der Bezugspunkt liegt dabei in der Sendequelle des Lasers. Die Ausrichtung der Achsen entspricht der des b-frames, mit Ausnahme eines Offsets in z-Richtung.
- **o-frame(Orthogonalframe):** Hierbei handelt es sich um ein objektbezogenes Bezugssystem, dessen Orientierung um seine z-Achse und die Position des Ursprungs im n-frame abhängig von dem Orthogonal über der Ebene befindlichen b-frame ist. Dadurch wird der Quadrocopter in der zu Navigierenden xy-Ebene abgebildet.

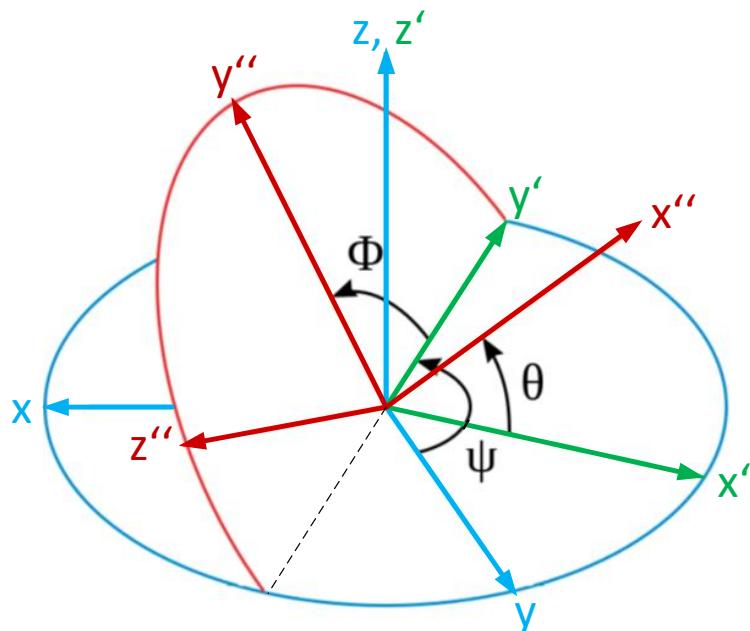
Da sich Messwerte wie zum Beispiel die *IMU*-Daten oder die Laserdaten auf unterschiedlichen Koordinatensysteme beziehen, benötigt man Koordinatentransformation(Kapitel 3.2.2).

Mit Hilfe derer lassen sich die Vektoren und Koordinaten in die verschiedenen Bezugssysteme übertragen.

### 3.2.2 Koordinatentransformationen

Damit Daten eines Referenzsystems in einen anderen transformiert werden können, muss deren Orientierung zueinander beschreibbar sein. Nach [Buchholz Flugregelung] ist dies über die Rotationswinkel  $\phi$  (Rollwinkel/engl. roll), Rotation um die x-Achse sowie der Winkel  $\theta$  (Nickwinkel/engl. pitch) und  $\psi$  (Gierwinkel/engl. yaw) für die y- und z-Achse möglich. Die Reihenfolge um die Achsen gedreht werden, ist dabei nicht beliebig. Sie ist in verschiedenen Konventionen festgelegt. In dieser Arbeit wird die in der Luftfahrt- und Fahrzeugtechnik gebräuchliche  $z,y',x''$ -Konvention (Abbildung 3.5) angewendet.

Das Koordinatensystem wird zu Beginn um den Winkel  $\psi$ , d.h. um die z-Achse gedreht. Daraus ergibt sich das in Abbildung 3.5 grün eingezeichnete Koordinatensystem. Dieses rotiert man anschließend um die Achse  $y'$ , sprich den Winkel  $\theta$ . Zuletzt erfolgt eine Drehung mit dem Winkel  $\phi$  um die  $x''$ -Achse. Das Ergebnis ist das rote  $x'',y'',z''$ -Koordinatensystem. Es sei nochmal darauf hingewiesen, dass die Reihenfolge der Winkel einzuhalten ist, da sonst die



**Abbildung 3.5:**  $z,y',x''$ -Konvention

beschriebene von der tatsächlichen Lage abweicht. Ein veranschaulichendes Beispiel worin unterschiedliche Abfolgen bei der Rotation führen ist in der Literatur [Literaturverzeichnis] von Herr Thielecke zu finden.

Nach Luftfahrtkonvention lässt sich eine Transformationsmatrix  $M$  aufstellen, mit der Vektoren und Koordinaten vom xyz-Koordinatensystem (Bsp.: n-frame) in das x"y"z"-Koordinatensystem (Bsp.: b-frame) überführen lassen. Dafür benötigt man zunächst die drei Transformationsmatrizen, die jeweils eine Rotation um eine Koordinatenachse beschreiben[Literaturverzeichnis]. Diese sind wie folgt definiert:

- Drehung um die z-Achse mit dem Winkel  $\psi$

$$M_z = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

- Drehung um die y-Achse mit dem Winkel  $\theta$

$$M_y = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \quad (3.2)$$

- Drehung um die x-Achse mit dem Winkel  $\phi$

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \quad (3.3)$$

Aus diesen Rotationsmatrizen lässt sich über Matrizenmultiplikation eine Gesamttransformationsmatrix aufstellen. Die Reihenfolge der Multiplikation entspricht der in der Konvention festgelegten Drehfolge, von rechts nach links gelesen. Somit er gibt sich:

$$\begin{aligned}
 M_{bn} &= M_x \cdot M_y \cdot M_z \\
 &= \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi & \sin \phi \cos \theta \\ \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi & \cos \phi \cos \theta \end{bmatrix} \quad (3.4)
 \end{aligned}$$

Mit Hilfe dieser Transformationsmatrix lässt sich jetzt ein Vektor zum Beispiel aus dem n-frame ins b-frame übertragen.

$$\begin{bmatrix} x^b \\ y^b \\ z^b \end{bmatrix} = M_{bn} \cdot \begin{bmatrix} x^n \\ y^n \\ z^n \end{bmatrix} \quad (3.5)$$

Handelt es sich bei um eine Koordinate, ist zusätzlich noch der Abstand der Koordinatenursprünge zu addieren. Für die Rücktransformation muss die Gesamttransformationsmatrix transponiert werden. Daraus folgt,

$$\begin{bmatrix} x^n \\ y^n \\ z^n \end{bmatrix} = M_{bn}^T \cdot \begin{bmatrix} x^b \\ y^b \\ z^b \end{bmatrix} = M_{nb} \cdot \begin{bmatrix} x^b \\ y^b \\ z^b \end{bmatrix} \quad (3.6)$$

Nun lassen sich Vektoren in beide Richtungen in die verschiedenen Bezugssysteme überführen. Nachteil der Methode mit Eulerwinkel ist, das diese auf Grund der trigonometrischen Funktionen nur für Winkel  $\phi, \theta, \psi = \{x \in \mathbb{R} | -\pi \leq x \leq \pi\}$  eindeutig durchführbar ist. Wird dieser Bereich überschritten, lässt sich die Lage über Quaternionen beschreiben. Durch die Beschreibung der dreidimensionalen Orientierung in einem vierdimensionalen Raum lassen, ist die Lage auch für Rotationen um eine Vielfache von  $2\pi$  eindeutig charakterisiert. Die genaue Definition findet sich in der Literatur [16] und [2]. Da die Definitionsbereich der Eulerwinkel für den in der Arbeit betrachteten Bereich ausreicht ist ausschließlich die Umrechnung der in Quaternion  $(w_q, x_q, y_q, z_q)$  angegebenen Orientierungsdaten der IMU in Eulerwinkel  $(\phi, \theta, \psi)$ . Zu beachten ist, das die nun folgende Umwandlung nur für die  $z, y', x''$ -Konvention Gültigkeit besitzt.

$$\phi = \arctan\left(\frac{2(y_q z_q + w_q x_q)}{w_q^2 - x_q^2 - y_q^2 + z_q^2}\right) \quad (3.7)$$

$$\theta = \arcsin(2(w_q y_q - x_q z_q)) \quad (3.8)$$

$$\psi = \arctan\left(\frac{2(x_q y_q + w_q z_q)}{w_q^2 + x_q^2 - y_q^2 - z_q^2}\right) \quad (3.9)$$

Mit dieser letzten Umrechnung sind alle Grundlagen für die Arbeit gelegt. So bilden die Koordinatensystem und Transformationen die Basis für die nachkommende Positionsbestimmung.

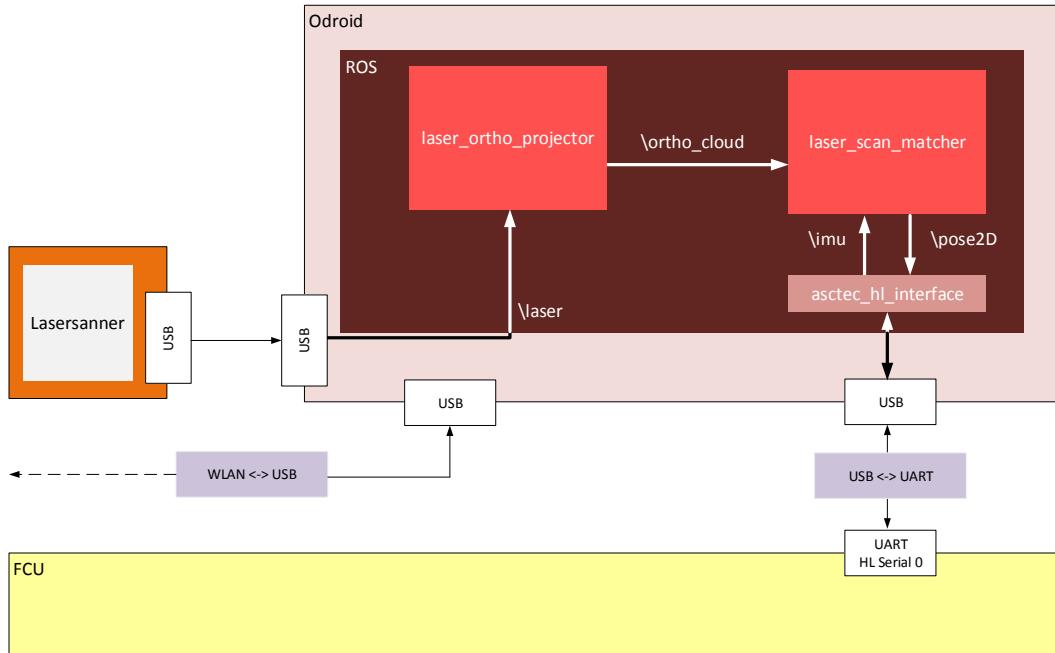
# KAPITEL 4

---

## Zweidimensionale Positionsbestimmung des Quadrocoters in xy-Ebene des Navigationsframes

---

Sowie in der Aufgabenstellung beschrieben, erfolgt die Positionsbestimmung über den auf dem Quadrocopter montierten Lasersanner. Man spricht hierbei von einem Onboard-Lokalisierungssystem. Dabei erzeugt der Laser einen zweidimensionalen Scan der Umgebung. Die aufgenommen Entferungen sind im l-frame definiert, d.h. sie beziehen sich auf Ursprung des Lasers. Damit enthalten diese Rohdaten keine Information über die Position und Orientierung des Quadrocopters in der xy-Ebene des Navigationskoordinatensystem (n-frame). Jedoch lässt sich aus Umgebungsscans mittels der Methode des „scanmatching“ die Position in einer zweidimensionalen Ebenen bestimmen (Kapitel 4.2). Damit diese jedoch für die Bestimmung der Quadrocopterposition in der horizontalen Ebene eingesetzt werden kann, ist es von Nöten die Laserdaten in das o-frame zu projizieren (Kapitel 4.1). Erst dann lässt sich für das o-frame und somit dem Quadrocopter, Position und Orientierung in der xy-Ebene des n-frames bestimmen. Beide Vorgänge sind bereits für *ROS* implementiert und sind Bestandteil der *scan\_tools*. Genauer gesagt handelt es sich dabei um dem „laser\_ortho\_projector“- und dem „laser\_scan\_matcher“-Knoten (Abbildung 4.1). Ziel der dieses Kapitel 4 ist es die Mathematik dahinter zu erläutern, sodass man eine Verständnis der Funktionsweise der Knoten erhält.



**Abbildung 4.1:** Verknüpfung des „laser\_ortho\_projector“- und des „laser\_scan\_matcher“-Knoten GRAFIK EVENTUELLE ÜBERARBEITEN DA ASC-TEC\_HL\_INTERFACE KNOTEN DER FÜR DIE KOMMUNIKATION VERANTWORTLICH IST FEHLT

## 4.1 Projektion der Laserdaten in das o-frame auf der xy-Ebene des n-frames („laser\_ortho\_projector“)

Die Projektion der Laserdaten ins o-frame erfolgt orthogonal zur xy-Ebene des n-frames.

In allgemeiner Form ist dies in Abbildung 4.2a dargestellt. Grundlage für die orthogonale Projektion ist die Annahme, dass es sich bei den erfassten Objekten um senkrechte Gegenstände handelt. Das heißt ihre Form variiert nicht mit der Höhe in der sie erfasst werden. Für geschlossene Räumen ist diese Annahme zutreffend, da es sich bei den Objekten hauptsächlich um senkrechte Wände handelt. Durch Erfüllung dieser Voraussetzungen kann die Flughöhe des Quadrocopters vernachlässigt werden. Dies kann man aus Abbildung 4.2b entnehmen. Eine Verschiebung des Koordinaten Ursprungs des b-frames auf der z-Achse des o-frames hat demzufolge keinen Einfluss auf die Projektion. Folglich kann für beide Koordinatensystem der identischen Ursprung angenommen werden. Unter Beachtung dieser Annahmen kann der Einfluss des Roll-( $\phi$ ) und Nickwinkels ( $\theta$ ) auf die Entfernungsmessung

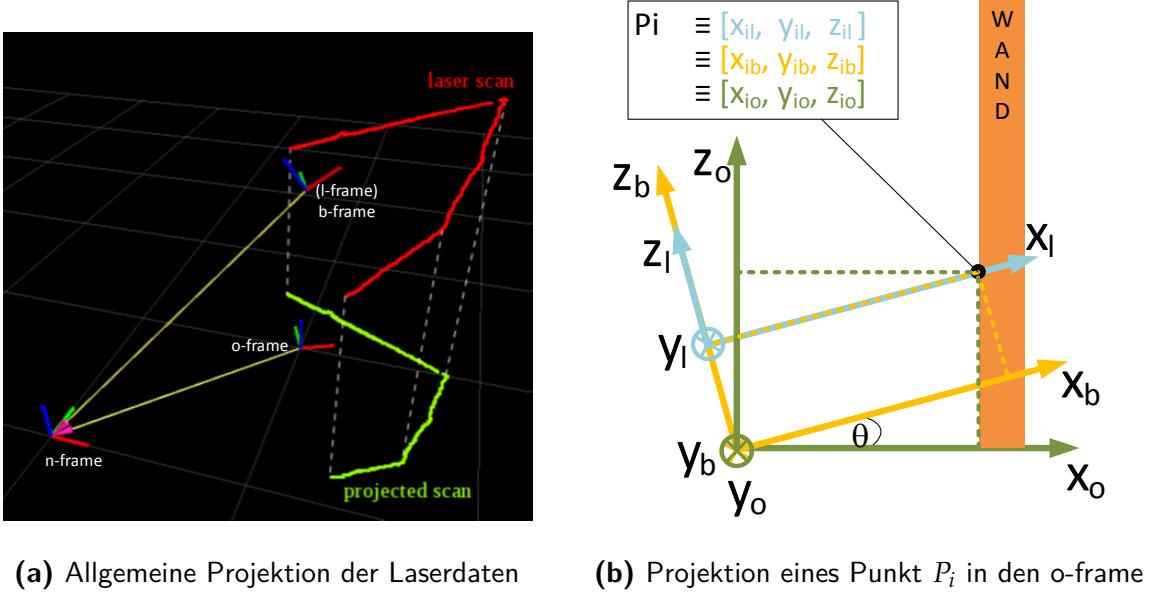


Abbildung 4.2: Projektion der Laserdaten

eliminiert werden. Die Winkelgrößen liefern die *IMU*. Der mathematische Ablauf der orthogonalen Transformation wird basierend auf Literatur [14] im Folgenden dargelegt.

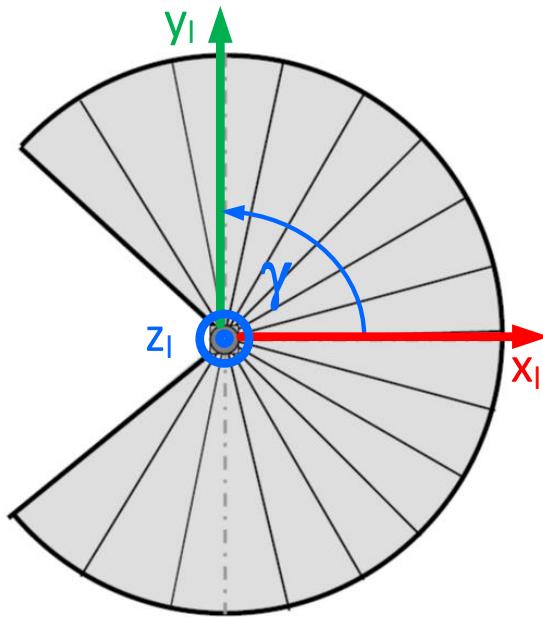
Entfernungsdaten eines Laserumlaufs besteht aus mehreren diskreten Abtastungen (Abbildung 4.3). Übergeben werden sie in Form von Entfernung  $r_i$  in einem Array (Topic \laser Abbildung 4.1). Mittels der Schrittweite von  $0.25^\circ$  lässt sich anhand des Indizes  $i$  für jeden Messpunkt einen Winkel  $\gamma_i$  zuweisen.

$$\gamma_i = 135^\circ - 0.25^\circ \cdot i \quad (4.1)$$

Die Entfernung eines Punktes  $p_i$  ist somit über  $\{r_i, \gamma_i\}$  definiert. Zur weiteren Verwendung ist es notwendig die Messungen im kartesischen Koordinatensystem des l-frame zu übertragen.

$$p_i^l = [\cos(\gamma_i) \cdot r_i, \sin(\gamma_i) \cdot r_i, 0]^T \quad (4.2)$$

Da der Bezugspunkt des o-frames im Schwerpunkt des Quadrocoptes liegen soll, in dem auch der b-frame seinen Ursprung hat, ist es von nötigen die Laserdaten vom l-frame ins



**Abbildung 4.3:** Draufsicht I-frame

b-frame zu transformieren. Wie schon in Kapitel 3.2 erwähnt handelt es sich dabei um eine konstante Transformation. Genauer gesagt um einen Offset von  $10\text{cm}$  auf der  $z^b$ -Achse, da der Laser oberhalb des Quadrocopterschwerpunktes montiert ist.

$$p_i^b = \begin{bmatrix} \cos(\gamma_i) \cdot r_i, & \sin(\gamma_i) \cdot r_i, & 0.1 \end{bmatrix}^T \quad (4.3)$$

Nun da die Laserpunkte im b-frame definiert sind, kann die Transformation der Umgebungsdaten in die xy-Ebene des o-frames erfolgen. Angesichts der Tatsache, dass die Winkel  $\phi$  und  $\theta$  als Verdrehung um die Achsen des o-frames definiert sind, benötigt man zur Umrechnung der Laserdaten die in Kapitel 3.2 eingeführte Gleichung 3.6 zur inversen Koordinatentransformation. Dabei wird der Yaw-Winkel zu Null gesetzt. Grund hierfür ist, dass Ausrichtung der Flugrichtung in der zweidimensionalen Ebene mit der des b-frames übereinstimmen sollen. Daraus ergibt sich für die Transformationsmatrix

$$M_{ob} = \begin{bmatrix} \cos \theta & \sin \phi \sin \theta & \cos \phi \sin \theta \\ 0 & \cos \phi & -\sin \phi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix} \quad (4.4)$$

, über die sich die Positionen der Laserpunkte  $P_i^b$  im o-frame bestimmten lassen.

$$p_i^o = M_{ob} \cdot P_i^b \quad (4.5)$$

Erneut kann unter der Annahme von rechtwinkligen Objekten die Höhe des Punktes in  $z^o$ -Achse zu Null gesetzt werden. Somit sind die Punkte  $p_i^l$  auf der xy-Ebene des o-frame folgendermaßen abgebildet.

$$p_i^o = \begin{bmatrix} \cos \theta \cos(\gamma_i) \cdot r_i + \sin \phi \sin \theta \sin(\gamma_i) \cdot r_i + \cos \phi \sin \theta \cdot 0.1 \\ \cos \phi \sin(\gamma_i) \cdot r_i - \sin \phi \cdot 0.1 \\ 0 \end{bmatrix} \quad (4.6)$$

Alle Punkte  $p_i^o$  eines Umlaufs einen Scan  $S$ .

$$S^o = [p_i^o | i = 1..1080] \quad (4.7)$$

Diese Beschreibung  $S$  der Laserscans im o-frame ist die Basis für die im anschließende Kapitel 4.2 behandelte Positionsbestimmung in der zweidimensionalen Ebene des n-frames.

## 4.2 Positionsbestimmung anhand der ins o-frame überführten Laserdaten über scanmatching

Aufbauend auf den im Kapitel 4.1 vorgestellten Laser\_ortho\_projektor, ist es Aufgabe des Folgenden Teilkapitels zu erläutern wie anhand eines Referenzscans  $S_{ref}$  und einem weiteren Scan  $S_{neu}$  die Position in der euklidischen xy-Ebene des n-frames bestimmt werden kann. Zur Anwendung kommt hier die Methode des Scanmatching. Dabei gilt die Annahme, das für jeden Scan  $S_{neu}$  und dessen dazugehörigen Position  $P_{neu}$  eine Rotation  $M_z^o$  um  $\psi^o$ , inklusive Translation  $T$  existiert, so dass die beiden Datenwolken  $S_{neu}$  und  $S_{ref}$  übereinander liegen (Abbildung 4.4). Definiert ist  $S_{ref}$  dabei an der Stelle  $P_{ref}$ .

Ausgangspunkt sind zwei im o-frame definierte Datenwolken.

$$S_{ref} = [p_{ref_i} | i = 1..n_{ref}] \quad (4.8)$$

$$S_{neu} = [p_{neu_i} | i = 1..n_{neu}] \quad (4.9)$$

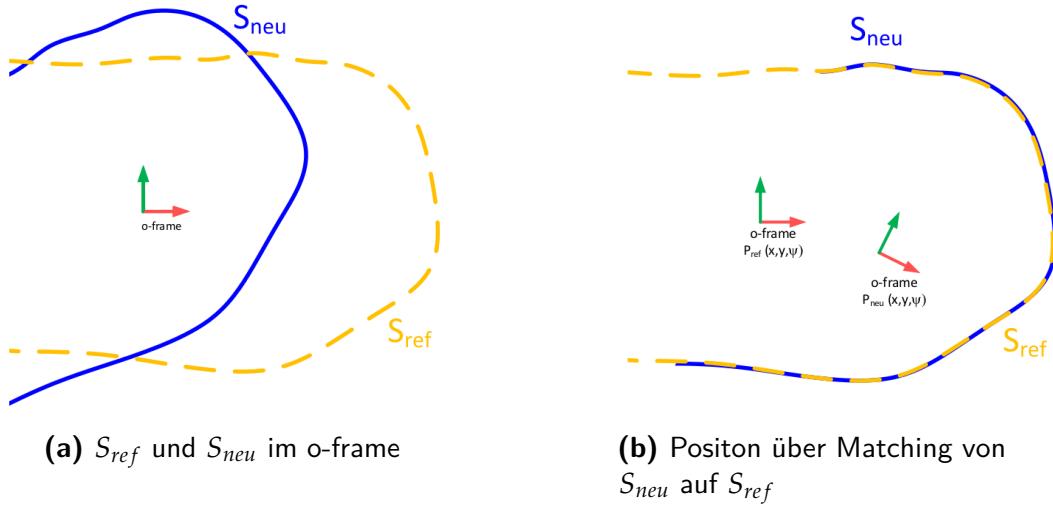


Abbildung 4.4: Prinzip Scanmatching

Dargestellt in Abbildung 4.4a.

Zur Bestimmung des Matchings bzw. der Rotation  $M_z^o$  und der Translation  $T$  wurde im Jahr 1992 unter anderem von Paul Besl der *Iterative Closest Point (ICP)*-Algorithmus entwickelt. Dabei handelt es sich um einen iterativen Algorithmus. Abgebildet sind die einzelnen Integrationsschritte in einem Flussdiagramm in Abbildung 4.5. Daran orientierend wird im folgenden jeder einzelne Vorgang erläutert.

- **Schritt 1,** Punktekorrespondenz.

Hierbei bekommt jeder Punkt des  $S_{neu}$  einen korrespondierenden Punkt aus der Datenwolke  $S_{ref}$  zugewiesen. Man spricht hierbei von der Point-to-Point Arithmetik. Als Startwert werden als Korrespondenzpunkte die am nächsten liegenden Nachbar übergeben. Die Suche der entsprechenden Werte erfolgt über die Methode der erschöpfenden Suche (Brute-Force-Methode), d.h. jeden Punkt werden alle Punktstände ermittelt und der Punkt mit der geringsten Entfernung zugewiesen. Das Ergebnis ist eine Datenmenge  $S'_{ref}$ , deren Anzahl an Werten denen von  $S_{neu}$  entspricht.

$$S'_{ref} = [p'_{ref_i} | i = 1..(n' = n_{neu})] \quad (4.10)$$

Dieser Schritt stellt auf Grund des nicht optimierten Suchalgorithmus den rechenaufwendigsten Teil dar. Laut [LITERATUR AUT4] stellt dies für die Verarbeitung von 2D-Laserscans kein Problem dar.

- **Schritt 2,** Bestimmung des Rotationswinkel  $\Delta\psi^o$  und der Translation  $T$

Wie schon zu beginn angeklungen, soll eine rotatorische und translatorische Transformation zur Überlappung von  $S_{neu}$  mit  $S_{ref}$  führen. Idealität und eine sehr kleine Änderung vorausgesetzt könnte jeder Punkt von  $S_{neu}$  in den entsprechenden Punkt  $S_{ref}$  umgerechnet werden.

$$p_{ref_i}(M_z^o(\psi^o), T) = M_z^o(\psi^o) \cdot p_{neu_i} + P_{ref} + T \quad (4.11)$$

Die Zweidimensionale Roationsmatrix  $M_z^o$  entspricht dabei der in Kapitel 3.2.2 eingeführten Koordinatentransformationsmatrix (3.1) reduziert auf die x und y Anteile.

$$M_z^o(\psi^o) = \begin{bmatrix} \cos \psi^o & -\sin \psi^o \\ \sin \psi^o & \cos \psi^o \end{bmatrix} \quad (4.12)$$

In der Praxis ist die Umgebung nicht ideal und aufgrund der hohen Dynamik können die Änderungen zwischen zwei Messwerten größer Ausfallen. Dadurch sind nicht alle Werte von  $S_{ref}$  und  $S_{neu}$  Indize für Indize vergleichbar. Der Grund warum in Vorgang 1 zu zum Scan  $S_{neu}$  ein korrespondier Scan  $S'_{ref}$  eingeführt wurde. Der im Folgenden Anwendung findet. Ein einsetzen von  $S'_{ref}$  in Gleichung (4.11) ist jedoch nicht die Lösung des Problems, da es zu keiner eindeutigen Ergebnis führt. Laut [11] kann aus dieser Formel (4.11) als Fehlgleichung herangezogen werden. Mit dieser lässt sich über die least-squar Methode die kleinste Summe der quadratischen Abweichungen ermitteln.

$$E(M_z^o(\Delta\psi^o), T) = \frac{1}{n'} \sum_{i=1}^{n'} \|p'_{ref_i} - (M_z^o(\Delta\psi^o) \cdot p_{neu_i} + T)\|^2 \quad (4.13)$$

Um das Minimum von abhängig  $E(M_z^o(\Delta\psi^o), T)$  bestimmen zu können wird nach [13] der Schwerpunkt ( $c_{ref}$ ,  $c_{neu}$ ) der korrespondierenden Punkte ermittelt.

$$c_{ref} = \frac{1}{n'} \sum_{i=1}^{n'} p'_{ref_i} \quad (4.14)$$

$$c_{neu} = \frac{1}{n'} \sum_{i=1}^{n'} p_{neu_i} \quad (4.15)$$

Damit ergibt sich die Datenwolken folgendermaßen beschreiben.

$$\tilde{S}'_{ref} = [\tilde{p}'_{pref_i} = p'_{pref_i} - c_{ref} | i = 1..n'] \quad (4.16)$$

$$\tilde{S}_{neu} = [\tilde{p}_{neu_i} = p_{neu_i} - c_{neu} | i = 1..n'] \quad (4.17)$$

Setzt man 4.16 und 4.17 in die Fehlgleichung 4.2 resultiert.

$$\begin{aligned} E(M_z^o(\Delta\psi^o), T) &= \frac{1}{n'} \sum_{i=1}^{n'} ||\tilde{p}_{ref_i} - M_z^o(\Delta\psi^o) \cdot \tilde{p}_{neu_i} - (T - c_{ref} + M_z^o(\Delta\psi^o) \cdot c_{neu})||^2 \\ &= \frac{1}{n'} \sum_{i=1}^{n'} ||\tilde{p}_{ref_i} - M_z^o(\Delta\psi^o) \cdot \tilde{p}_{neu_i} - \tilde{T}||^2 \end{aligned} \quad (4.18)$$

Über  $\tilde{T}$  ist die Abweichung der Schwerpunkte translatorisch als auch rotatorisch beschrieben. Damit beide Schwerpunkte genau über einander liegen ist  $\tilde{T} = 0$  zusetzen. Daraus ergibt sich.

$$0 = T - c_{ref} + M_z^o(\Delta\psi^o) \cdot c_{neu} \quad (4.19)$$

und eine Fehlgleichung die nun mehr nur noch von der Rotation abhängig ist und dessen Betrag sich wie folgt darstellen lässt.

$$\begin{aligned} E(M_z^o(\Delta\psi^o)) &= \frac{1}{n'} \sum_{i=1}^{n'} ||\tilde{p}_{ref_i} - M_z^o(\Delta\psi^o) \cdot \tilde{p}_{neu_i}||^2 \\ &= \frac{1}{n'} \sum_{i=1}^{n'} (\tilde{p}_{ref_i}^T \tilde{p}_{ref_i} + \tilde{p}_{neu_i}^T \tilde{p}_{neu_i} - 2\tilde{p}_{ref_i}^T \cdot M_z^o(\Delta\psi^o) \cdot \tilde{p}_{neu_i}) \end{aligned} \quad (4.20)$$

Weiterhin auf die Literatur [13] beziehend ist es zur Minimierung von  $E(M_z^o(\Delta\psi^o))$  ausreichend den gemischten Term zu  $\tilde{E}(M_z^o(\Delta\psi^o))$  maximieren.

$$\tilde{E}(M_z^o(\Delta\psi^o))_{max} = \underset{M_z^o(\Delta\psi^o)}{\operatorname{argmax}} \sum_{i=1}^{n'} (2\tilde{p}_{ref_i}^T \cdot M_z^o(\Delta\psi^o) \cdot \tilde{p}_{neu_i}) \quad (4.21)$$

Beziehungsweise abhängig von  $\Delta\psi^o$ .

$$\tilde{E}(\Delta\psi^o)_{max} = \operatorname{argmax} \sum_{i=1}^{n'} (2(\cos(\Delta\psi^o)(\tilde{x}_{ref} \cdot \tilde{x}_{neu} + \tilde{y}_{ref} \cdot \tilde{y}_{neu}) \\ + \sin(\Delta\psi^o)(\tilde{y}_{ref} \cdot \tilde{x}_{neu} + \tilde{x}_{ref} \cdot \tilde{y}_{neu})) \quad (4.22)$$

Aufgrund der trigonometrischen Addition besitzt der Summand eine Maximum für  $\Delta\psi^o$  wenn

$$\frac{\delta \tilde{E}(\Delta\psi^o)_{max}}{\delta(\Delta\psi^o)} = 0 \quad (4.23)$$

$$0 = \sum_{i=1}^{n'} (-\sin(\Delta\psi^o)(\tilde{x}_{ref} \cdot \tilde{x}_{neu} + \tilde{y}_{ref} \cdot \tilde{y}_{neu}) \\ + \cos(\Delta\psi^o)(\tilde{y}_{ref} \cdot \tilde{x}_{neu} + \tilde{x}_{ref} \cdot \tilde{y}_{neu})) \quad (4.24)$$

daraus folgt für  $\Delta\psi^o$

$$\Delta\psi^o = \arctan\left(\frac{\sum_{i=1}^{n'} (\tilde{y}_{ref} \cdot \tilde{x}_{neu} + \tilde{x}_{ref} \cdot \tilde{y}_{neu})}{\sum_{i=1}^{n'} (\tilde{x}_{ref} \cdot \tilde{x}_{neu} + \tilde{y}_{ref} \cdot \tilde{y}_{neu})}\right) \quad (4.25)$$

Mit dem Ergebnis für  $\Delta\psi^o$  kann unter Verwendung von Gleichung 4.19 die Translation T bestimmt werden.

$$T = c_{ref} - M_z^o(\Delta\psi^o) \cdot c_{neu} \quad (4.26)$$

Somit sind Translation und Rotation bestimmt.

- **Schritt 3,** Ermittlung der Summe der quadratischen Fehler  $E(M_z^o(\Delta\psi^o), T)$

Die aus der Formel 4.25 stammende Rotation und die mit Hilfe der Gleichung 4.26 berechneten Translation werden in die Formel des quadratischen Fehlers eingesetzt.

- **Schritt 4,** Vergleich von  $E(M_z^o(\Delta\psi^o), T)$  mit Schwellwert  $E_{max}$

Der in Vorgang 4 berechnete quadratische Fehler wird mit dem Schwellwert des Maximal zulässigen Fehlers  $E_{max}$  verglichen. Wird unterschritten ist das Abbruchkriterium erfüllt, handelt es bei Rotation  $\Delta\psi^o$  und Translation  $T$  Werte die Transformation bzw. die neue Position  $P_{neu}$  mit einer ausreichenden Genauigkeit beschreiben. Ist das Kriterium beginnt der *ICP*-Algorithmus bei Vorgang 1 und bestimmt neue Korrespondenzen.

Für die Positionsbestimmung im n-frame wird mit der ersten Position  $P_{ref}$  der Bezugspunkt des Navigationskoordinatensystems gesetzt. Darauf aufbauend wird die weiteren Translationen bzw. Rotationen addiert.

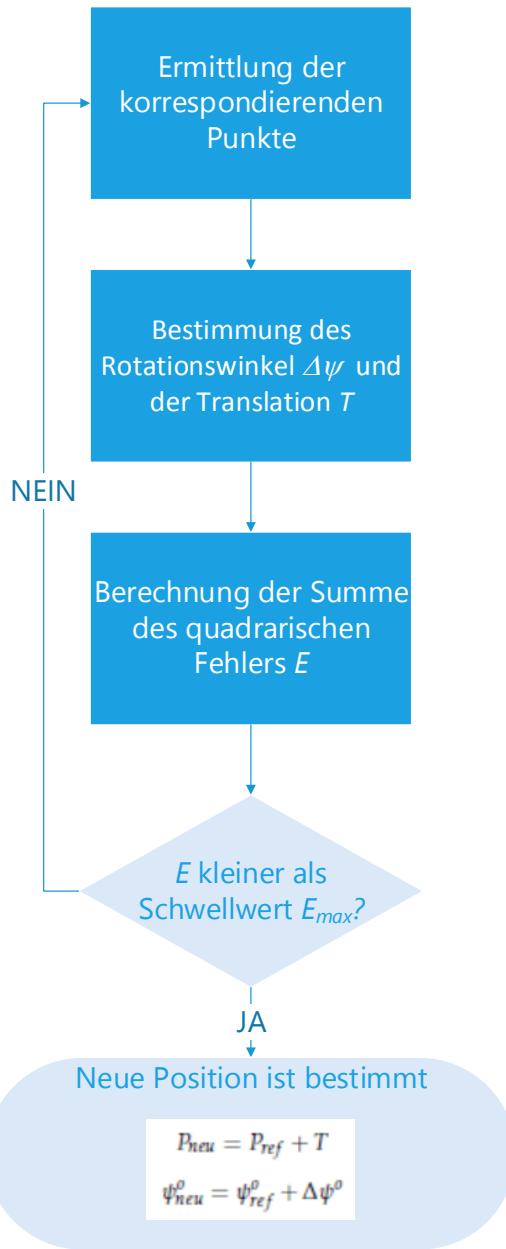
$$P_{neu} = P_{ref} + T \quad (4.27)$$

$$\psi_{neu}^o = \psi_{ref}^o + \Delta\psi^o \quad (4.28)$$

Anzumerken sei hier das  $P_{ref}$  nicht im Ursprung verweilt, sondern sich auf den Referenzscan  $S_{ref}$  bezieht. Dieser kann der vorige Umlauf des aktuellen Scans  $S_{neu}$  darstellen. Oder einen in der nahen Vergangenheit liegenden Scan, bei dem die Summe der quadratischen Fehler sehr gering war.

Der vorgestellte *ICP*-Algorithmus kann weiterhin verbessert werden. So kann anstelle der Point-to-Point Arithmetik in Vorgang 1 eine Point-to-Line Arithmetik zur Ermittlung der Konvergenzpunkte angewendet werden. Mit diesem Thema beschäftigt sich das Paper [3]. Außerdem ist möglich über die Inertialsenorik ausgehend von  $P_{ref}$  eine neue Position  $P_{imu}$  zu bestimmen. In  $P_{imu}$  wird der neue  $S_{neu}$  gelegt. So muss lediglich der Fehler der *IMU*-Positionsschätzung über den *ICP*-Algorithmus eliminiert werden. Dies ist besonders nützlich wenn zwei Scans weit auseinander liegen, wie zum Beispiel in Abbildung 4.4b. Es vereinfacht Vorgang 1. Behandelt wird dies unter anderem in [14].

Mit der Beschreibung Scanmatchingverfahren ist der letzte Baustein zur Lokalisierung des Quadrocopters in der horizontalen Ebene eines geschlossenen Raums mittels eines 2D Laser geliefert worden. Darauf aufbauend ist es möglich eine Positionsregelung zu implementieren.

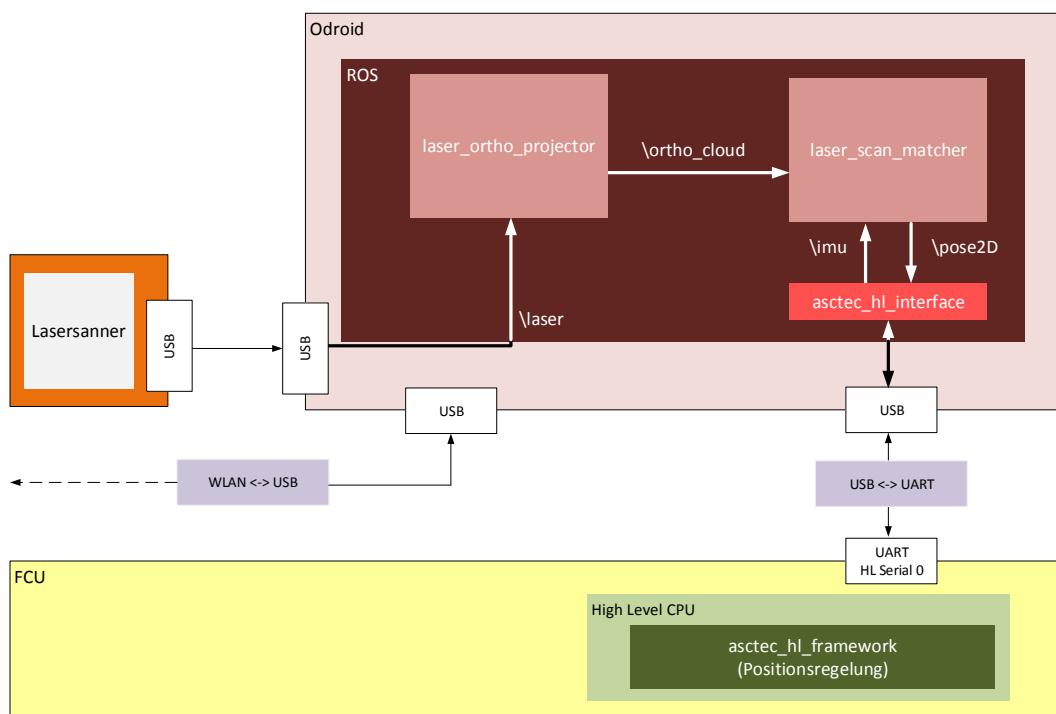


**Abbildung 4.5:** Flussdiagramm ICP-Algorithmus

# KAPITEL 5

## Verifizierung der Positionsregelung der ETH-Zürich in Verbindung mit einem Laserscanner

In einer Zusammenarbeit von *AscTec* und der ETH-Zürich wurde eine Regler zur Positionierung des Pelican Quadrocopters in geschlossenen Räumen entworfen und veröffentlicht. Der Entwurf basierte dabei auf einer monokularen Kamera über die mittels eines *Visual*



**Abbildung 5.1:** AscTec\_hl\_framework in der Kommunikationstruktur

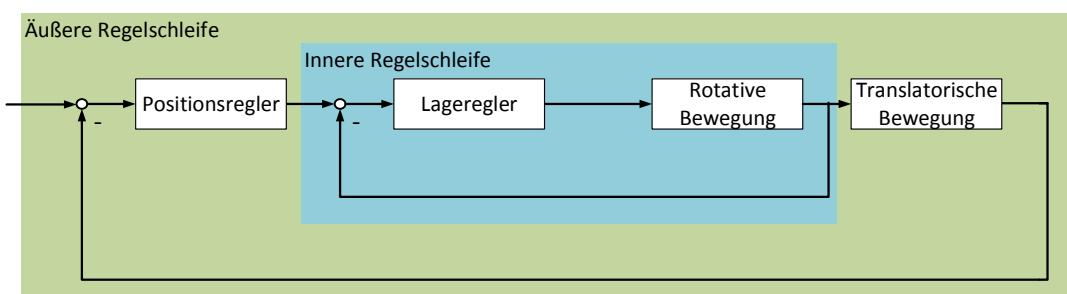
*Simultaneous Localization and Mapping (VLSAM)*-Algorithmus und der Fusion der *IMU* die Position des Flugobjekts in der unbekannten Umgebung ermittelt wird. Die dazugehörigen Regelung- und Fusionsalgorithmen sind im Paper [1] beschrieben. Außerdem stehen diese als asctec\_framework zum Download zur Verfügung und wurden im Rahmen dieser Arbeit auf den *HLP* geladen (Abbildung 5.1).

Aufgabe dieses Kapitel ist es die veröffentlichten Annahmen und Formel der Fusion und Regelung zu verifizieren. Dies erfolgt über Literaturrecherchen und Herleitung der publizierten Gleichungen. Unter Berücksichtigung, dass die Position nun mehr vom Laser über die Kapitel 4 vorgestellten Algorithmen bestimmt wird, ist Abschnitt 5.1 darauf ausgelegt eine Übersicht über die Bestandteile der Regelung zu geben. Herleitung der Komponenten erfolgt in den anschließend Unterkapiteln.

## 5.1 Struktur der Positionsregelung

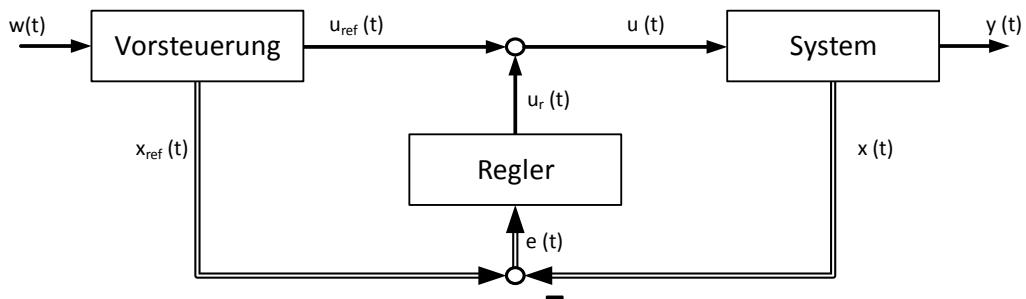
Das Hauptaugenmerk dieses Unterkapitels liegt darauf, wie sich die Positionsregelung in die in Kapitel 6.3 vorgestellte Systemstruktur einfügt. Welche Softwarekomponenten dafür auf dem *HLP* integriert wurden. Wie die Kommunikation zwischen ihnen und der Umgebung aussieht. Mit dem Ziel ein grundlegendes Verständnis für die Funktionsweise der Regelung zu generieren.

Die Positionsregelung wird auf die Lageregelung (engl. Attitudecontrol) aufgesetzt. Daraus resultiert eine Kaskadenstruktur (Abbildung 5.2). Diese Vorgehensweise ist nachvollziehbar, da die Lageregelung bereits fest auf dem *LLP* implementiert ist. Diese besteht aus einem Regelalgorithmus der anhand der Regeldifferenz der Orientierung  $e_{ori}$ , resultierend aus der



**Abbildung 5.2:** Kaskadenstruktur der vereinfachten Positionsregelung

Abweichung Soll-Orientierung  $O_{des}$  und Ist-Orientierung  $O$ , in Verbindung mit der Sollschubvorgabe  $T_s$  die Drehzahlen  $n_{1..4}$  der Rotoren berechnet und einstellt. Die Ist-Orientierung  $O = [\phi \ \theta \ \psi]^T$  wird mittels eines Fusionsfilter bestimmt. Dieser fusioniert die Messwerte der auf der *IMU* befindlichen Gyroscope mit den Daten des 3D-Kompass. Wie dieser unterlagerte Regler und der dazugehörige Zustandsschätzer genau ausgeführt sind ist nicht bekannt. Eine mögliche Ausführung ist in Paper [7] beschrieben. Die Ungewissheit über den Regleraufbau der Lageregelung stellt für den Entwurf der überlagerten Positionsregelung kein Problem dar. Von Interesse ist lediglich, das die Annahme einer sehr hohen Dynamik dieses Reglers zur einer Vereinfachung des für die Positionsregelung benötigen Modells (Kapitel 5.2) führt. Hohe Dynamik bedeutet, das der Sollwinkel in einer sehr kurzen Zeit  $t \rightarrow 0 \text{ s}$  erreicht wird. Basierend auf einer Exakten Ein-/Ausgangslinearisierung der inneren Schleife (Kapitel ??), ist die äußere Reglerschleife zur Positionsregelung auf dem *HLP* realisiert. Dank der Inversion, realisiert Ein-/Ausgangslinearisierung, kann für die Positionierung des Quadrocopters eine linerare zwei Freiheitsgrade Regelung angewandt werden. Diese besteht aus einer Vorsteuerung und einem Folgeregel. Die Vorsteuerung auf dem *HLP* ist in Form eines Referenzmodell (Kapitel 5.4), das dem Ein-/Ausgangslinearisierung nachempfunden ist, ausgeführt. Anhand der vorgegebenen Soll-Position  $P_{des}^n = [x \ y \ z]^T_{des}$  wird eine Referenz-Trajektorie<sup>1</sup> zur Überführung des Quadrocopters aus der aktuellen Ist-Position in die Soll-Position berechnet. Ergebnis ist ein Stellwert für die Inversion, der unter theoretischer Betrachtung die gewünschte Bahnbewegung des Quadrocopters zur Ursache hat. In einem realen System ist dies durch ein Flüsse der Umgebung, bsp. Wind nicht gewähr-



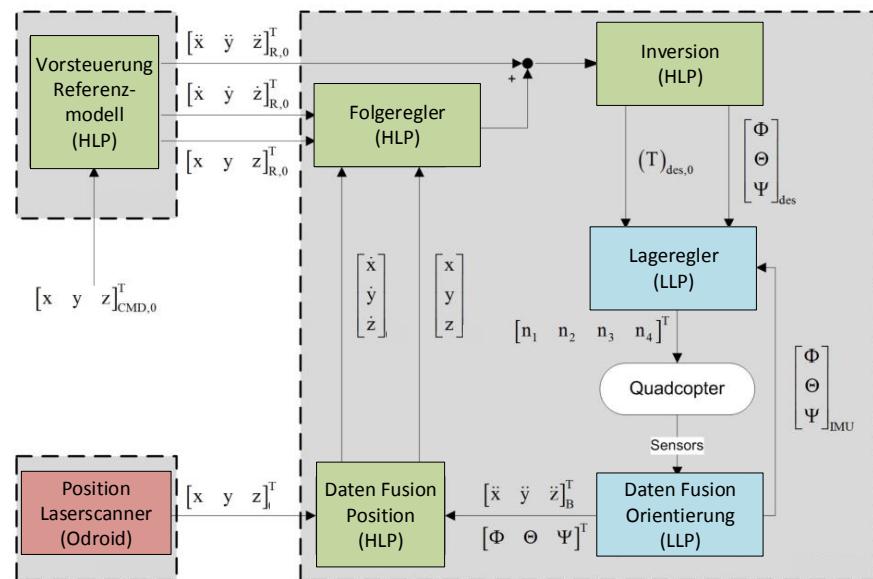
**Abbildung 5.3:** Struktur zwei Freiheitsgraderegelung

<sup>1</sup> Trajektorien beschreiben einen zeitabhängigen Verlauf eines Wertes in einem Bezugssystem

leiste. Deshalb ist zusätzlich der Folgeregler (Kapitel 5.5) implementiert, dessen Aufgabe besteht darin Abweichung der realen Zustände des Quadrocopters von den Referenzwerten auszuregeln. Die dafür benötigten Zustandsgrößen des Flugsystems (Kapitel 5.6) werden durch die Fusion der in über den Laser bestimmten Position (Kapitel 4) mit den mit den Beschleunigungswerten der *IMU* ermittelt.

Bevor jede einzelne Komponente in den Folgekapiteln hergeleitet wird, ist in Abbildung 5.4 die Verknüpfung aller Komponenten noch einmal grafisch dargestellt.

Anzumerken ist, das die in Verbindung mit *AscTec* entworfene Positionsregelung, zur Ausrichtung des Quadrocopters in einem dreidimensionalen Raum entworfen ist. Für die vertikale Positionierung ist jedoch in der vorrangingen Arbeit[9] von Jan Kallwies bereits eine Regelung entworfen worden. Da diese Regelung Effekte wie den Groundeffekt<sup>1</sup> berücksichtigt, ist es Aufgabe einer dieser Arbeit folgenden Studentischen Projekt diese in das System der ETH-Zürich einzupflegen. Somit reduziert sich die Validierung der Reglerstruktur auf den horizontale Ebene.



**Abbildung 5.4:** Struktur der Regelung (AscTec\_Framework)

<sup>1</sup> In Bodennähe verhindert der Untergrund das schnelle Abströmen des durch die Rotoren erzeugten Luftstroms. Die daraus resultierende Krafterhöhung bei gleichbleibender Drehzahl der Rotoren, wird als Groundeffekt bezeichnet.

## 5.2 Modellbildung

Die Modellbildung ist die Grundlage für den systematischen Entwurf einer Zustandsregelung. Dabei wird das Systemverhalten in Form von Differentialgleichungen abgebildet. Diese beschreiben die zeitliche Veränderung einer Ausgangsgröße in Abhängigkeit ihrer zeitlichen Ableitungen sowie veränderlicher Eingangsgrößen. So lassen sich unter Beachtung der physikalischen Gesetze Bewegungsgleichungen aufstellen, welche das räumliche und zeitliche Verhalten einen mechatronischen Systems abbilden.

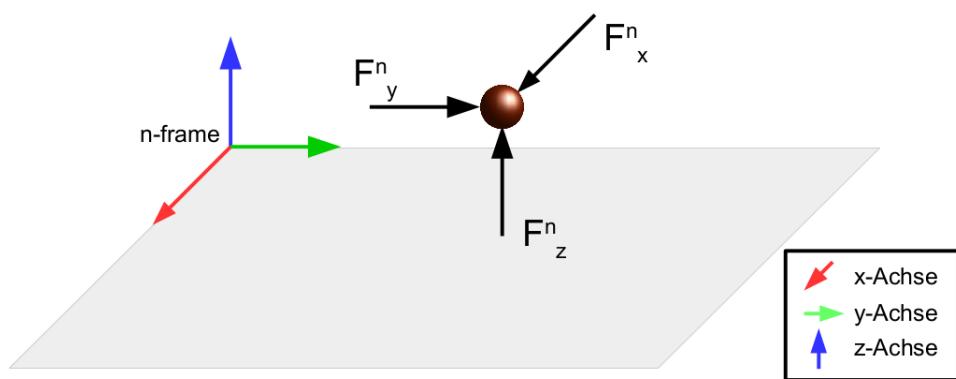
Bevor jedoch die Bewegungsgleichungen des Quadrocopter aufgestellt werden können, ist es notwendig die Freiheitsgrade des dynamischen Systems gegenüber des Bezugssystem zu bestimmen. Da zwischen dem n-frame(Bezugssystem) und dem b-frame(Quadrocopter) keine mechanische Verbindung oder konstante Koordinatentransformation existiert besitzt das Flugsystem sechs Freiheitsgrade. Bestehend aus drei rotativen  $[\phi \ \theta \ \psi]^T$  und drei translatorischen  $[x \ y \ z]^T$ . Sechs Freiheitsgrade sind gleichzusetzen mit sechs Differentialgleichungen zur Beschreibung der Bewegung im Raum. Damit ist jedoch nicht alle Dynamiken des Systems abgebildet. So sind zur Beschreiben des Gesamtsystems weitere Differentialgleichungen aufzustellen. Zum einen ist hier die Dynamik der Elektromotoren zu einzukalkulieren welche die Rotoren antreiben. Als auch die durch die Rotation der Rotorblätter ausgelöste Schubentwicklung nach den Gesetzen der Strömungslehre. Unter Beachtung aller Aspekte entsteht so eine mathematisch meist nichtlineare sowie sehr aufwendige und komplexe Systembeschreibung. Da mit der steigender Größe auch die Fehleranfälligkeit steigt, wird in der Modellbildung folgender Leitsatz immer wieder aufgeführt. „Ein Modell sollte das zu regelende Verhalten so einfach wie möglich, aber so detailliert wie nötig darstellen.“ Betrachtet nun die Struktur des implementierten Flugregelung (Abbildung 5.4) so lässt sich das Modell aus Sicht der Positionsregelung stark reduzieren.

Grund hierfür ist die in Bild 5.2 dargestellte Kaskadenstruktur der Regelung. In Verbindung mit Abbildung 5.4 ist zu erkennen, das der Lageregelung als Eingangsgrößen eine Soll-Orientierung  $O_{des} = [\phi \ \theta \ \psi]^T$  und ein Schubvorgabe  $T_{des}$  übergeben wird. Aus Sicht der Positionsregelung sind dies auch die Eingänge des zu regelnden Modells. Fest steht somit, das die rotative Dynamik als auch die Schubentwicklung über den auf dem *LLP* befindliche Regler eingestellt wird. Dieser ist ab Werk so gut parametriert, das die Zeitkonstante zwischen Vorgabe und Einstellen des Sollwerts sehr gering ist. Dies wurde auch durch einen Versuch bestätigt. Da der Aufbau der Lageregelung nicht bekannt ist wurde dazu

das experimentelle Systemidentifikationstool von Manfred Ottens herangezogen. Um die Zeitkonstante schätzen zu können wurde das Übergangsverhalten von Ist- zu Soll-Winkel als PT1-Glied abstrahiert. Aus den Messdaten des Sollwert und Istwert wurde daraus die Zeitkonstante T ermittelt. Dabei ergab das mittel über mehrere Messungen einen Zeitkonstante von  $T \approx 0.1$  s. Mit der Gewissheit, dass die Dynamik der Positionsregelung um ein vielfaches geringer ausfällt lässt sich beim Entwurf dieser die Zeitkonstante T vernachlässigen. Es gilt somit Soll- entspricht Ist-Winkel.

$$O_{des} = O = [\phi \ \theta \ \psi]^T \quad (5.1)$$

Somit reduziert sich die für Positionsregelung zu modellierende Dynamik auf die translatorische Differenzialgleichungen. Um zur Bestimmung dieser die Newtonsche Gesetze anwenden zu können, werden diese im n-frame definiert. Dabei kann der Quadrocopter als widerstandsfreie Kugel im dreidimensionalen Raum des Navigationskoordinatensystems abgebildet werden. Auf diese wirkt parallel zur z-Achse des n-frames die Gravitationskraft  $F_g^n$  und die in Richtung der  $z^b$ -Achse angreifender Gesamtschub  $T$  (Gleichung 6.1) der Rotoren. Letzt genannte Kraft muss zur Anwendung der Newtonischen Gesetze in Kraftkomponenten des n-frames zerlegt werden (Abbildung 5.5). Da die Dynamik der Lageregelung vernachlässigt wird, ist dies über eine einfache Koordinatentransformation vom b-frame ins n-frame realisierbar. Die entsprechende Transformation wurde in Kapitel 3.2.2 in Formel 3.6 eingeführt. Daraus ergibt sich,



**Abbildung 5.5:** Auf den Quadrocopter wirkende Kraft. Definiert im n-frame

$$F^n = \begin{bmatrix} F_x^n \\ F_y^n \\ F_z^n \end{bmatrix} = M_{nb} \cdot F^b - F_g = M_{nb} \cdot \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ F_g \end{bmatrix}. \quad (5.2)$$

Nach dem zweiten Newtonschen Gesetz lässt sich nun die Beschleunigung  $a$  des Körpers bestimmen. Dieses besagt, eine Änderung der Bewegung resultiert proportional und gradlinig in Richtung der wirkenden Kraft. Dabei gilt die Beziehung.

$$F^n = \begin{bmatrix} F_x^n \\ F_y^n \\ F_z^n \end{bmatrix} = m \cdot a = m \cdot \begin{bmatrix} a_x^n \\ a_y^n \\ a_z^n \end{bmatrix} \quad (5.3)$$

Für die konstante Masse  $m (= 1.863 \text{ kg})$  des Quadrocoters, lassen sich die Beschleunigungen des Körpers im dreidimensionalen Raum durch einsetzen von (5.2) in (5.3) berechnen.

$$\begin{bmatrix} a_x^n \\ a_y^n \\ a_z^n \end{bmatrix} = \frac{1}{m} \cdot (M_{nb} \cdot \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ F_g \end{bmatrix}) \quad (5.4)$$

Basierend auf dem Weg-Zeit-Gesetz [? ] lässt sich für eine Anfangsgeschwindigkeit  $v_0^n$  und eine Startposition  $P_0^n$  die Position des Quadrocopters über die doppelte Integration der Beschleunigung aus Gleichung (5.4) bestimmen.

$$\begin{aligned} a^n &= \dot{v}^n = \ddot{P}^n \\ v^n &= \dot{a}^n = \int a^n dt + v_0^n \\ P^n &= \int v^n dt + P_0^n \end{aligned} \quad (5.5)$$

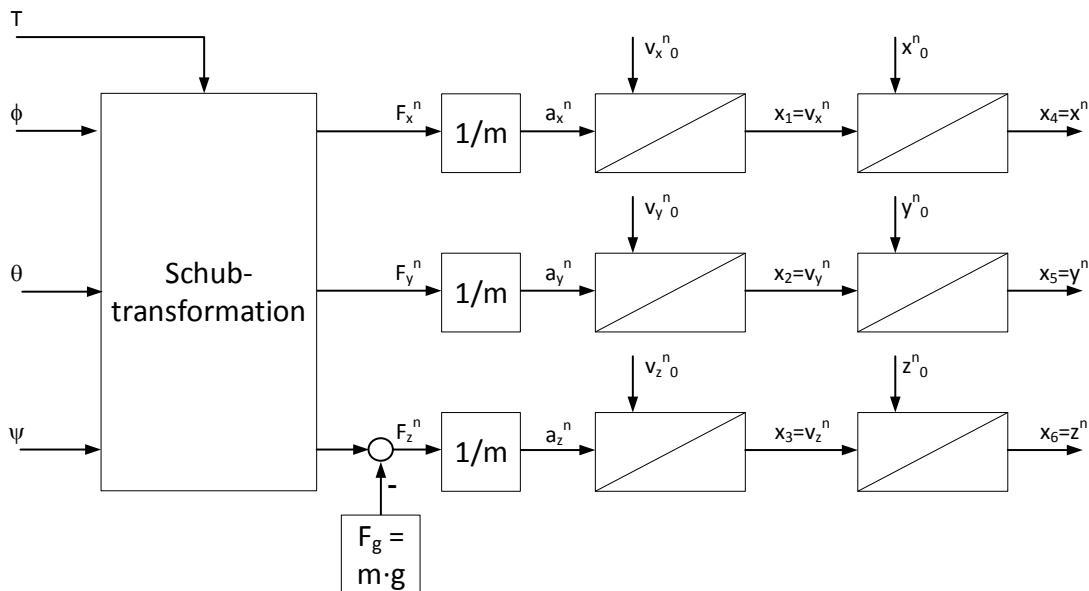
Mit diesen Gleichung 5.5 und 5.4 ist das translatorische Systemverhalten des Quadrocopters im n-frame mathematisch beschreibbar. Abhängig der Einganggrößen  $O_{des}$  und  $T_{des}$ .

Die resultierende Beschreibung der Zustände  $x = [v_x \ v_y \ v_z \ x \ y \ z]^T$  des Modells ergibt.

$$\begin{aligned}\dot{x}_1 &= \frac{1}{m} \cdot ((\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi) \cdot T) \\ \dot{x}_2 &= \frac{1}{m} \cdot ((\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi) \cdot T) \\ \dot{x}_3 &= \frac{1}{m} \cdot ((\cos \phi \cos \theta) \cdot T) - g \\ \dot{x}_4 &= x_1 \\ \dot{x}_5 &= x_2 \\ \dot{x}_6 &= x_3\end{aligned}\quad (5.6)$$

Zur Veranschaulichung ist das Modell zusätzlich in Abbildung visualisiert.

Durch die Koordinatentransformation bzw. die trigonometrischen Funktionen ist die Systembeschreibung nichtlinear. Das bedeutet, dass von der Änderung der Eingangsgrößen keine direkt proportionale Änderung der Ausgangsgröße ableitbar ist. Somit ist eine direkte Ansteuerung der Lageregelung über einen linearen Positionsregler nicht möglich ist. Da



**Abbildung 5.6:** Modellstruktur des Quadrocopters (FEHLER, GEWICHTSKRAFT FEHLT UND SUBTRANS ALS NICHTLINEAR MARKIEREN)

allerdings einen ein solche linearer Regler vorgesehen ist, benötigt man einen Baustein. Der für fiktive Eingänge das Systemverhalten linearisiert.

## 5.3 Exakte Zustandslinearisierung

Zur Linearisierung eines nichtlinearen Modells wird in den Grundlagen der Regelungstechnik [12] die Arbeitspunktlinearisierung gelehrt. Nachteil eine solche Linearisierung besitzt oft nur für eine kleinen Umgebung um den Arbeitspunkt Gültigkeit. Um mit dem Quadrocopter schnelle Positionswechsel vollziehen zu können, werden jedoch hohe Stellwinkel benötigt. Somit ist eine annähernde Linearisierung für einen Arbeitspunkt nicht ausreichend. Es wird daher eine Methode benötigt, die das Modell über den gesamten Arbeitsbereich linearisiert.

In [5] wird dafür die exakte Zustandslinearisierung eingeführt. Dabei wird dem nichtlinearen System eine Inversion vorgeschaltet. Diese beinhaltet eine linearisierendes Stellgesetz so, dass sich das zu regelnde System für jeden Ausgang als entkoppelt und lineare Integrierekette darstellt.

Das bedeutet für das Quadrocoptermodell (Kapitel 5.2). Es ist ein Stellgesetz für  $u = [T \ \phi \ \theta \ \psi]^T$  gesucht, welches das System für die Ausgänge  $y = [x \ y \ z]^T$  entkoppelt. Für die Positionsregelung steht so die fiktiven Eingänge  $u_f = [a_x^n \ a_y^n \ a_z^n]^T$  zur Verfügung. Das bedeutet es können ohne Berücksichtigung der Orientierung des Quadrocopters Beschleunigungswerte im n-frame vorgegeben werden.

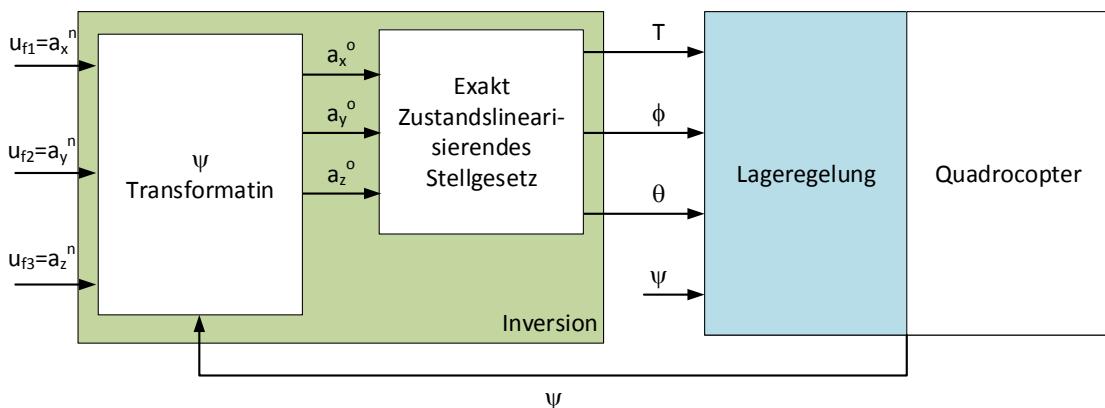


Abbildung 5.7: Gesamtinversion

Damit eine solche Inversion für die Position als Ausgang  $y = [x \ y \ z]$  möglich ist, muss es sich bei diesem Ausgang um einen so genannten flachen Ausgang  $y_f = [y_{f1} \ y_{f2} \ y_{f3}]$  handeln.

Per Definition (Anhang B) muss dafür das Kriterium, Anzahl der flachen Ausgänge  $p_y$  entspricht der Anzahl der Eingänge  $p_u$  des nichtlinearen Systems, erfüllt sein. Dieses ist für  $y_f = [x \ y \ z]$  und die Eingangsgrößen  $u = [T \ \phi \ \theta \ \psi]$  verletzt. Nach [5] heißt das, es muss ein neuer fiktiver Ausgang gesucht werden, sodass  $p_y = p_u$  gilt. Dieser neue Ausgang würde nicht mehr der Position entsprechen. Das bedeutet Positions vorgaben müssten erst in die neuen Ausgänge transformiert werden. Auch der gewünschte fiktive Eingang der  $u_f = [a_x^n \ a_y^n \ a_z^n]^T$  wäre nicht mehr gegeben.

Deshalb bedient sich die in Kooperation mit AscTec entwickelte Inversion [1] einem Trick.

Um die Position als flachen Ausgang zu erhalten, bedient man sich der Tatsache, dass eine Veränderung des  $\psi$ -Winkels zu keiner Auslenkung des Schubvektors  $T$  führt. So können die Beschleunigungsvorgaben  $u_f$  über eine einfache Transformation (Gleichung 3.1) um den Winkel  $\psi$  ins o-frame übertragen werden  $\tilde{u}_f = [a_x^o \ a_y^o \ a_z^o]^T$ . Gleichermaßen gilt auch für den zugehörigen Ausgang  $y_f$ . Somit ergibt sich für diesen  $\tilde{y}_f = [y_{f1}^o \ y_{f2}^o \ y_{f3}^o]$ . Durch diesen Trick reduziert sich die Anzahl der Eingänge der für Inversion um einen. So entspricht die Anzahl der verbliebenen Eingängen  $\tilde{u} = [\phi \ \theta \ T]$  denen des gewünschten flachen Ausgangs. Damit ist die Flachheit des Ausgangs  $\tilde{y}_f$  nicht erwiesen. Es fehlt per Definition (vgl. Anhang B) noch der Beweis, dass sich alle Zustände  $\tilde{x}$  (vgl. Abbildung 5.6) und die reduzierten Eingänge  $\tilde{u}$  durch die flachen Ausgänge  $\tilde{y}_f$  und dessen Ableitungen darstellen lassen. Für die Zustände ist dieser Beweis, ohne großen Rechenaufwand möglich.

$$\begin{aligned}
 \tilde{y}_{f1}^o &= x^o = x_4 \\
 \tilde{y}_{f2}^o &= y^o = x_5 \\
 \tilde{y}_{f3}^o &= z^o = x_6 \\
 \dot{\tilde{y}}_{f1}^o &= \dot{x}^o = x_1 \\
 \dot{\tilde{y}}_{f2}^o &= \dot{x}^o = x_2 \\
 \dot{\tilde{y}}_{f3}^o &= \dot{x}^o = x_3
 \end{aligned} \tag{5.7}$$

Für die Eingangsgrößen  $\tilde{u}$  müssen die Zustandsgleichungen für  $\dot{\tilde{x}}_1$ ,  $\dot{\tilde{x}}_2$  und  $\dot{\tilde{x}}_3$  (vgl. Gleichung 5.6) nach  $\phi$ ,  $\theta$  und  $T$  aufgelöst werden. Dabei entspricht  $\dot{\tilde{x}}_1 = a_x^o = \ddot{y}_{f1} = \tilde{u}_{f1}$ ,  $\dot{\tilde{x}}_2 = a_y^o = \ddot{y}_{f2} = \tilde{u}_{f2}$  sowie  $\dot{\tilde{x}}_3 = a_z^o = \ddot{y}_{f3} = \tilde{u}_{f3}$ . Damit kann der Schub  $T$  als Betrag der im o-frame wirkenden Kräfte, bzw. geforderten Beschleunigungen multipliziert mit der Masse dargestellt werden.

$$\begin{aligned} T &= m \cdot \sqrt{\ddot{y}_{f1}^2 + \ddot{y}_f^2 + (\ddot{y}_{f3} + g)^2} \\ &= m \cdot \sqrt{a_x^o{}^2 + a_y^o{}^2 + (a_z^o + g)^2} \end{aligned} \quad (5.8)$$

Nun da  $T$  bestimmt, ist unter Beachtung das  $\psi = 0$ , durch die vorgelagerte Transformation,  $\dot{\tilde{x}}_2$  aus Gleichung 5.6 nach  $\phi$  aufzulösen. Damit ergibt sich,

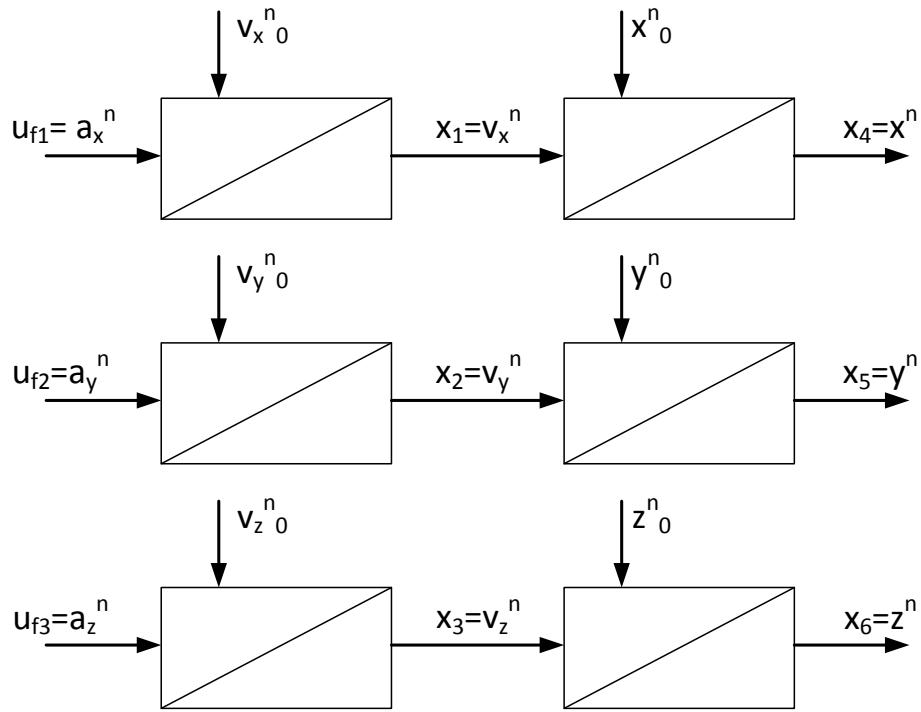
$$\begin{aligned} \phi &= -\arcsin\left(\frac{\ddot{y}_{f2} \cdot m}{T}\right) \\ &= -\arcsin\left(\frac{a_y^o \cdot m}{T}\right). \end{aligned} \quad (5.9)$$

$\theta$  lässt sich aufgrund der trigonometrischen Beziehung  $\tan\alpha = \frac{\sin\alpha}{\cos\alpha}$  aus den Zustandsgleichungen für  $\dot{\tilde{x}}_1$  und  $\dot{\tilde{x}}_3$  berechnen.

$$\begin{aligned} \theta &= \arctan\left(\frac{\ddot{y}_{f1}}{\ddot{y}_{f3} + g}\right) \\ &= \arctan\left(\frac{a_x^o}{a_z^o + g}\right) \end{aligned} \quad (5.10)$$

Die Stellgesetze der exakten Zustandslinearisierung sind damit über Gleichung (5.8), (5.9) und (5.10) mathematisch beschrieben. In Verbindung mit der Koordinatentransformation von  $\psi$  ergibt sich die in Abbildung 5.7 dargestellte Inversion.

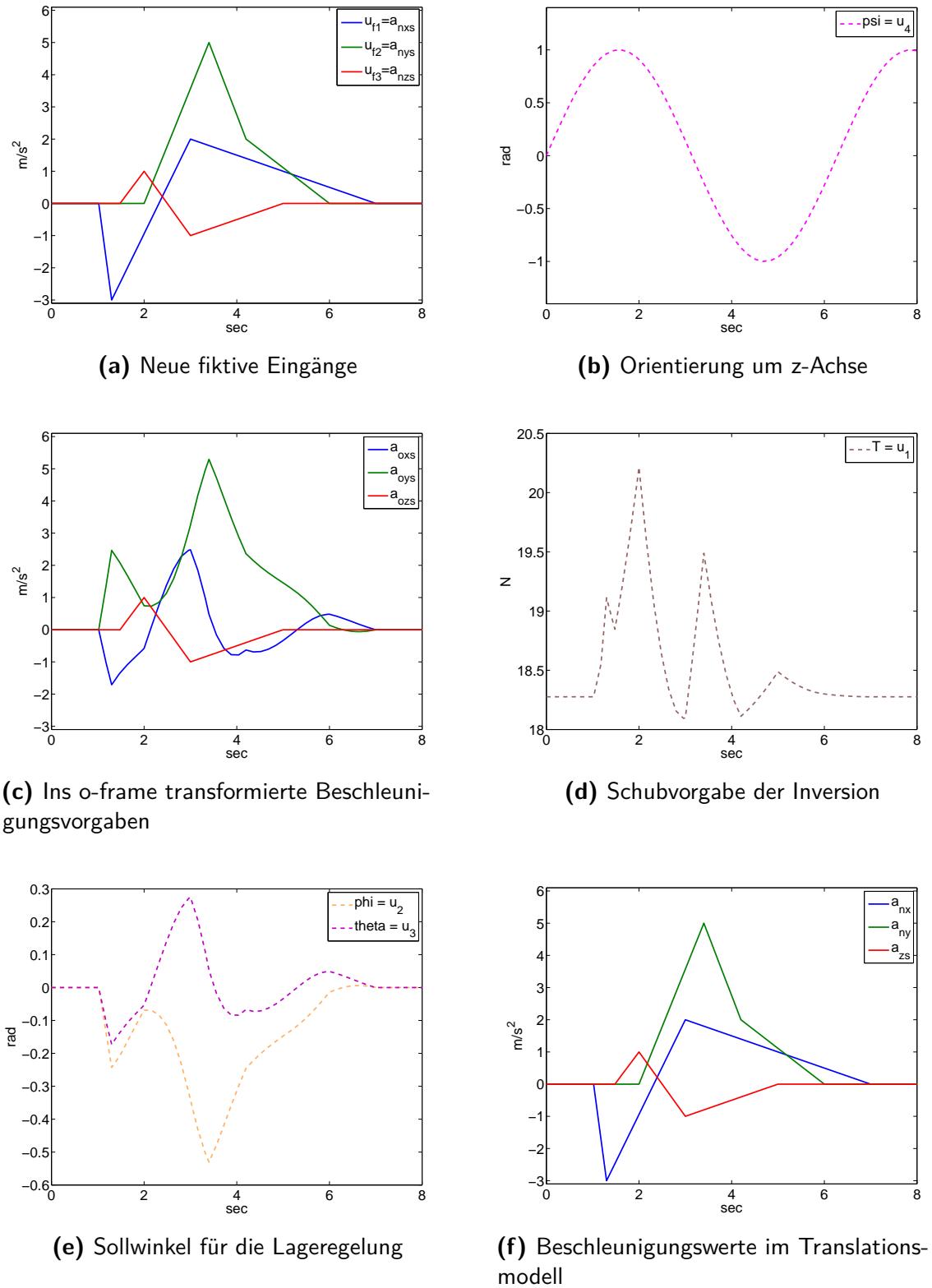
Die Richtigkeit dieser lässt sich über eine Simulation beweisen. Das Modell besteht dabei aus Inversion (Abbildung 5.7) und Translationsmodell (Abbildung 5.6). Über den neuen Eingang  $u_f$  werden dabei Beschleunigungswerte im n-frame vorgegeben (Abbildung 5.9a). Unter Einfluss ständig wechselnden Orientierung um die Hochachse (Abbildung ), ergeben sich die in Abbildung dargestellten Vorgaben für das Exaktzustandslinearisierten Stellgesetz. Die daraus resultierende Schubvorgabe und Winkel werden dem Bewegungsmodell



**Abbildung 5.8:** Modell aus Sicht der Positionsregelung dank Inversion

übergeben. Mit dem Ergebnis, das die resultierende Beschleunigungen des Modells denn Beschleunigungsvorgaben entspricht.

Dadurch kann nun das zu regelnde System als lineares Modell bestehend aus drei entkoppelten Integrierketten (Abbildung 5.8) gesehen werden. Dieses System ist jedoch instabil. Das bedeutet, für Vorgabe einer konstanten Beschleunigung, schwingt der Quadrocopter auf keiner konstanten Position ein, sondern Integriert seine Position immer weiter auf. Begründet ist dieses Verhalten auch durch das Weg-Zeit-Gesetz (Gleichung 5.5). Möglich ist es jedoch einen Beschleunigungsverlauf vorzugeben, für den der Quadrocopter in eine neue Position überführt wird. Dafür benötigt wird eine Vorsteuerung.



**Abbildung 5.9:** Simulation der Inversion

## 5.4 Vorsteuerung/Referenzmodell

Eine Vorsteuerung erzeugt anhand einer Sollwertvorgabe einen Stellgrößenverlauf, der benötigt wird um System im störungsfreien Fall in diesen Zustand zu überführen. In [1] wird deshalb auch von Referenzmodell gesprochen.

Im Sachen Positionsregelung bedeutet dies, es wird eine anzufliegende Position  $P_{cmd}^n$  übergeben. Daraus generiert die Vorsteuerung eine Trajektorie. Der Pfad  $P_{ref}^n$  mit dem der Quadrocopter in den Zustand überführt werden soll muss dabei einmal stetig differenzierbar sein. Der Grund, die Stellgröße der Inversion sind Beschleunigungsvorgaben. Demzufolge muss die zweite Ableitung der Position mindestens stückweise existieren. Damit dies gewährleistet ist, ist nach [5] folgende Differenzialgleichung für das Referenzmodell aufzustellen.

$$\ddot{P}_{ref}^n + c_1 \cdot \dot{P}_{ref}^n + c_0 \cdot P_{ref}^n = c_0 \cdot P_{cmd}^n \quad (5.11)$$

Hierbei kann die Dynamik mit der das Fluggerät überführt werden soll, über die Koeffizienten eingestellt werden. Stellt man für (5.11) die Übertragungsfunktion  $G(s)$  auf.

$$\begin{aligned} G(s) &= \frac{P_{cmd}^n}{P_{ref}^n} = \frac{c_0}{s^2 + c_1 \cdot s + c_0} \\ &= \frac{1}{\frac{1}{c_0}s^2 + \frac{c_1}{c_0} \cdot s + 1} \end{aligned} \quad (5.12)$$

Kann man erkennen das diese in ihrer Form einem PT2-Glied entspricht.

$$G_{PT2}(s) = \frac{1}{(\frac{1}{\omega_0})^2 s^2 + \frac{2D}{\omega_0} \cdot s + 1} \quad (5.13)$$

Setzt man (5.13) mit (5.12) gleich kann man die Koeffizienten  $c_0$  und  $c_1$  abhängig der gewünschten Eigenfrequenz  $\omega_0$  und Dämpfung  $D$  wählen.

$$c_0 = \omega_0^2 \quad (5.14)$$

$$c_1 = 2D\omega_0 \quad (5.15)$$

Der Vorteil, die Dynamik der Vorsteuerung ist über die in den Grundlagen der Regelungstechnik vermittelten Entwurfskriterien für PT2-Glieder einstellbar. So lässt sich über  $\omega$

einstellen wir schnell die Zielkoordinate erreicht werden soll. Das wie, lässt sich über die Dämpfung  $D$  bestimmen. So wird für alle  $D \leq 1$  eine Pfad ohne Überschwinger generiert. Die Dynamikvorgabe ist jedoch durch die maximal zulässige Stellgröße beschränkt. So kann zumindest  $\omega$  nicht beliebig Groß gewählt werden.

Der Stellgrößenverlauf  $u_{f_{ref}} = \ddot{P}_{ref}^n$  ist somit über (5.11) mit (5.14) und (5.15) folgendermaßen festgelegt.

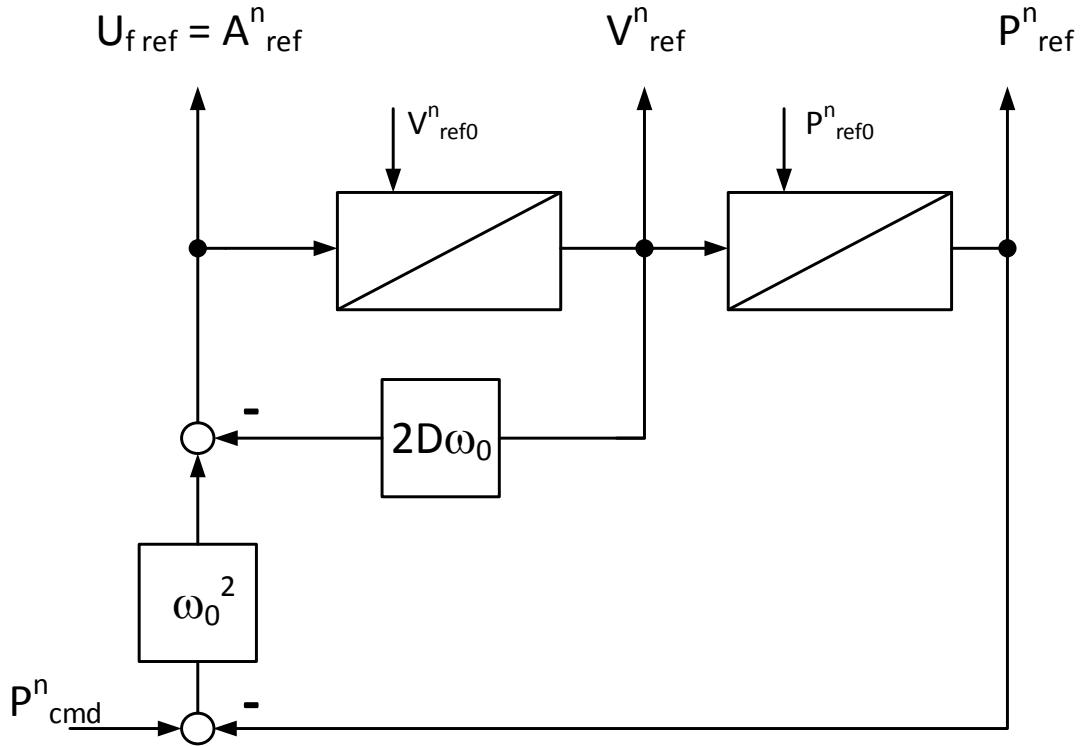
$$u_{f_{ref}} = \omega_0^2 \cdot (P_{cmd}^n - P_{ref}^n) - 2D\omega_0 \cdot \dot{P}_{ref}^n \quad (5.16)$$

Das resultierende Strukturbild der Vorsteuerung ist in Abbildung 5.10 visualisiert. Für ein Positionsverschiebung von  $1 \text{ m}$  erzeugt das Referenzmodell die in Abbildung 5.11 Trajektorie bzw. den in Grafik 5.11c dargestellten Stellwertverlauf. Hier ist anzumerken, das aufgrund der entkoppelten Integriererketten (Kapitel 5.3) eine separate Dynamikvorgabe für jeden Zweig des zustandslinearisierten Modells möglich ist. Weiterhin besteht die Möglichkeit die Vorsteuerung auszubauen, so dass Geschwindigkeitsvorgaben gemacht werden können. Dies erleichtert unter anderem das manuelle Navigieren mittels einer Fernsteuerung. Dabei erfolgt die Vorgabe der Soll-Geschwindigkeiten  $v_{cmd}^o$  im o-frame. Diese Vorgabe wird ins n-frame transformiert und dort über die Zeit integriert. Somit ergibt sich eine stetig Positions vorgabe für  $P_{cmd}^n$ .

$$P_{cmd}^n = \int (M_z^T \cdot v_{cmd}^o) dt + P_{cmd_0}^n \quad (5.17)$$

Mit dieser wird das Referenzmodell (5.16) gespießt. Es wird ein Stellgröße erzeugt für das die Position des Quadrocopter der stetigen Positions vorgabe (5.17) folgt. Die maximale Geschwindigkeit ist deshalb durch die Grenzfrequenz  $\omega_0$  des Referenzmodells begrenzt. Je größer  $\omega_0$  die desto höher die mögliche Geschwindigkeit.

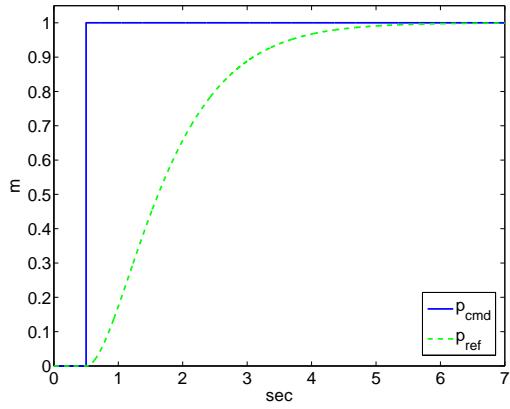
Die Vorsteuerung ist somit für alle Anwendungszwecke entworfen. Speist man das exakt Zustandslinearisierte mit dem in Abbildung 5.11 generierten Sollwertverlauf so folgt der Quadrocopter der Trajektorie. Dies gilt allerdings nur für den Fall konsistenter Anfangszustände (Abbildung 5.12a). Ist dies nicht gegeben, ist wie in Abbildung zu erkennen das Positionsverlauf des Translationsmodell für den gleichen Stellgrößenverlauf nicht der Referenz entspricht. Das gleiche gilt für Unsicherheiten der Modellparameter (Abbildung 5.12c). Hierbei berechnet die Inversion einen unzureichenden oder überdimensionierten Schubvektor.

**Abbildung 5.10:** Strukturbild Vorsteuerung

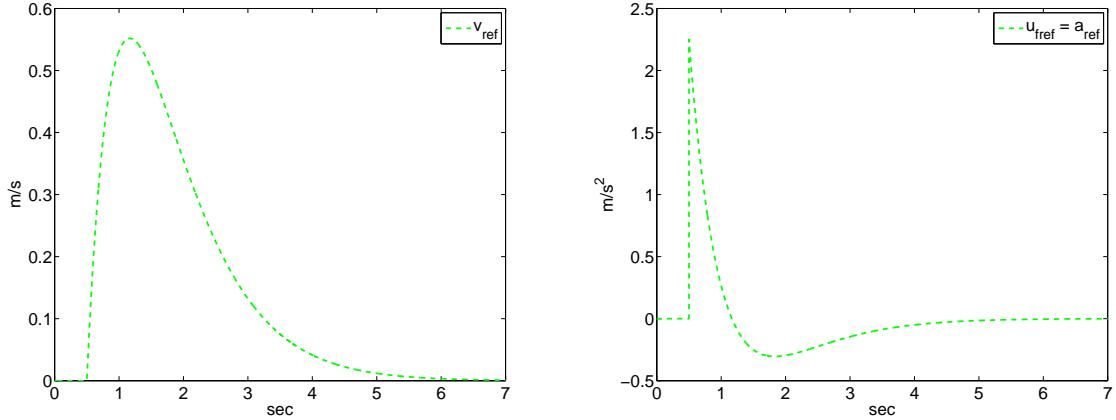
Außerdem können äußere Krafteinwirkungen wie zum Beispiel Winde wirken, die den Quadrocopter von der Trajektorie auslenken. Um all diesen Effekten entgegenzuwirken benötigt man einen Folgeregler. Dessen Aufgabe besteht darin diese Einflüsse zu eliminieren, sodass das Modellverlauf auch im gestörten Fall der Trajektorie entspricht.

## 5.5 Folgeregler

Inkonsistente Anfangsbedingungen, externe Störungen sowie Modellunsicherheiten bewirken einen Ausgangsfolgefehler  $e = P_{\text{ref}}^n - P^n$ . Dieser pflanzt sich weiter fort, da es sich bei dem Stellsignal  $u_{f,\text{ref}}$  (5.16) nicht um das Stellsignal  $u_f$  handelt welches den Quadrocopter zurück auf die Trajektorie führt. Dies gilt auch im Falle eines unzureichenden Stellsignalverlaufs in Folge von Modellunsicherheiten. Modellieren lässt sich die Fortpflanzung des Folgefehlers aufgrund des linearisierten Modells in Form einer doppelten Integriererkette. Zweifache Integriererkette ergo instabiles System. Damit später die Flugbahn des Quadrocopter mit der



(a) Positionsvorgabe über  $p_{cmd}$  und generierte Positionsvorgabe  $p_{ref}$



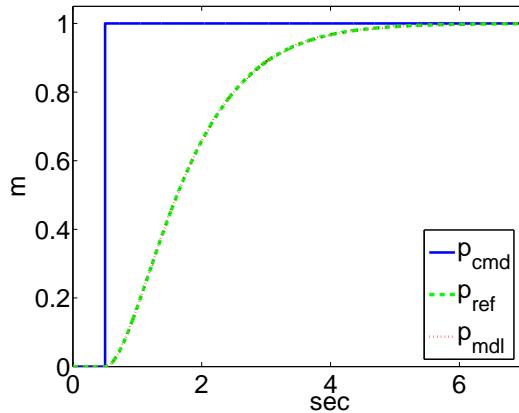
(b) Geschwindigkeitsvorgabe  $v_{ref} = \dot{p}_{ref}$  der Trajektorie

(c) Beschleunigungsvorgabe  $u_{fref} = a_{ref} = \ddot{p}_{ref}$

**Abbildung 5.11:** Simulationsergebnis des Referenzmodells für eine Achse mit  $\omega_0 = 1.5 \text{ Hz}$   $D = 1$

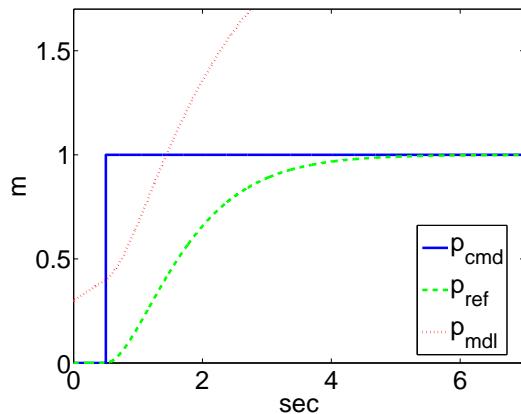
Solltrajektorie konvergiert, muss die Fehlerdynamik stabilisiert werden. Zu diesem Zwecke werden die Zustandsgrößen des Ausgangsfolgefehlermodells zurückgeführt (Abbildung 5.13). Daraus ergibt sich für die Vorgabe, dass der Folgefehler  $e$  in einer endlichen Zeit gegen 0 strebt, folgende Differentialgleichung.

$$\ddot{e} + \tilde{c}_1 \cdot \dot{e} + \tilde{c}_0 \cdot e = 0 \quad (5.18)$$



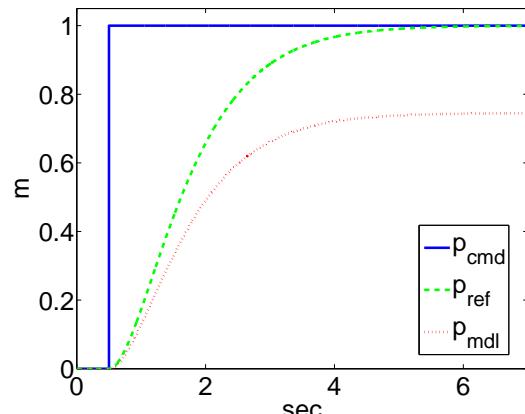
(a) Konsistente Anfangszustände

$p_{ref0} = p_{mdl0} = 0$  und  
 $v_{ref0} = v_{mdl0} = 0$ , ideales Modell



(b) Inkonsistente Anfangszustände

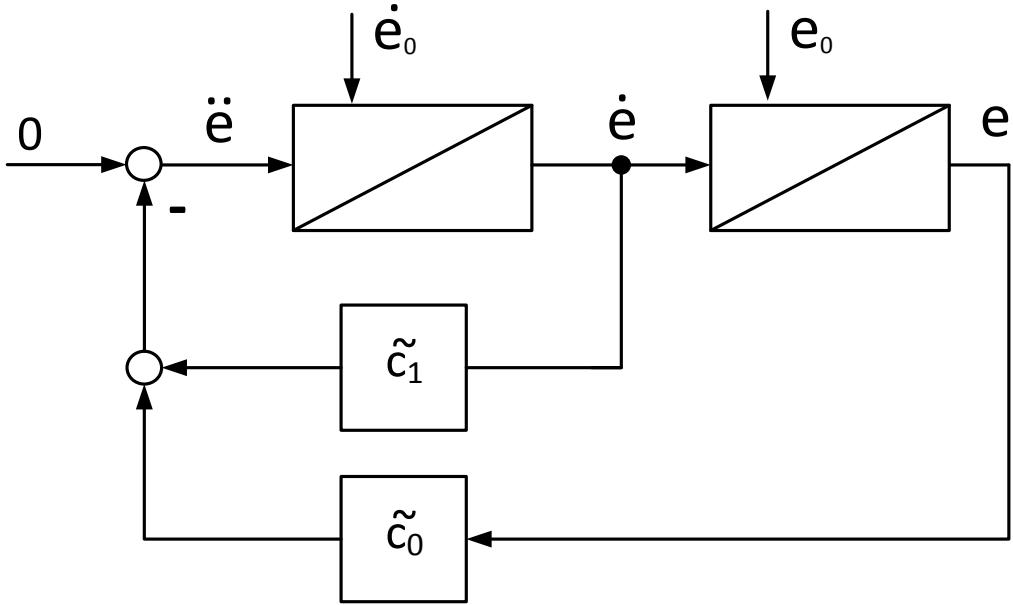
$p_{ref0} = 0 \neq p_{mdl0} = 0.3$  und  
 $v_{ref0} = 0 \neq v_{mdl0} = 0.2$ , ideales  
Modell



(c) Unsicherheit der Modellparameter

$m_{inv} \neq m_{mdl}$

Abbildung 5.12: Steuerung der Regelstrecke über das Referenzmodell



**Abbildung 5.13:** Ausgangsfolgefehlermodell mit Zustandsrückführung

mit

$$\begin{aligned}
 e &= P_{ref}^n - P^n \\
 \dot{e} &= \dot{P}_{ref}^n - \dot{P}^n \\
 \ddot{e} &= u_{f_{ref}} - u_f
 \end{aligned} \tag{5.19}$$

Diese kann nach  $u_f$  um gestellt werden. Damit ergibt sich das nachfolgende Stellgesetz für die Eingangsgröße  $u_f$  des zustandslinearsierenden Systems welches den Folgefehler ausregelt.

$$u_f = \underbrace{u_{f_{ref}}}_{\text{Vorsteuerung}} + \tilde{c}_1 \cdot (\dot{P}_{ref}^n - \dot{P}^n) + \tilde{c}_0 \cdot (P_{ref}^n - P^n) \tag{5.20}$$

Die Annäherungsverhalten ist Abhängig von den Koeffizienten  $\tilde{c}_0$  und  $\tilde{c}_1$ . Anhand von 5.18 lassen sich für diese mittels der Polvorgabe (Kapitel 5.5.1) die Dynamik vorgegeben.

Nicht bekämpft werden durch dieses Stellgesetz konstante Dauerstörungen. Hervorgerufen zum Beispiel durch Winde die über einen längeren Zeitraum als Konstant anzusehen sind.

Mit einer konstanten Kraft wirken diese auf das Flugsystem. Beachtet man die von dieser Kraft hervorgerufenen Beschleunigungsbeitrag  $a_{st}$  in der Fehlerdifferenzialgleichung.

$$\ddot{e} + \tilde{c}_1 \cdot \dot{e} + \tilde{c}_0 \cdot e + a_{st} = 0 \quad (5.21)$$

Folgt im stationären Zustand, das heißt alle Zeitableitungen gleich Null, ein dauerhafter Positionsfehler.

$$e = -\frac{a_{st}}{\tilde{c}_0} \quad (5.22)$$

Lösung dieses Problems ist nach [5] die Erweiterung des Folgeregler mit einer Integrierenden Komponente.

$$\ddot{e} + \tilde{c}_1 \cdot \dot{e} + \tilde{c}_0 \cdot e + \tilde{c}_{-1} \int_{I-\text{Anteil}} e(\tau) d\tau + a_{st} = 0 \quad (5.23)$$

Der I-Anteil liefert einen Signalanteil zur Kompensation der Störung im stationären Zustand. Den Beweis dafür erhält man, wenn man die Integro-Differentialgleichung (5.23) nach der Zeit ableitet. Danach ergibt sich Folgende Differentialgleichung.

$$\ddot{\ddot{e}} + \tilde{c}_1 \cdot \ddot{e} + \tilde{c}_0 \cdot \dot{e} + \tilde{c}_{-1} \cdot e = 0 \quad (5.24)$$

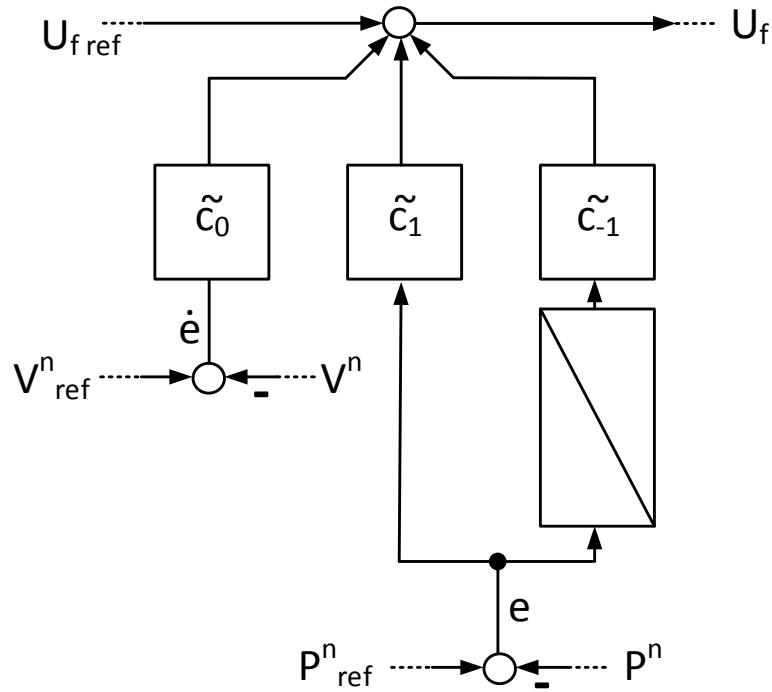
Im eingeschwungenen Zustand, ergibt sich so für den Positionsfehler

$$e = 0 \quad (5.25)$$

Das Stellgesetz für einen Regler mit I-Anteil lässt sich unter Vernachlässigung der Dauerstörung  $a_{st}$  aus Gleichung (5.23) entwickeln (vgl. Abbildung 5.14).

$$u_f = \underbrace{u_{f_{ref}}}_{\text{Vorsteuerung}} + \tilde{c}_1 \cdot (\dot{P}_{ref}^n - \dot{P}^n) + \tilde{c}_0 \cdot (P_{ref}^n - P^n) + \tilde{c}_{-1} \int_{Folgeregler \text{ inklusive } I-\text{Anteil}} e(\tau) d\tau \quad (5.26)$$

Dieses Stellgesetz führt den Quadrocopter entlang Referenztrajektorie. Zur erkennen ist dies auch in der Simulation (5.15) mit Regler Regler. So wird für Inkonsistene Anfangsbedingungen (Abbildung 5.15a) das Fluggerät auf die Trajektorie geführt. Für Modellunsicherheiten (Abbildung 5.15b) passt der Folgeregler das Stellsignal so an, das der Quadrocopter entlang des vorgegebenen Pfad fliegt. Vergleich hierzu reine Vorsteuerung (Abbildung 5.12).

**Abbildung 5.14:** Folgeregler

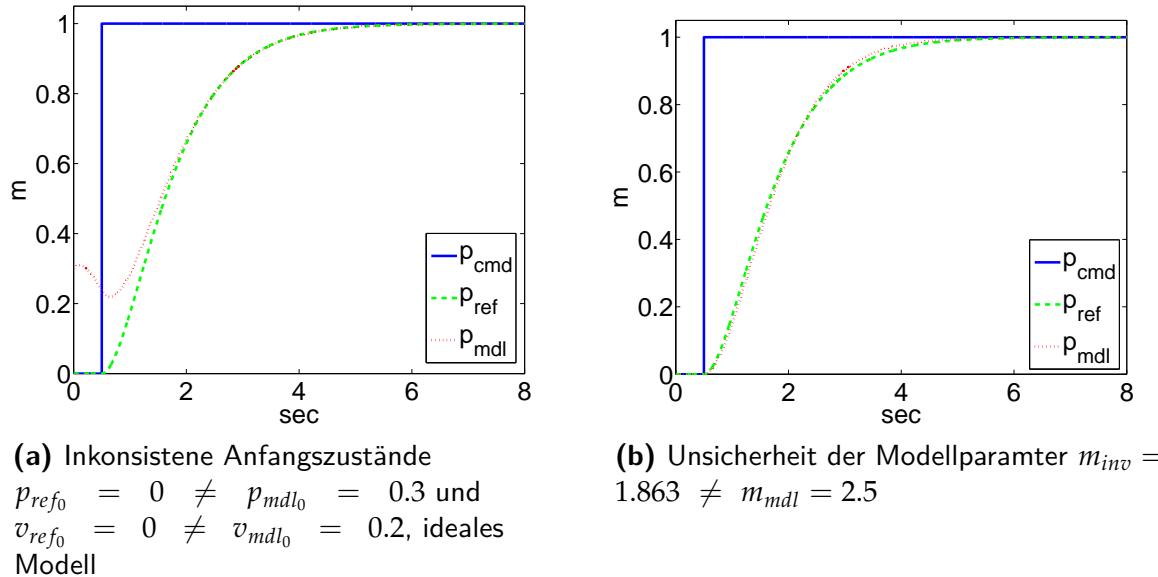
Wie auch schon zuvor kann das Einschwingverhalten für den Folgeregler mit I-Anteil über die Polvorgabe festgelegt werden. Für diesen ist die Differentialgleichung (5.24) zu verwenden.

### 5.5.1 Einstellung der Dynamik mittels Polvorgabe

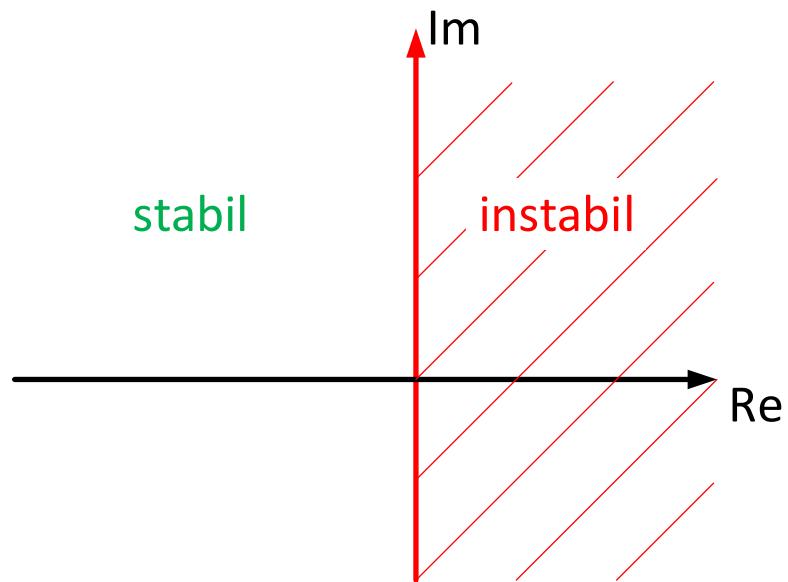
Anhand der Lage der Polstellen einer Übertragungsfunktion, können Rückschlüsse über die Stabilität und das Einschwingverhalten getroffen werden. Ziel einer Zustandsregelung, wie der Folgereregelung, ist die Pollagen der zu regelnden Strecke so zu verschieben, dass das Einausgangsverhalten die gewünschte Dynamik aufweist.

Zunächst ist zu klären was die Polstellenlage in der komplexen Ebene über das Verhalten aussagt. Dabei entspricht die Anzahl der Pole  $\lambda_i$  der Ordnung  $n$  des Systems.

- Befinden sich alle Pole  $\lambda_i$  links der Imaginärachse ( $\text{Re}(\lambda_i) < 0$ ), so ist das System asymptotisch stabil, ansonsten instabil (Abbildung 5.16).



**Abbildung 5.15:** Auswirkung von Folgeregler ( $\tilde{c}_1 = 4$ ,  $\tilde{c}_0 = 4$ ,  $\tilde{c}_{-1} = 0.01$ ) auf inkonsistente Anfangszustände und Modellunsicherheiten



**Abbildung 5.16:** Stabilitätsgebiet

- Komplexe Pole können nur in Formen von Polpaaren auftauchen. Ist ein Komplexes Polpaar vorhanden, führt das System nach Anregung eine Schwingung aus . Befindet sich die Polpaare in der linken Halbebene des Imaginärteils, nimmt die Amplitude der Schwingung exponentiell ab.
- Je weiter links sich die Pole auf der reellen Achse befinden, desto schneller ist das System.
- Bei mehreren Polstellen wird das Verhalten hauptsächlich über den Pole bzw. die Polpaare mit dem größten Realteil bestimmt. Sie werden deshalb als dominante Pole bezeichnet.

Betrachtet man eine beliebige Übertragungsfunktion im Bildbereich.

$$G_e(s) = \frac{Z(s)}{N(s)} \quad (5.27)$$

Entsprechen die Polstellen  $\lambda_i$  des Systems, denn Nullstellen  $s_i$  des Nennerpolynoms.

$$N(s) = s^n + c_{n-1} \cdot s^{n-1} + \cdots + c_1 \cdot s + c_0 = \prod_n^{i=1} (s - \lambda_i) = 0 \quad (5.28)$$

Das dynamische Verhalten kann anhand dieser Pollage analysiert werden. Im Umkehrschluss können für frei wählbare Koeffizienten des Nennerpolynoms die Dynamik über Vorgabe von Polstellen  $\lambda_{si}$  erfolgen. Dies erfolgt über einen Koeffizientenvergleich.

$$N(s) = s^n + c_{n-1} \cdot s^{n-1} + \cdots + c_1 \cdot s + c_0 = \prod_n^{i=1} (s - \lambda_{si}) \quad (5.29)$$

Somit ist es möglich über die Koeffizienten von (5.18) oder (5.24) das Einschwingverhalten des Folgereglers mittel der Platzierung von Polen vorzugeben. Wie jedoch die Polstellen optimal bestimmt, dafür gibt es keine generelle Vorgehensweise. In der Regel werden sie empirisch über Simulationen festgelegt. Anschließend am realen Modell getestet und gegebenenfalls optimiert. Diese Optimierung geschieht meist händisch. Allerdings gibt es hier Bemühungen diese letzte Optimierung am realen System automatisiert durchzuführen.

### 5.5.2 Automatische Optimierung der Reglerparameter

In der Simulation ermittelte Parameter müssen meist für das realen System leicht angepasst werden. Von Vorteil ist es deshalb, einen Algorithmus zu implementieren, der online und kontinuierlich die Einstellung optimiert. Die Bezeichnung dieses Vorgangs heißt selftuning.

Da der implementierte Folgeregler inklusive I-Anteil besteht aus einem Proportionalverstärker  $P$ , einem Differentiellen Anteil  $D$  sowie einem Integrationsanteil  $I$  zusammensetzt.

$$u_f = \underbrace{u_{f_{ref}}}_{\text{Vorsteuerung}} + \underbrace{\tilde{c}_1 \cdot (\dot{P}_{ref}^n - \dot{P}^n)}_{D-\text{Anteil}} + \underbrace{\tilde{c}_0 \cdot (P_{ref}^n - P^n)}_{P-\text{Anteil}} + \underbrace{\tilde{c}_{-1} \int e(\tau) d\tau}_{I-\text{Anteil}} \quad (5.30)$$

Können Tuningalgorithmen angewandt werden, die für einen PID-Regler entwickelt worden sind. So kann der unter [10] vorgestellte Algorithmus angewendet werden. Dieser tunt die Regelparameeter über eine adaptive Interaktion tunt. Die Interaktion besteht dabei aus drei Differentialgleichungen zur Anpassung der Koeffizienten des Reglers.

$$\begin{aligned} \dot{k}_p &= \dot{\tilde{c}}_0 = \gamma \cdot e^2 \\ \dot{k}_i &= \dot{\tilde{c}}_{-1} = \gamma \cdot e \cdot \int e(\tau) d\tau \\ \dot{k}_d &= \dot{\tilde{c}}_1 = \gamma \cdot e \cdot \dot{e} \end{aligned} \quad (5.31)$$

Dabei entspricht  $\gamma$  dem Anpassungskoeffizient. In [10] ist  $\gamma = 10$  empfohlen. Die Struktur des sich daraus ergebenden Folgeregler mit selbstoptimierung ist in Abbildung dargestellt. Anzumerken ist das die Qualität der Optimierung Abhängig von den zu Beginn übergebenen Parameter ist. Welche zuvor zum Beispiel über die Polvorgabe bestimmt worden sind.

## 5.6 Zustandsschätzung

Bei der Folgeregelung handelt es sich um einen Zustandsregler. Betrachtet man das Stellgesetz 5.26 bedeutet dies, das Position  $P^n$  und die Geschwindigkeit  $v^n$  mit der sich der Quadrocopter im n-frame bewegt müssen zur Verfügung stehen. Bereits bekannt ist, das die Position  $P^n$  mittels des Laser (Kapitel 4.2) bestimmt wird. Die Updatrate beträgt dabei nur 40Hz. Daher ist es nützlich Zwischenwerte zu schätzen, sodass die Positionsdaten dem Folgeregler in einer höheren Taktrate zur Verfügung stehen.

# PLATZHALTER GRAFIK

**Abbildung 5.17:** Folgeregler inklusive Selftuning

Der Zustand der Geschwindigkeit  $v^n$  ist messtechnisch sehr schwer zu ermitteln. Eine Möglichkeit zur messtechnischen Erfassung wäre die Integration eines Dopplerradars [17] auf dem Quadrocopter. Dieser basiert auf dem Doppler-Effekt. Dabei wird ein periodisches Signal ausgesendet und je nach Geschwindigkeit verkleinert sich Periodendauer, bzw. vergrößert sich für negative Geschwindigkeiten. So kann anhand der Frequenzänderung des reflektierten Signals auf die Geschwindigkeit geschlossen werden. Diese Umsetzung bedeutet die Integration eines weiteren Sensors. Aus Gewichts- und Kostengründen sind somit Methoden gefragt, die über die vorhandene Sensorik (IMU und Laser) die Geschwindigkeit des Quadrocopters feststellen.

## 5.6.1 Geschwindigkeitsbestimmung über die Inertialsenorik

Die *IMU* beinhaltet einen 3D-Bewegungsseensor. Dieser nimmt die auf Beschleunigung in x-, y- und z-Achse des Quadrocopters auf und stellt die Messwerte mit einer Frequenz von 1kHz ( $T_{imu} = 1ms$ ) zur Verfügung. Aufgrund des Weg-Zeit-Gesetzes (5.5) lässt sich die Geschwindigkeit über die Integration der Beschleunigung bestimmen. Für den diskreten Fall lässt sich diese mittels des Eulerverfahren approximieren. Zuvor müssen die Beschleunigungsmesswerte jedoch ins n-frame Transformiert (Gleichung (3.6)) werden.

$$v_{k+1}^n = v_k^n + T_{imu} \cdot (M_{nb} \cdot a_k^b) = v_k^n + T \cdot a_k^n \quad (5.32)$$

Somit ist die Geschwindigkeit bestimmt. In der Praxis führt diese Methode allerdings zu keinem guten Ergebnis. Grund dafür ist das Sensorrauschen  $r_{a_k}$  sowie der Bias  $b$ , der trotz Initialisierung nicht vollständig zu eliminiert ist. Somit ist der Messwert  $a_k^b$  nicht der tatsächlichen Beschleunigung  $a_{k_{tat}}^b$ .

$$a_k^b = a_{k_{tat}}^b + r_{a_k} + b \quad (5.33)$$

Das Problem, der Fehler wird mit integriert. Damit driften geschätzte und reelen Geschwindigkeit auseinander. Somit ist eine Geschwindigkeitsbestimmung ausschließlich über die Inertialsensorik nicht möglich.

### 5.6.2 Geschwindigkeit als Ableitung der Position

Ebenfalls auf Basis des Weg-Zeit-Gesetzes (5.5) lässt sich die Geschwindigkeit über die Ableitung der Quadrocopterpositionen bestimmen. Da es bei den Positionsdaten um diskrete Werte handelt, lässt sich die Ableitung anhand des Euler-Verfahren (5.34) approximieren.

$$P_k^n = P_{k-1}^n + T_l \cdot v_k^n \quad (5.34)$$

Für die Geschwindigkeit ergibt sich so

$$v_k^n = \frac{P_k^n - P_{k-1}^n}{T_l} \quad (5.35)$$

Die Abtastzeit  $T_l = 25ms$  ( $f_l = 40Hz$ ). Diese entspricht der Updaterate der Umgebungs-scans des Lasers mit denen die Algorithmen zur Positionsbestimmung gespeist werden. Bei der Approximation der Geschwindigkeit ist zu beachten, dass das Positionssignal aufgrund von Sensorrauschen von Laser und Gyroskop sowie der Varianz des *ICP*-Algorithmus eine Abweichung vom tatsächlichen Positions Wert  $P_{k_{tat}}^n$  aufweist. Diese Abweichung ist nicht konstant und ist als Positionsrauschen  $r_{P_k}$  darzustellen.

$$P_k^n = P_{k_{tat}}^n + r_{P_k} \quad (5.36)$$

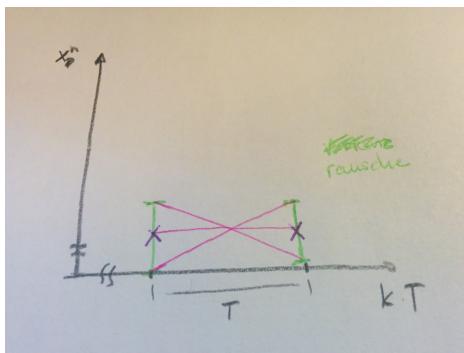
Dieses Positonsrauschen  $r_P$  überträgt sich auf die Geschwindigkeit.

$$v_k^n = \frac{P_k^n - P_{k-1}^n}{T_l} + \frac{r_{P_k} - r_{P_{k-1}}}{T_l} \quad (5.37)$$

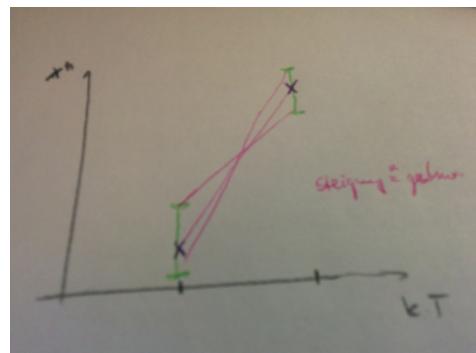
Diese führt jedoch nicht dazu, dass wie bei der Geschwindigkeitsbestimmung mittels der Beschleunigungssensoren, das Schätzwert und Realwert auseinander divergiert. Nichtsdestotrotz hat das Rauschen einen Einfluss auf die Qualität der Geschwindigkeitsberechnung. So fällt bei niedrigen Geschwindigkeiten der Abstand zwischen zwei Positionsgeraden geringer aus. Der Betrag des Rauschanteils jedoch bleibt konstant. Dies führt vor allem bei Schwebeflügen sowie Flügen mit geringer Dynamik zu einer erhöhten Unsicherheit der Schätzung (Abbildung 5.18a). Zwar zeigt Grafik 5.18b das der Einfluss des Rauschen auf die Varianz der errechneten Geschwindigkeit (5.37) mit der Größe der Positionsverschiebung abnimmt. Ungeachtet dessen ist es jedoch notwendig gerade für Bewegungen mit geringer Geschwindigkeit den Einfluss des Positionsrauschen zu minimieren. Geht man davon aus, dass die Frequenz des Rauschanteils oberhalb des Frequenzbades der Positionsänderungen liegt. Kann das Nutzsignal mittels eines Tiefpassfilters vom Rauschanteil getrennt werden. Das Tiefpassfilter lässt sich dabei mittels eines rekursiven *Infinite Impulse Response (IIR)*-Filters realisieren [6]. Die gefilterte Geschwindigkeit  $\hat{v}_k^n$  hängt dabei nicht nur von vorherigen Messwerten  $v_k^n$  sondern auch von den vorherigen Filterergebnissen.

$$\hat{v}_k^n = \sum_{j=0}^n a_j \cdot v_{k-j}^n + \sum_{i=1}^n b_i \cdot \hat{v}_{k-i}^n \quad (5.38)$$

Mittels der Koeffizienten  $a_j$  und  $b_i$  kann nun das gewünschte Filterverhalten, zum Beispiel das eines Butterworth-Filters, mit der benötigten Grenzfrequenz realisiert werden. Die



(a) Keine/kleine Positionsverschiebung, hohe Varianz



(b) Große Positionsverschiebung, geringe Varianz

**Abbildung 5.18:** Auswirkung der Größe der Positionsverschiebung innerhalb eines Schätzwertes auf die Varianz der Geschwindigkeitsschätzung

Ordnung  $n$  des Filter beschreibt wie viele der vorangegangenen Messwerten in die Filterung mit einzubeziehen sind. Beim Design des Filters müssen jedoch Kompromisse zwischen Zeitverzögerung, Phasenverzerrung, Dämpfung und Stoppbandeigenschaften getroffen werden. Für die Folgeregelung ist es dabei wichtig, dass die beiden ersten genannten Kriterien möglichst gering ausfallen. Ist dies nicht der Fall, wirkt es destabilisierend auf den Regler.

Des Weiteren lassen sich Aufgrund der geringen Updaterate von 40Hz der Positionsdaten, die Dynamik des Quadrocopters und das Positionsrauschen Spektral nicht von einander trennen. Möchte man also das Positionsrauschen unterdrücken, werden auch schnelle Positionsänderungen des Quadrocopters weg gefiltert. Für die hohe Dynamik des Quadrocopters ist die Anwendung eines Tiefpassfilter mit Grenzfrequenz nicht möglich. Gesucht ist nun eine Methode, die für Flüge mit geringer Dynamik die Varianz der Geschwindigkeitsschätzung verursacht durch Positionsrauschen möglichst stark verringert. Gleichzeitig jedoch der hohen Dynamik des Quadrocopters folgen leisten kann.

### 5.6.3 Geschwindigkeitsbestimmung über die Methode First-Order Adaptive Windowing

(Hier fehlt 1kHz und Position zwar höherer Abtastrate verfügbar, Fehler driftet durch doppelte Integration zu schnell zu weit weg) Wie schon in Kapitel 5.6.2 erkannt, nimmt die Varianz der Geschwindigkeitsschätzung ab je weiter die Positionsdaten auseinander liegen. Hierfür sei auf Grafik 5.18b verwiesen. Der gleiche Effekt tritt auf, je mehr Abtastschritte  $n$  der für die Eulerapproximation verwendete Bezugspunkt  $P_{k-n}^n$  in der Vergangenheit liegt.

$$\hat{v}_k^n = \frac{P_k^n - P_{k-n}^n}{nT_l} = \quad (5.39)$$

Dabei ist die Verwendung der Bezugsposition  $P_{k-n}^n$  gleichzusetzen mit der Mittlung der letzten  $n$  Geschwindigkeitsschätzungen  $(v_k^n, v_{k-1}^n, \dots, v_{k-n}^n)$  mittels (5.35). Deshalb spricht man auch von Fensterung beziehungsweise Windowing. Siehe zur Veranschaulichung Abbildung 5.19. Vergrößert man das Fenster hat das eine äquivalente Wirkung wie die Reduzierung der Abtastrate. Das Windowing verhält sich wie ein Filter. So führt ein großes Fenster bei geringen Dynamiken zu einer sehr präzisen Geschwindigkeitsschätzung durch Rauschunterdrückung. Schätzungen für Hochfrequenter Bewegungen des Quadrocopter werden jedoch stark gedämpft und Zeitverzögert ausgegeben, womit die Verlässlichkeit der Schätzung abnimmt. Es wirkt somit ähnlich wie das Filter (5.38). Allerdings mit dem Vorteil,

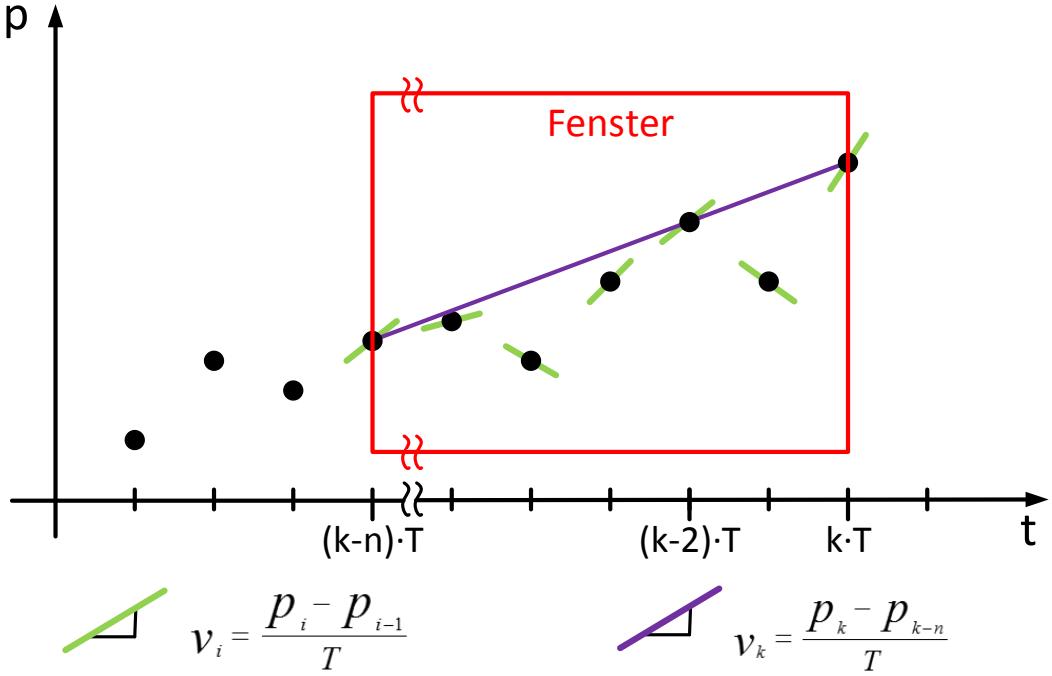
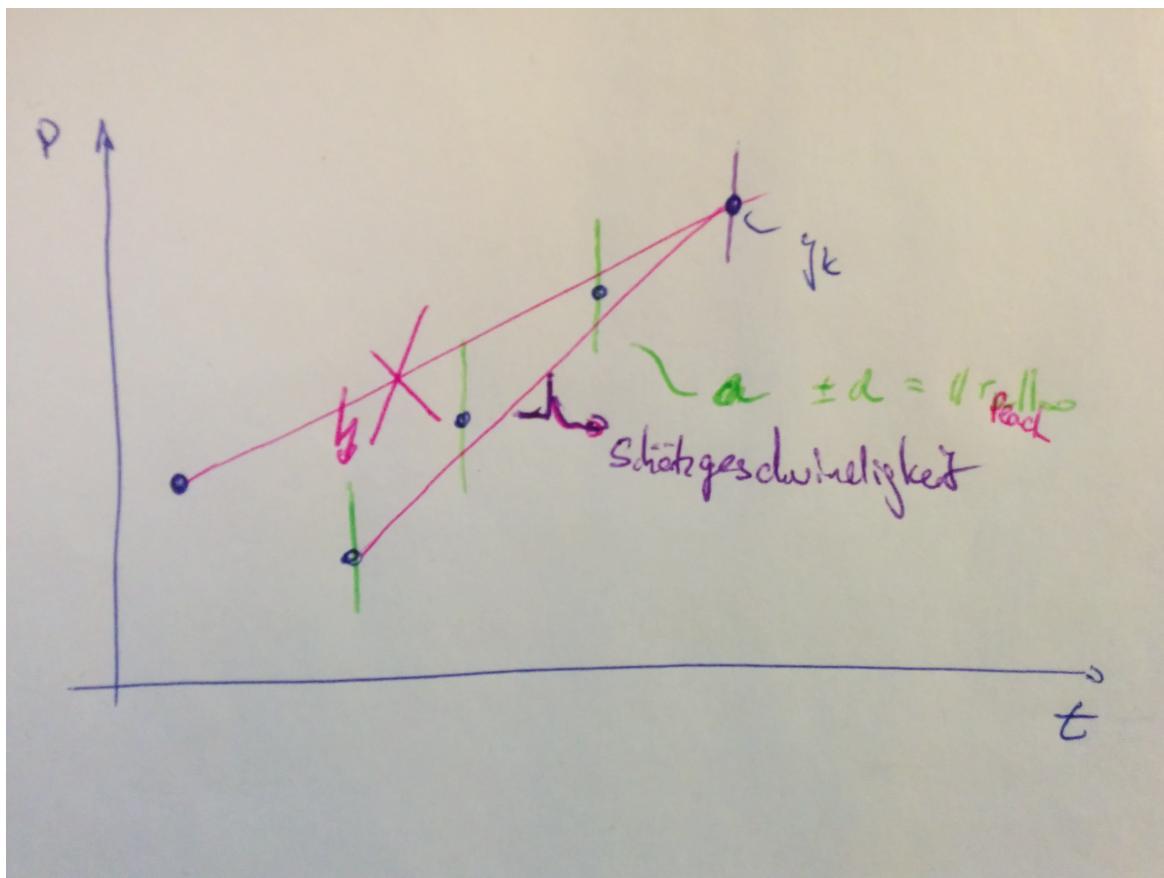


Abbildung 5.19: Fensterung

das Filterverhalten nur über eine Parameter einstellbar ist. Der Fenstergröße  $n$ . So sollte das Window klein sein, wenn der Quadrocopter Hochfrequente Änderungen der Bewegung vornimmt, damit die abrupten Geschwindigkeitsänderungen möglichst ungedämpft erfasst werden. Bei geringen Dynamiken soll die Fenstergröße jedoch zunehmen, wodurch der Einfluss des Positionsrauschens minimiert werden soll. Dabei muss die Anpassung der Fenstergröße online und für jeden Punkt Abhängig der Achse des n-frame separat erfolgen. Im Paper [8] wird dafür die Methode des *FOAW* vorgestellt. Diese Besagt, existiert eine Grade zwischen den Positionswerten  $p_k^n$  und  $p_{k-n}^n$  einer Achse, die alle  $n$  Punkte innerhalb einer Toleranz  $\pm d$  schneidet, stellt die Steigung dieser und somit die Geschwindigkeit einen optimalen Kompromiss zwischen Rauschunterdrückung (Präzision) und Verlässlichkeit dar. Die möglichen Schätzwerte für die Geschwindigkeit abhängig von der Fenstergröße  $n$  und der Toleranz  $d$  sind im folgenden Datensatz dargestellt.

$$\hat{v}_k \in \left[ \frac{p_k^n - p_{k-n}^n}{nT_l} - \frac{2d}{nT_l}, \frac{p_k^n - p_{k-n}^n}{nT_l} + \frac{2d}{nT_l} \right] \quad (5.40)$$



**Abbildung 5.20:** End-fit First-Order Adaptive Windowing (FOAW)

Die Toleranz  $d$  ist dabei durch den Betrag der maximalen Rauschabweichung festgelegt und somit als Konstante zu sehen.

$$d = ||r_{max}|| \quad (5.41)$$

Zu erkennen ist, dass die Varianz der Schätzung für eine steigende Fenstergröße  $n$  abnimmt. Somit ist ausgehend von der aktuellen Position  $p_k^n$  die maximale mögliche Fenstergröße  $n = \max\{1, 2, 3, \dots\}$  zu ermitteln für die gilt,

$$|p_{k-i}^n - \hat{p}_{k-i}^n| \leq d \quad \forall i \in \{1, 2, \dots, n\} \quad (5.42)$$

Dabei entspricht  $\hat{p}_{k-i}^n$  den Punkten auf der approximierenden Geradengleichung.

$$\hat{p}_{k-i}^n = a_n + b_n \cdot (k - i) T_l \quad (5.43)$$

In [8] ist dazu ein iterativer Algorithmus zur Lösung dieses Problems beschrieben. Dabei sind die Parameter der Geradengleichung (5.43) wie folgt parametriert.

$$a_n = \frac{k \cdot p_{k-i}^n + (n-k)p_k^n}{n} \quad (5.44)$$

$$b_n = \frac{p_k^n - p_{k-n}^n}{nT_l} \quad (5.45)$$

Da die Steigung  $b_n$  aus den zwei Rändern des Fensters gebildet wird, spricht man von der End-fit-FOAW.

Der Ablauf der des Iterationsalgorithmus sieht dann wie folgt aus.

- **Schritt 1:** Man verschiebt die Zeitachse so, dass für den Zeitpunkt des neusten Wert  $p_k^n$  gilt  $t_k = k \cdot T_l = 0$ . Die Folge ist eine vom Faktor  $k$  unabhängige Geradengleichung.

$$\hat{p}_{k-i}^n = a_n + b_n \cdot (-i) \cdot T_l \quad (5.46)$$

Für die gilt,

$$a_n = p_k^n \quad (5.47)$$

und  $b_n$  weiterhin über (5.44) berechnet werden kann. Vorteil, es muss jeweils nur die Steigung neu berechnet werden.

- **Schritt 2:** Fenstergröße  $j = 1$ .
- **Schritt 2:** Berechnung des Parameters  $b_j$  (5.45) der Geradengleichung (5.46) in Abhängigkeit der Fenstergröße  $j$ .
- **Schritt 3:** Überprüfen ob die berechnete Gerade alle im Fester befindlichen Punkte  $p_{k-i}^n, i \in \{1, 2, \dots, j\}$  innerhalb der Toleranz passiert. Bedingung (5.42)
- **Schritt 4:** Ist Schritt 3 erfüllt, inkrementiere die Fenstergröße  $j = j + 1$  und gehe zu Schritt 2. Wenn Bedingung nicht erfüllt, ist die Steigung der vorhergegangen Geradengleichung als Schätzwert der Geschwindigkeit auszugeben.

$$\hat{v}_k = b_{j-1} = b_n \quad (5.48)$$

Die maximale Fenstergröße  $n$  entspricht für Position  $p_k^n$  somit  $n = j - 1$

Dieser Vorgang wird für jeden Aktualisierung  $k = k + 1$  des Positions Wert  $p_k^n$  neu gestartet. Zur Veranschaulichung ist der Algorithmus noch einmal in einen Flussdiagramm (Abbildung 5.21) dargestellt.

Die End-fit-FOAW verwendet für den Schätzvorgang die zwei Endpunkte des Fensters. In [8] wird deshalb noch die Best-fit-FOAW vorgestellt. Dabei wird die Steigung, über die Methode der kleinsten quadratischen Fehler bestimmt. Das bedeutet für die zeitnormierte Geradengleichung ist eine Steigung  $b_n$  gesucht für die die Summe der Fehlerquadrat  $e^2$  ein Minimum aufweist.

$$e_{min}^2 = \min \sum_{i=0}^n (p_{k-i}^n - \hat{p}_{k-i}^n)^2 \quad (5.49)$$

mit

$$\hat{p}_{k-i}^n = p_k^n - b_n \cdot i \cdot T_l \quad (5.50)$$

Um das Minimum der Fehlergleichung (5.49) in Abhängigkeit der Steigung bestimmen zu können, muss diese nach  $b_n$  abzuleiten. Da es sich um eine quadratische Gleichung handelt, weist der Fehler Tiefpunkt auf wenn diese Ableitung Null ergibt.

$$\frac{de^2}{db_n} = 0 \quad (5.51)$$

# PLATZHALTER GRAFIK

**Abbildung 5.21:** Flussdiagramm FOAW

Löst man unter dieser Bedingung die Ableitung nach  $b_n$  auf. Erhält man die Steigung die die Summe der kleinsten quadratischen Fehler aufweist.

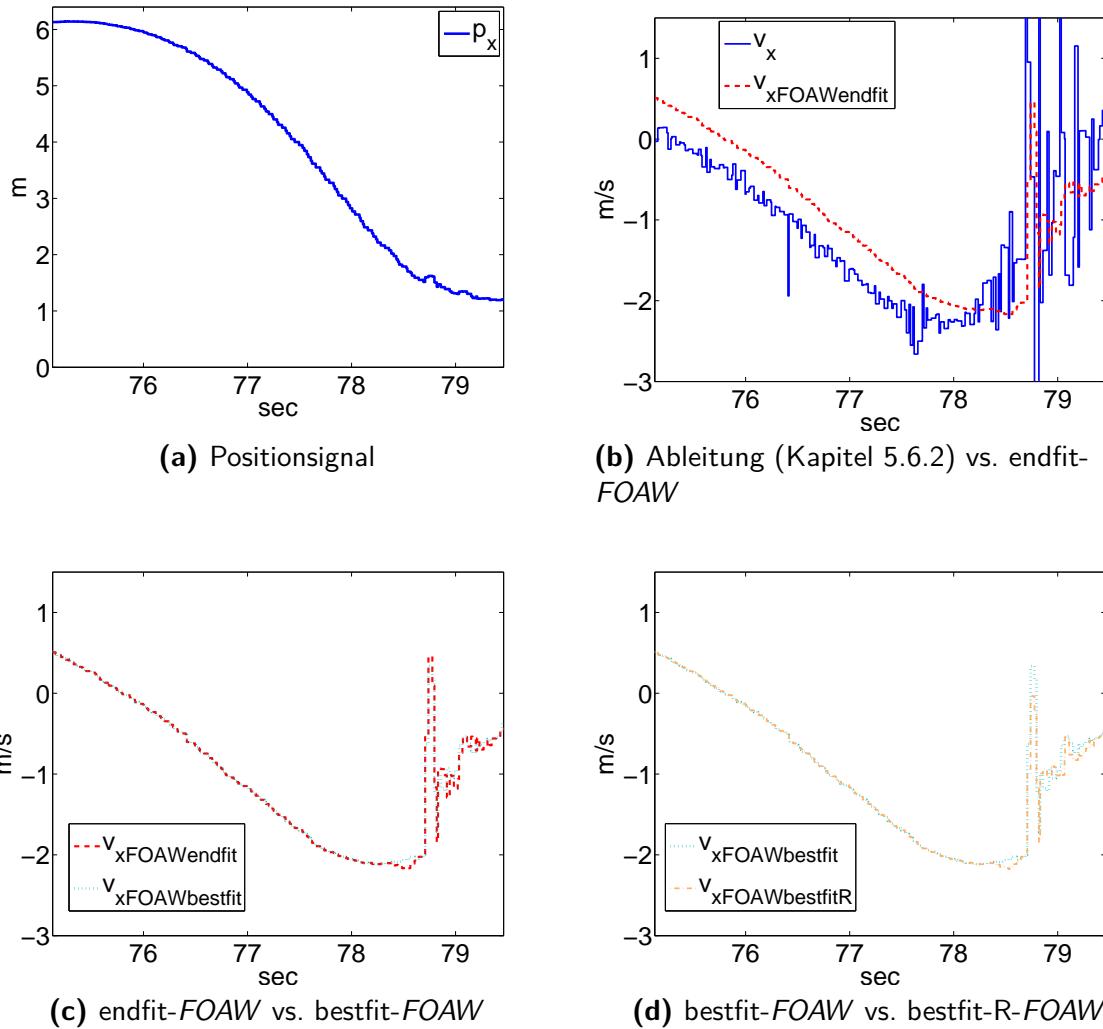
$$b_n = \frac{(n \sum_{i=0}^n y_{k-1} - 2 \sum_{i=0}^n i \cdot y_{k-1}) \cdot 6}{T_l n(n+1)(n+2)} \quad (5.52)$$

Nun ist die Steigung allgemeingültig für variable Fenstergrößen bestimmt. Die Umsetzung der best-fit-FOAW in der Software erfolgt nach dem gleichen Schema wie bei der end-fit-FOAW. Einziger Unterschied, zur Berechnung der Steigung  $b_n$  wir die Gleichung verwendet. Das Ergebnis ist eine präzisere Schätzung bei gleichbleibender Zuverlässigkeit der Messung. Beide Methoden sind jedoch anfällig gegen sogenannte Outliner. Einzelne Signalwerte die eine höheren Fehler als die maximale durch das Rauschen verursachte Abweichung  $d$  aufweisen. Outliner lassen sich jedoch sehr einfach durch eine robuster Abbruchkriterium in Schritt 3 bekämpfen. So gilt die Bedingung (5.42) erst dann für nicht bestimmt, wenn zwei aufeinanderfolgende Werte das Kriterium nicht erfüllen. Erst dann wird die Vergrößerung des Fensters gestoppt. Diese Variante nennt sich best-fit-FOAW-R.

Die Methoden des FOAW bietet eine sehr Möglichkeit die Geschwindigkeit auf Basis eines verrauschten Positionssignal zu ermitteln. Problematisch ist jedoch die geringe Abtastrate von  $40Hz$  des Lasers. Schon bei Flügen bei mittlerer Dynamik führt dies dazu, das die Fenstergröße stark reduziert wird. Somit auch der Einfluss des Positionsrauschens stark zunimmt. Somit kann selbst diese Konzept limitiert durch die Tastrate des Lasers der Dynamik des Quadrocopters nicht nachkommen. Es ist somit klar das eine Methode benötigt wird, die Position und Geschwindigkeit in einer höhren Taktrate zur Verfügung stellt. Gleichzeit müssen beide Signale ausreichend Rauschunterdrückt sein.

#### 5.6.4 Bestimmung der Zustände mittels eines Fusionsfilter

Wie aus den oberen Kapitel zuerkenne wird keine der Methoden den zuvor gestellten Ansprüche gerecht. So erfüllt die Integration der Beschleunigung (Kapitel 5.6.1) zwar das Kriterium zur Erhöhung der Taktrate. Beschleunigungsmesswerte werden in einem Takt von  $1kHz$  zur Verfügung gestellt. Doch durch den Bias des Beschleunigungssensors, divergieren innerhalb kürzester Zeit die geschätzten Zustände von den realen Zuständen. Dieses Phänomen tritt bei den Anwendungen Kaptiel und Kapitel nicht auf. Der Grund, sie beziehen sich auf die vom Laser berechneten Positionsverwerte. Die Abweichungen dieser



**Abbildung 5.22:** Vergleich der Filterergebnisse für FOAW

Zustandswerte befinden sich dabei in einem konstanten Rahmen um die tatsächliche Position. Diese Positionsrauschen sorgt bei der Geschwindigkeitsschätzung über die Ableitung der Positionswerte dafür, dass das errechnete Geschwindigkeitssignal ebenfalls einen Rauschanteil aufweist. Diesen Rauschanteil zu minimieren ist Ziel der FOAW Methode. Aufgrund der geringen Abtastrate von  $40\text{Hz}$  kann dieser Ansatz der hohen Dynamik nicht folgen. Schon für geringen Dynamiken ist die Fenstergröße stark dezimiert und somit auch die Rauschunterdrückung. Das Ergebnis, Bechleunigungssensor der *IMU* und Laser jeweils für sich bieten keine ausreichende Zustandswerte. Abhilfe, Fusion beider Sensordaten mittels eines Fusionsfilters. Dabei sorgen die Beschleunigungswerte mit ihrer hohen update Rate von

1Khz für eine schnelle Reaktionsfähigkeit und tragen somit der Dynamik des Quadrocopters sorge. Die Positionsdaten des Laser indes verhindern ein Abdriften der Schätzwerte. Somit stellen sie die Zuverlässigkeit der Fusionsergebnisse sicher.

Das auf dem *HLP* implementierte Fusionsfilter basiert auf einem Luenberger Beobachter[1]. Dieser besteht aus einen linearen Streckenmodell dessen Zustände  $\hat{x}$  mittels eines Korrekturterms ( $L \cdot (y - \hat{y})$ ) auf die Zustände  $x$  der Regelstrecke einschwingen. Die Zustandsdifferenzialgleichung eines Luenberger Beobachter stellt sich in allgemeiner Form [15] wie folgt dar.

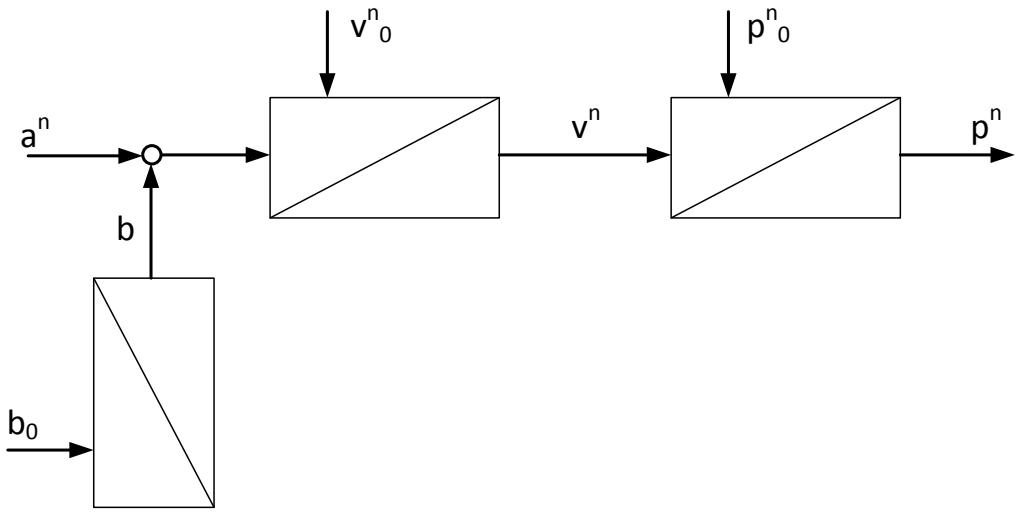
$$\begin{aligned}\dot{\hat{x}} &= A \cdot \hat{x} + L \cdot (y - \hat{y}) + B \cdot u \\ \hat{y} &= C \cdot \hat{x}\end{aligned}\tag{5.53}$$

Dabei entspricht  $A$  der Dynamikmatrix,  $B$  dem Eingangsmatrix und  $C$  dem Ausgangsmatrix des linearen Streckenmodells. Aufgrund der Inversion, ist das Bewegungsmodell für jede Achse im n-frame, als Integrationsglied 2.Ordnung gegeben (Abbildung 5.8). Dabei entspricht die Eingangsgrößen  $u$  den ins n-frame transformierten Beschleunigungswerten der *IMU*. Da die Ketten entkoppelt sind, muss der Beobachter nicht für alle drei Ketten gleichzeitig entworfen werden. Somit ist es ausreichen eine Integrationsglied 2.Ordnung als Streckenmodell heranzuziehen. Weiterhin ist bekannt, das die Beschleunigungswerte der *IMU* einen Bias aufweisen. Nach [4] kann das Streckenmodell deshalb wie in Abbildung ?? erweitert werden. Dieses erweiterte Modell ist über Zustandsdifferenzialgleichung (5.54) beschreiben

$$\begin{aligned}\dot{x} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot x + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot u \\ y &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot x\end{aligned}\tag{5.54}$$

mit,

$$\hat{x} = [p^n \ v^n \ b]^T\tag{5.55}$$


**Abbildung 5.23:** Erweitertes Streckenmodell

Anhand von (5.53) folgt daraus für den Beobachter.

$$\begin{aligned}\dot{\hat{x}} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot \hat{x} + \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} \cdot (y - \hat{y}) + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot u \\ \hat{y} &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \cdot \hat{x}\end{aligned}\quad (5.56)$$

mit,

$$\hat{x} = [\hat{p}^n \ \hat{\vartheta}^n \ \hat{b}]^T \quad (5.57)$$

Die Elemente der Beobachtermatrix  $L$  sind so zu parametrieren das die Fehlerdynamik das gewünschte Einschwingverhalten aufweist. Da der Zustandsfehler  $\tilde{x} = x - \hat{x}$  gilt, ergibt sich für die Dynamik des Zustandsfehlers

$$\dot{\tilde{x}} = \dot{x} - \dot{\hat{x}} = (A - LC) \cdot \tilde{x} \quad (5.58)$$

Um diese mittels Polvorgabe einstellen zu können, muss charakteristische Polynom gebildet werden [15].

$$N(s) = \det(A - LC) \quad (5.59)$$

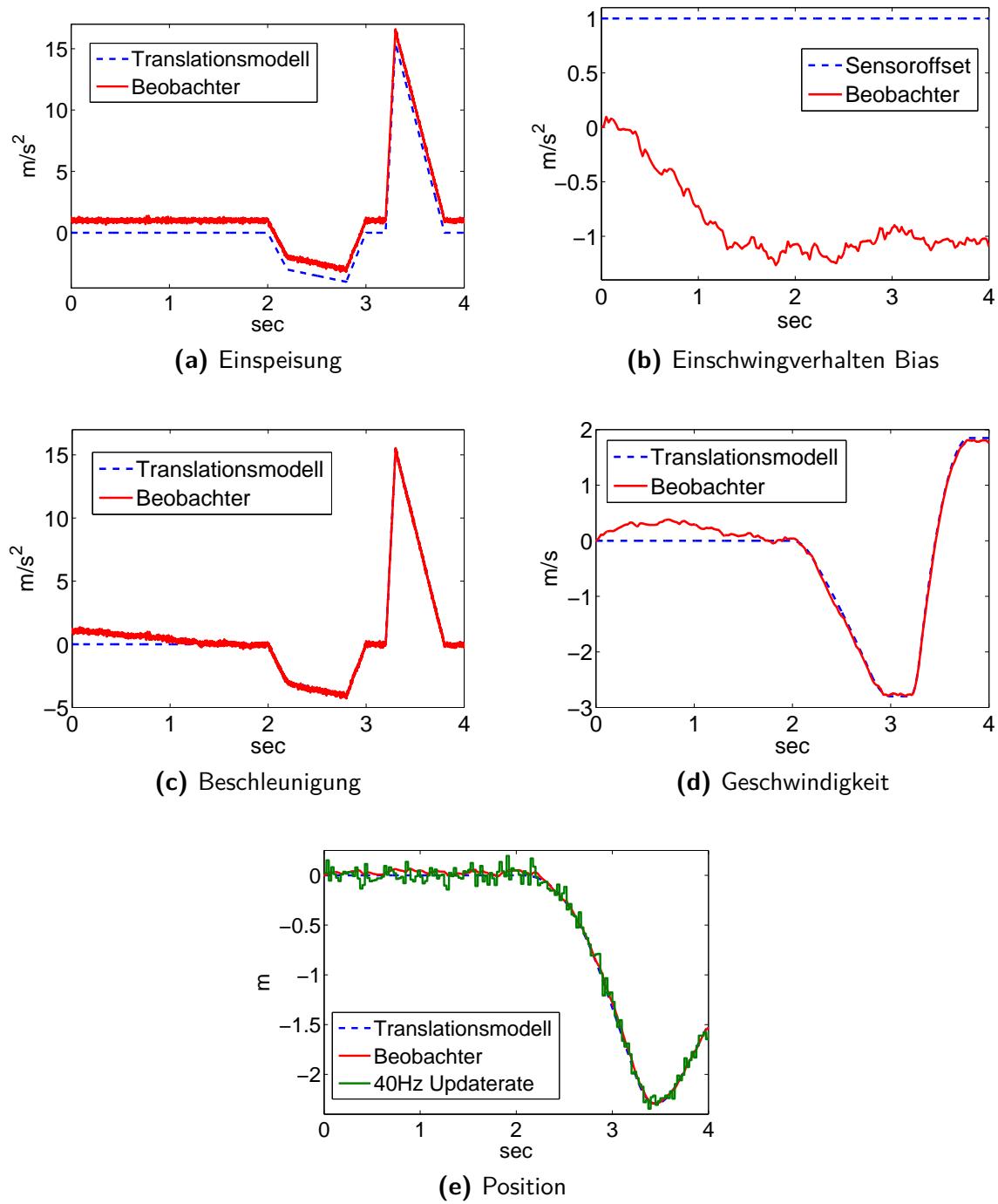
ergibt für das erweiterte Zustandsmodell,

$$N(s) = s^3 + l_1 \cdot s^2 + l_2 \cdot s + l_3 \quad (5.60)$$

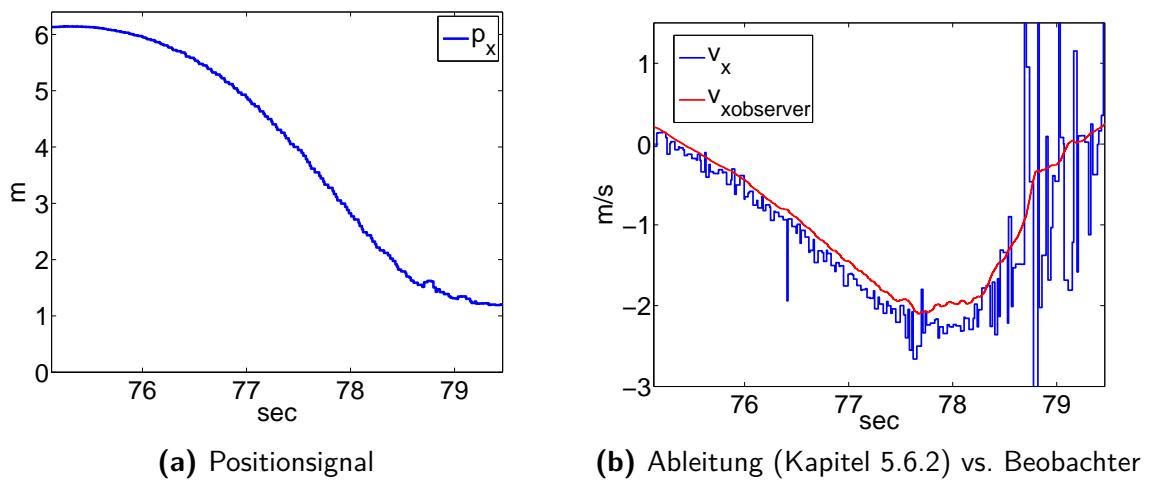
Anhand dieses Polynom sind Koeffizienten der Beobachtermatrix  $L$  nach Kapitel 5.5.1 über Polvorgabe vorgebbar.

# PLATZHALTER GRAFIK

**Abbildung 5.24:** Strukturbild Beobachter



**Abbildung 5.25:** Simulationsergebnisse für das Fusionsfilter



**Abbildung 5.26:** Verbesserung des Fusionsfilter im Vergleich FOAW

---

## Glossar

---

### *Asynchronous Receiver/Transmitter*

Eine gängige serielle Schnittstelle zum Senden und Empfangen von Daten über eine Datenleitung. Bildet den Standard der seriellen Schnittstellen an PCs und Mikrocontrollern.

### *Inter Integrated Circuit*

Von Philips Semiconductor entwickelter serielles Bussystem. Hauptsächlich zur gerätierten Kommunikation zwischen verschiedenen Schaltungsteilen eingesetzt.

### *Internet Protocol*

Ist das am weitverbreitete Netzwerkprotokoll in Computernetzen und stellt die Grundlage des Internets dar

### *Serial Peripheral Interface*

Ein von Motorola entwickelter Standard eines synchronen seriellen Datenbus zur Vernetzung zweier digitaler Schaltungen nach dem Master-Slave-Prinzip

### *Visual Simultaneous Localization and Mapping*

SLAM ist eine Methode der mobilen Robotik zur Schätzung der eigenen Position in einer unbekannten Umgebung. Dafür wird eine Karte des Raumes aus den Messdaten erstellt. Der Zusatz V sagt dabei aus, dass diese Messdaten von einem visuellen Sensor, sprich einer Kamera zur Verfügung gestellt werden. Im deutschen steht SLAM für Simultane Lokalisierung und Kartenerstellung.

---

## Literaturverzeichnis

---

- [1] ACHTELIK, Markus ; ACHTELIK, Michael ; WEISS, Stephan ; SIEGWART, Roland: Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In: *ICRA*, 2011 (Zitiert auf Seiten 30, 38, 42 und 63)
- [2] BUCHHOLZ, J/örg J.: Regelungstechnik und Flugregler / Hochschule Bremen. 2014. – Forschungsbericht (Zitiert auf Seite 16)
- [3] CENSI, Andrea: An ICP variant using a point-to-line metric. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Pasadena, CA, May 2008 (Zitiert auf Seite 27)
- [4] DEUTSCHER, Joachim: *Mehrgrößen Zustandsregelung MZR*. 2013  
(Zitiert auf Seite 63)
- [5] DEUTSCHER, Joachim: *Regelung nichtlinearer Systeme*. 2013  
(Zitiert auf Seiten 37, 38, 42 und 48)
- [6] GARDILL, Markus ; WEIGEL, Robert: *DES Digitale Elktrische Systeme*. 2014  
(Zitiert auf Seite 55)
- [7] HOFFMANN, Frank ; GODDEMEIER, Niklas ; BERTRAM, Torsten: Attitude estimation and control of a quadrocopter. In: *IROS'10*, 2010, S. 1072–1077 (Zitiert auf Seite 31)
- [8] JANABI-SHARIFI, Farrokh ; HAYWARD, Vincent ; J. CHEN, Chung shin: *Discrete-Time Adaptive Windowing for Velocity Estimation*. 2000  
(Zitiert auf Seiten 57, 59 und 60)
- [9] KALLWIES, Jan: *Höhenbestimmung und Höhenregelung bei einem Quadrocopter*, Diplomarbeit, 2013 (Zitiert auf Seite 32)

- [10] LIN, Feng ; BRANDT, Robert ; SAIKALIS, George: Self-Tuning of PID Controllers by Interaction. (2000) (Zitiert auf Seite 52)
- [11] LU, F. ; MILIOS, E.E: Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 1994, S. 935–938 (Zitiert auf Seite 24)
- [12] LUNZE, Jan: *Regelungstechnik 1*. Springer London, Limited, 2008 (Springer-Lehrbuch Bd. 1). – ISBN 9783540689096 (Zitiert auf Seite 37)
- [13] MAY, Stefan: *Skriptum AUT4 Mobile Robotik*. 2013 (Zitiert auf Seiten 24 und 25)
- [14] MORRIS, William ; DRYANOVSKI, Ivan ; XIAO, Jizhong ; MEMBER, Senior: 3D indoor mapping for micro-uavs using hybrid range finders and multi-volume occupancy grids. In: *In RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2010 (Zitiert auf Seiten 20 und 27)
- [15] ROPPENECKER, Günter: *Regelungstechnik B*. 2012 (Zitiert auf Seiten 63 und 64)
- [16] THIELECKE, Jörn: *Eingebettete Navigationssysteme*. 2014 (Zitiert auf Seite 16)
- [17] VOSSIEK, Martin: *Radarssysteme RAS*. 2012 (Zitiert auf Seite 53)

---

## Abbildungsverzeichnis

---

2.1	Hardwareaufbau	3
2.2	Hardwareaufbau	5
2.3	fcuplatine	6
2.4	Kommunikationsstruktur	8
3.1	Topic und Service	10
3.2	Registrierung der Knoten	10
3.3	Kovention	12
3.4	Koordinatensysteme	12
3.5	$z,y',x''$ -Konvention	14
4.1	Verknüpfung der Scantools	19
4.2	Laserprojektion	20
4.3	I-frame	21
4.4	Laserprojektion	23
4.5	I-frame	28
5.1	AscTec_hl_framework in der Kommunikationstruktur	29
5.2	Kaskadenstruktur	30
5.3	Gesamtstruktur Regelung	31
5.4	Gesamtstruktur Regelung	32
5.9	Simulation Inversion	41
5.11	Simulationsergebnisse Referenzmodell	45
5.12	Referenzmodell als Vorsteuerung	46
5.14	Folgeregler	49
5.15	Grafiken Folgeregler	50

5.16 Stabilitätsgebiet . . . . .	50
5.17 Selftuning Folgeregler . . . . .	53
5.18 Auswirkung der Positionsverschiebung auf Varianz der Geschwindigkeit . .	55
5.19 Fensterung . . . . .	57
5.20 End-fit <i>FOAW</i> . . . . .	58
5.21 Flussdiagramm <i>FOAW</i> . . . . .	60
5.22 Messergebnis für <i>FOAW</i> . . . . .	62
5.23 Erweitertes Streckenmodell . . . . .	64
5.24 Beobachter . . . . .	65
5.25 Simulation Beobachter . . . . .	66
5.26 Verbesserung des Fusionsfilter im Vergleich <i>FOAW</i> . . . . .	67
6.1 Hardwareaufbau . . . . .	69
6.2 Hardwareaufbau . . . . .	71
6.3 fcuplatine . . . . .	72
6.4 Kommunikationsstruktur . . . . .	74

---

## Tabellenverzeichnis

---

# **ANHANG A**

---

Datenblätter

---

Date: 2012.11.27

# Scanning Laser Range Finder UTM-30LX/LN Specification

△ × 2	Correction of Repeated Accuracy Representation			3	2012.11.27	Kamon	RS-0155
△ × 1	LED Display in Specificaions added			3	2012.10.23	Kamon	RS-0143
△ × 2	Important Notes on IF is added. <u>External dimension error correction.</u>			3,4	2011.7.6	Kamon	PR-6178
△ × 3	Changes in output signal			3,4,6	2010.7.26	Kamon	PR-5893
△ × 2	Correction on synchronization output			2,4	2009.5.18	Takai	PR-5647
△ × 2	Changes in laser( $\lambda$ :870nm → 905nm)			2,3	2009.4.14	Kamon	PR-5635
△ × 1	Correction			4	2008.8.18	Kamitani	PR-5503
△ × 1	Cautions were added			6	2008.5.1	Kamitani	PR-5466
Symbol	Amendment Details			Amendment	Date	Amended by	Number
Approved by	Checked by	Drawn by	Designed by	Title	<b>UTM-30LX/LN</b> Specification		
MORI	KAMITANI	KAMON	HINO		Drawing No.	<b>C-42-3615</b>	

## 1. Introduction

### 1.1 Operation principles 905nm $\triangle$

UTM-30LX/LN use laser source ( $\lambda = 870\text{nm}$ ) to scan  $270^\circ$  semicircular field (Figure 1). It measures distance to objects in the range and co-ordinates of those point calculated using the step angle. Sensor's measurement data along with the angle are transmitted via communication channel. Laser safety class 1.

Sensor is divided into two types depending upon the type of output.

### 1.2 Type

#### 1.2.1 U TM-30LX

Synchronous output signal is available. The timing chart of this signal is shown in section 6 (Figure 3).  $\triangle$  This synchronous signal can be obtain at each scan. These are mainly intended for robotic applications.

#### 1.2.2 UTM-30LN

It outputs warning signal whenever there is any object in the preset area. These are mainly intended for area protection.

## 2. Structure (Laser range figure)

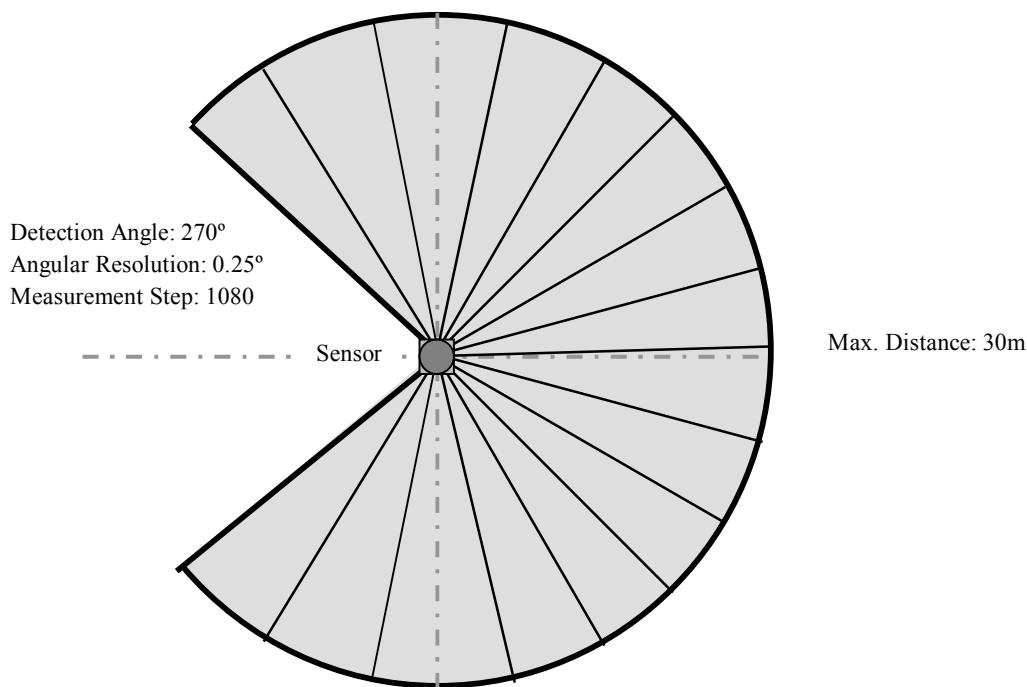


Figure 1

## 3. Important note

- This sensor is not a safety device/tool
- This sensor is not for use in military applications
- Read specifications carefully before use.

Title	UTM-30LX/LN Specification	Drawing No	C-42-3615	2/6
-------	---------------------------	------------	-----------	-----

## 4. Specifications

Product Name	Scanning Laser Range Finder	
Model	<b>UTM-30LX</b>	<b>UTM-30LN</b>
Light Source	Laser Semiconductor $\lambda = 870\text{nm}$ , 905nm	△ Laser Class 1
Supply Voltage	12VDC $\pm 10\%$	
Supply Current	Max: 1A, Normal : 0.7A	
Power Consumption	Less than 8W	
Detection Range and Detection Object	Guaranteed Range: 0.1 ~ 30m (White Kent Sheet) Maximum Range : 0.1 ~ 60m Minimum detectable width at 10m : 130mm (Vary with distance)	
Accuracy	Under 3000lx : White Kent Sheet: $\pm 30\text{mm}^{\ast 1}$ (0.1m to 10m) Under 100000lx : White Kent Sheet: $\pm 50\text{mm}^{\ast 1}$ (0.1m to 10m)	
Measurement Resolution and Repeated Accuracy	1mm 0.1 – 10m : $\sigma < 10\text{mm}$ , 10 – 30m : $\sigma < 30\text{mm}$ (White Kent Sheet) Under 3000lx : $\sigma < 10\text{mm}^{\ast 1}$ (White Kent Sheet up to 10m) ▲ Under 100000lx : $\sigma < 30\text{mm}^{\ast 1}$ (White Kent Sheet up to 10m) ▲	
Scan Angle	270°	
Angular Resolution	0.25° (360°/1440)	
Scan Speed	25ms (Motor speed : 2400rpm)	
Interface	USB Ver2.0 Full Speed (12Mbps)	
Output	Synchronous Output 1- Point	Detection Output 1- Point ▲
LED Display △	Green : Power supply Red : Normal Operation (Continuous), Malfunction (Blink)	Power supply Object detection inside area (Continuous) Malfunction (Blink)
Ambient Condition (Temperature, Humidity)	-10°C ~ +50°C Less than 85%RH (Without Dew, Frost)	
Storage Temperature	-25~75°C	
Environmental Effect	Measured distance will be shorter than the actual distance under rain, snow and direct sunlight* <sup>2</sup> .	
Vibration Resistance	10 ~ 55Hz Double amplitude 1.5mm in each X, Y, Z axis for 2hrs. 55 ~ 200Hz 98m/s <sup>2</sup> sweep of 2min in each X, Y, Z axis for 1hrs.	
Impact Resistance	196m/s <sup>2</sup> In each X, Y, Z axis 10 times.	
Protective Structure	Optics: IP64	
Insulation Resistance	10MΩ DC500V Megger	
Weight	210g (Without cable)	
Case	Polycarbonate	
External Dimension (W×D×H)	60mm×60mm×87mm ▲ MC-40-3127	

\*<sup>1</sup> Under Standard Test Condition (Accuracy can not be guaranteed under direct sunlight.)

\*<sup>2</sup> For sensor functions, please verify the in an indoor environment of 1000 lx or less. In avoiding unnecessary disturbance cause by the raindrops, perform necessary signal processing for LX type and switch OFF the delay function for LN type.

## 5. Quality Reference Value

Vibration resistance during operation	10~150Hz 19.6m/s <sup>2</sup> Sweep of 2min in each X,Y,Z axis for 30min
Impact resistance during operation	49m/s <sup>2</sup> X, Y,Z axis 10 times
Angular Speed	$2\pi/\text{s}$ (1Hz)
Angular Acceleration	$\pi/2\text{rad}/\text{s}^2$
Life-span	5 Years (Varies with operating conditions)
Noise Level	Less than 25dB at 300 mm
Certification	FDA Approval (21 CFR part 1040.10 and 1040.11)

Title	UTM-30LX/LN Specification	Drawing No	<b>C-42-3615</b>	3/6
-------	---------------------------	------------	------------------	-----

## 6. Interface

### 6.1 Robot Cable 4 Pin

Color	Function
Brown	+12 V
Blue	0 V
Green	Synchronous Output/ Detection Output $\triangle$
White	COM Output (0V: Common to Power)

Note: 0 V of the power supply and Output is not internally connected.

Short circuit the 0V (Blue) and COM Output (White) during wiring.  $\triangle$   $\triangle$

### 6.2 USB Connector

TYPE-A

**Note:**

SG for communication and GND are connected internally (Isolated with Input -VIN).

Isolate the device from any connection that generate electric noise.

This sensor is compatible with SCIP2.0 communication protocol standard.

### 6.3 Output circuit diagram

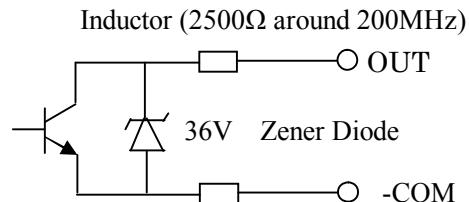


Figure 2

## 7. Control Signal

### 7.1 Synchronous Output (UTM-30LX)

1 pulse is approximately 1 ms. Output signal Synchronization timing chart is shown below. (Figure 3).

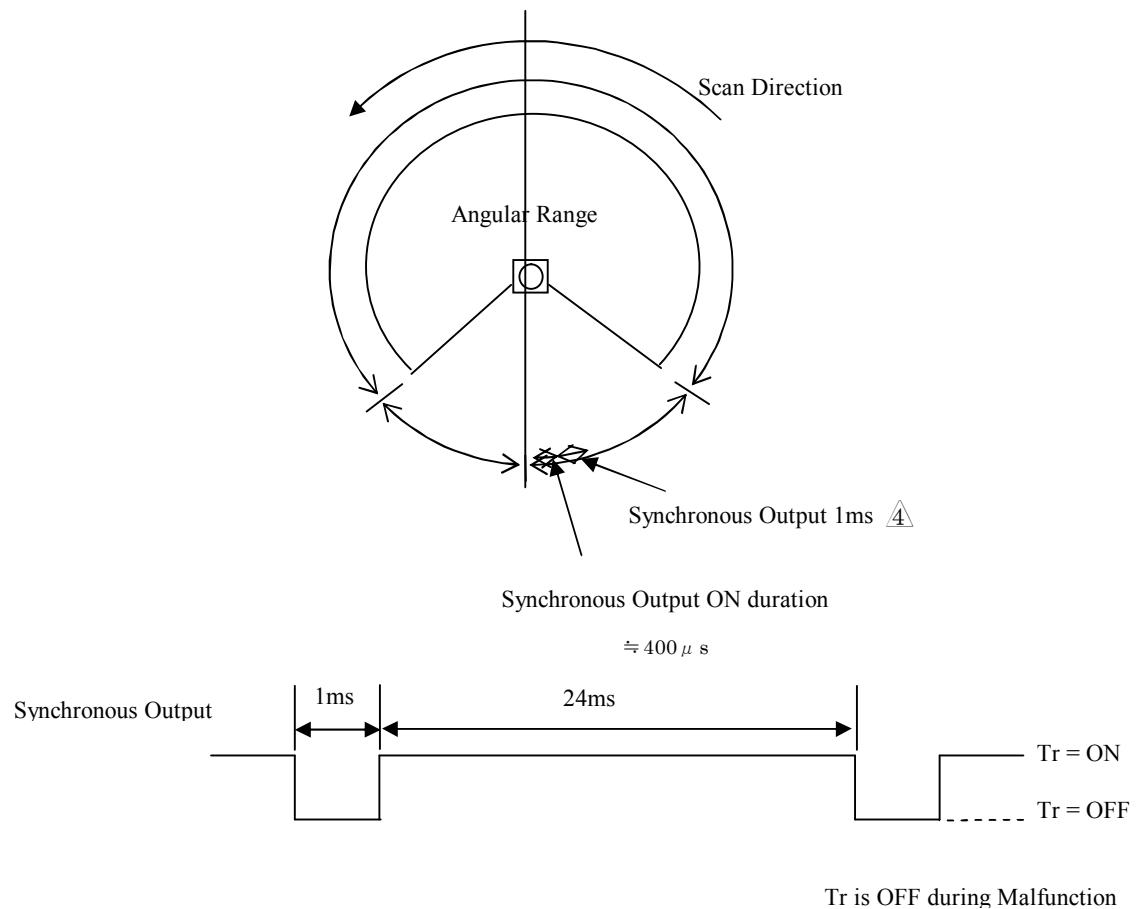


Figure 3

## 7.2 Detection Output (UTM-30LN)

When the signal is set for detection output .The signal switches OFF when obstacle exist inside the area.  
(Output signal is ON when obstacle does not exist.)

Area can be set using 3~7 co-ordinate points.

Maximum of the output delay is 128 times (3.2 sec)

Example

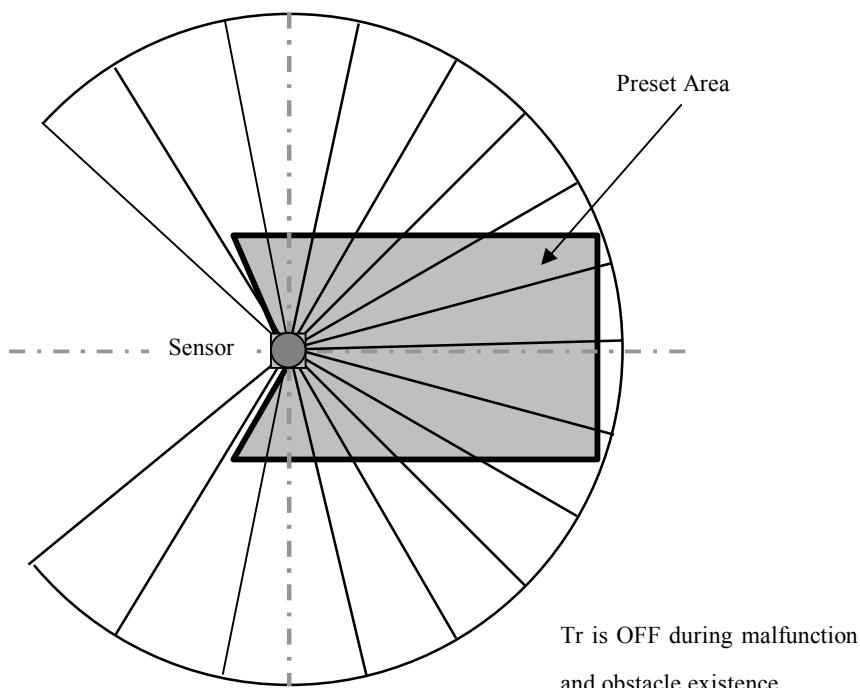


Figure 4

## 8. Malfunction Output:

1. Laser malfunction : When laser does not radiate or exceeds safety class 1.
2. Motor malfunction : When rotation speed is differ from the default value (> 25 m/s).

Synchronous/Warning signal will be turned OFF when these malfunctions are detected. Error details can be obtain via communication.

## 9. Cautions

Heat is generated as the sensor runs at a very high speed. The heat generated is concentrated at the bottom of the sensor. Please mount heatsinks or any appropriate component to release the generated heat. An aluminum plate (200 x 200 x 2) is recommended as the heatsinks.

Error could happen when 2 or more identical sensor is mounted at the same detection plane. This is because the sensor could not identify the origin of the received laser pulses. When this error occur, it will cause 1 -2 step difference, performing data filtering could overcome this problem.

Title	UTM-30LX/LN Specification	Drawing No	C-42-3615	6/6
-------	---------------------------	------------	-----------	-----

## **ANHANG B**

---

Definition flacher Mehrgrößensysteme

---

## Vorlesung: „Regelung nichtlinearer Systeme“

### Flachheit nichtlinearer Mehrgrößensysteme

#### 1. Definition

Ein nichtlineares Mehrgrößensystem  $n$ -ter Ordnung

$$\dot{x} = f(x, u) \quad \text{mit} \quad \text{rang } \frac{\partial f(x, u)}{\partial u} = p, \quad \forall x \in U(x_0), \quad \forall u \in U(u_0) \quad (1)$$

mit  $p$  Eingangsgrößen  $u$  heißt *flach*, wenn es einen fiktiven Ausgang  $y_f = [y_{f1} \dots y_{fp}]^T$  gibt, der die folgenden Bedingungen erfüllt:

1. Die Komponenten  $y_{fi}$ ,  $i = 1, 2, \dots, p$ , des fiktiven Ausgangs lassen sich als Funktion der Zustandsgrößen  $x_\nu$ ,  $\nu = 1, 2, \dots, n$ , und  $u_i$ ,  $i = 1, 2, \dots, p$ , sowie einer endlichen Anzahl von Zeitableitungen  $u_i^{(k_i)}$ ,  $k_i = 1, 2, \dots, \alpha_i$ , ausdrücken, d. h.

$$y_f = \Phi(x, u_1, \dot{u}_1, \dots, u_1^{(\alpha_1)}, \dots, u_p, \dot{u}_p, \dots, u_p^{(\alpha_p)}) = \Phi(x, u, \dot{u}, \dots, u^{(\alpha)}) \quad (2)$$

mit dem Ableitungstupel  $u^{(k)} = (u_1^{(k_1)}, \dots, u_p^{(k_p)})$  und  $k = (k_1, \dots, k_p)$ .

2. Die Zustandsgrößen  $x_\nu$ ,  $\nu = 1, 2, \dots, n$ , und die Eingangsgrößen  $u_i$ ,  $i = 1, 2, \dots, p$ , lassen sich als Funktion der  $y_{fi}$ ,  $i = 1, 2, \dots, p$ , und einer endlichen Anzahl von deren Zeitableitung  $y_{fi}^{(k_i)}$ ,  $k_i = 1, 2, \dots, \kappa_i$ , darstellen, d. h.

$$x = \psi_x(y_{f1}, \dots, y_{f1}^{(\kappa_1-1)}, \dots, y_{fp}, \dots, y_{fp}^{(\kappa_p-1)}) = \psi_x(y_f, \dots, y_f^{(\kappa-1)}) \quad (3)$$

$$u = \psi_u(y_{f1}, \dots, y_{f1}^{(\kappa_1)}, \dots, y_{fp}, \dots, y_{fp}^{(\kappa_p)}) = \psi_u(y_f, \dots, y_f^{(\kappa)}). \quad (4)$$

3. Die Komponenten von  $y_f$  sind *differentiell unabhängig*, d. h. sie erfüllen keine Differentialgleichung der Form

$$\varphi(y_f, \dots, y_f^{(\gamma)}) = 0. \quad (5)$$

Wenn der fiktive Ausgang  $y_f$  die Bedingungen 1–3 erfüllt, so ist er ein *flacher Ausgang* des Systems (1). Ist Bedingung 2 erfüllt, dann ist Bedingung 3 äquivalent zu  $\dim y_f = \dim u = p$ .

## 2. Anmerkungen zur Flachheitsdefinition

- Die Beziehungen (3) und (4) besagen, dass für jede beliebige aber hinreichend oft differenzierbare Trajektorie  $y_{f\star}(t)$  des flachen Ausgangs  $y_f$  die Trajektorien

$$\begin{aligned}x_\star(t) &= \psi_x(y_{f\star}(t), \dots, y_{f\star}^{(\kappa-1)}(t)) \\ u_\star(t) &= \psi_u(y_{f\star}(t), \dots, y_{f\star}^{(\kappa)}(t))\end{aligned}$$

Lösung des Anfangswertproblems

$$\text{DGL: } \dot{x}(t) = f(x(t), u(t)), \quad t > 0 \quad (7)$$

$$\text{AB: } x(0) = x_0 \quad (8)$$

für  $x_0 = \psi_x(y_{f\star}(0), \dots, y_{f\star}^{(\kappa-1)}(0))$  sind. Dabei muss die Systemdifferentialgleichung (7) nicht gelöst werden. Da sich somit die Systemdynamik (d.h. alle Trajektorien des Systems) durch  $y_f$  und endlich viele von dessen Zeitableitungen darstellen lässt, spricht man auch von einer (*endlichen*) *differentiellen Parametrierung* des Systems (7) durch den flachen Ausgang  $y_f$ .

- Die differentielle Unabhängigkeit der Komponenten  $y_{fi}$  des flachen Ausgangs (siehe Bedingung 3 in Abschnitt 1) bedeutet, dass die Trajektorien der Komponenten beliebig und unabhängig voneinander vorgegeben werden dürfen, da sie keine Lösung einer Differentialgleichung sind. Die Trajektorien müssen jedoch hinreichend oft differenzierbar sein, damit die Trajektorien von  $x$  und  $u$  gemäß (3) und (4) existieren.
- Die Bedingung  $\text{rang } \frac{\partial f(x,u)}{\partial u} = p$  an das System (1) stellt sicher, dass die Eingangsgrößen  $u$  des System zumindest lokal voneinander (differentiell) unabhängig sind.
- Der flache Ausgang ist nicht eindeutig, d. h. für ein System kann es mehrere flache Ausgänge geben. Verschiedene flache Ausgänge für dasselbe System können jedoch stets gemäß

$$y_f = \theta(\bar{y}_f, \dot{\bar{y}}_f, \dots, \bar{y}_f^{(s)}) \Leftrightarrow \bar{y}_f = \bar{\theta}(y_f, \dot{y}_f, \dots, y_f^{(s)}) \quad (9)$$

ineinander umgerechnet werden.

- Die Funktion  $\psi_u$  in (4) hängt von den Zeitableitungen des flachen Ausgangs bis zur Ordnung  $\kappa$  ab, da aufgrund von (3) und (1) die Beziehung  $\dot{\psi}_x = f(\psi_x, \psi_u)$  gelten muss.
- Regelgrößen  $y = h(x)$ , die nicht mit dem flachen Ausgang übereinstimmen, lassen sich stets unter Verwendung von (3) gemäß

$$y = h(\psi_x) = \psi_y(y_f, \dots, y_f^{(\beta)}) \quad (10)$$

differentiell parametrieren. Dabei gilt  $\beta = (\beta_1, \beta_2, \dots, \beta_p)$  mit  $\beta_i \leq \kappa_i - 1$ .

- Wenn Bedingung 2 erfüllt ist, dann lässt sich (4) als  $p$  Differentialgleichungen für den flachen Ausgang  $y_f$  in Abhängigkeit von  $u$  interpretieren. Folglich kann der flache Ausgang  $y_f$  nicht die Differentialgleichung (5) erfüllen, da er Lösung der Differentialgleichung (4) ist.

# **ANHANG C**

---

## **Simulinkmodell**

---