

**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!*

Da sich diese Arbeit um ein aktuelles technisches Thema dreht, ist die Verwendung von englischen Begriffen unumgänglich. Es wurde soweit wie möglich versucht, für englische Begriffe eine sinnvolle deutsche Übersetzung zu finden und diese stattdessen zu verwenden. Bei Ausdrücken, bei denen dies nicht möglich war, die aber eine wichtige Bedeutung für diese Arbeit haben, wird mit einer Fußnote eine kurze Erklärung gegeben. Begriffe und Bezeichnungen aus den Standards wurden allgemein nicht übersetzt. Englische Begriffe sind im Text kursiv geschrieben. Wörter, die inzwischen in den alltäglichen Gebrauch der deutschen Sprache eingeflossen sind, wie beispielsweise Computer, Software, Internet etc., werden nicht kursiv geschrieben.

Bei Abkürzungen wird bei der ersten Nennung die volle Bezeichnung ausgeschrieben und die Abkürzung dahinter in Klammern gesetzt. Im Folgenden wird dann nur noch die Abkürzung verwendet.

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

Am Anfang der Arbeit findet sich ein Abkürzungsverzeichnis, in dem alle in dieser Arbeit genannten Abkürzungen und deren ausgeschriebene Formen enthalten sind. Zusätzlich befindet sich im Anschluss an den Ausblick ein Glossar, das die wichtigsten Begriffe nochmals kurz erläutert.



---

## Inhaltsverzeichnis

---

|       |   |    |
|-------|---|----|
| 1     | Einleitung  | 1  |
| 2     | Systemarchitektur des Quadrocopters   | 2  |
| 2.1   | Grundlegende Funktionsweise eines Quadrocopters . . . . .   | 2  |
| 2.2   | Hardwareaufbau . . . . .  | 4  |
| 2.3   | Softwarerestruktur . . . . .  | 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   | Projektion der Laserdaten in das o-frame auf der xy-Ebene des n-frames<br>("laser_ortho_projector") . . . . . | 18 |
| 4.2   | Positonsbestimmung über Scanmatching . . . . .  | 22 |
| 5     | Positionsregelung   | 29 |
| 5.1   | Struktur der Positionsregelung . . . . .  | 29 |
| 5.2   | Modellbildung . . . . .   | 32 |
| 5.3   | Exakte Zustandslinearisierung . . . . .   | 36 |
| 5.4   | Vorsteuerung/Referenzmodell . . . . .   | 39 |
| 5.5   | Folgeregler . . . . .   | 41 |
| 5.5.1 | Einstellung der Dynamik mittels Polvorgabe . . . . .  | 43 |
| 5.5.2 | Automatische Optimierung der Reglerparameter . . . . .  | 46 |

|  |    |
|--|----|
| 5.6 Zustandsschätzung . . . . .  | 46 |
| 5.6.1 Geschwindigkeitbestimmung über die Inertialsenorik . . . . .                         | 47 |
| 5.6.2 Geschwindigkeit als Ableitung der Position . . . . .                                 | 48 |
| 5.6.3 Geschwindigkeitsbestimmung über die Methode First-Order Adaptive Windowing . . . . . | 50 |
| Literaturverzeichnis   | 56 |
| Abbildungsverzeichnis  | 58 |
| Tabellenverzeichnis  | 60 |
| A Anhang   | 61 |

# 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.

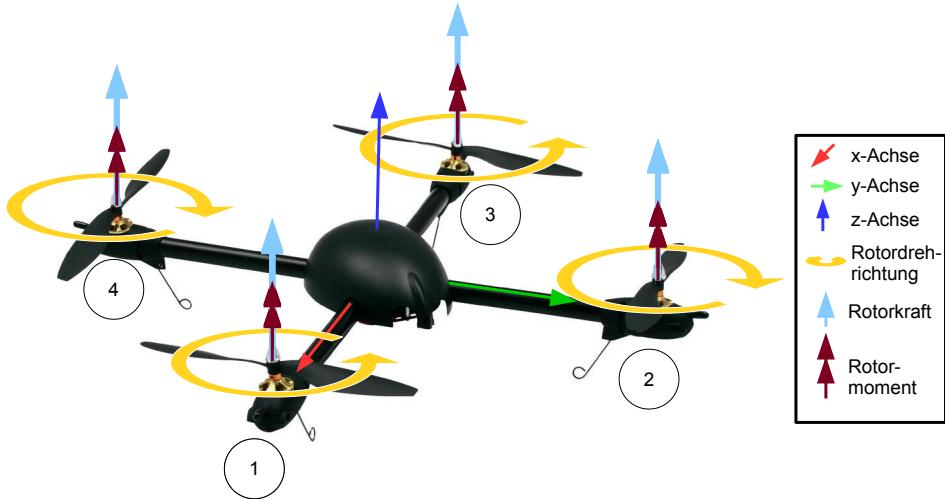
### 2.1 Grundlegende Funktionsweise eines Quadrocopters

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.

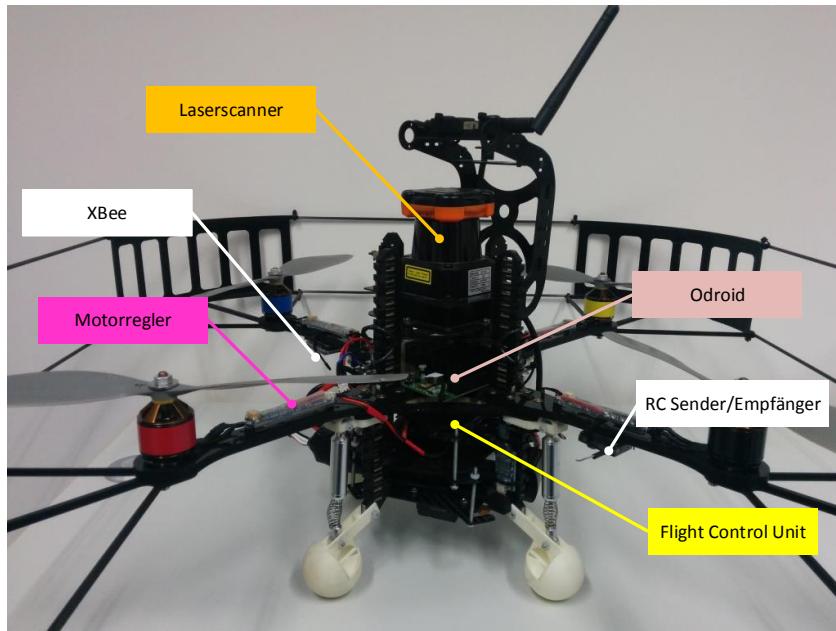
## 2.2 Hardwareaufbau

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 2.2 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 2.4). 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 2.3).

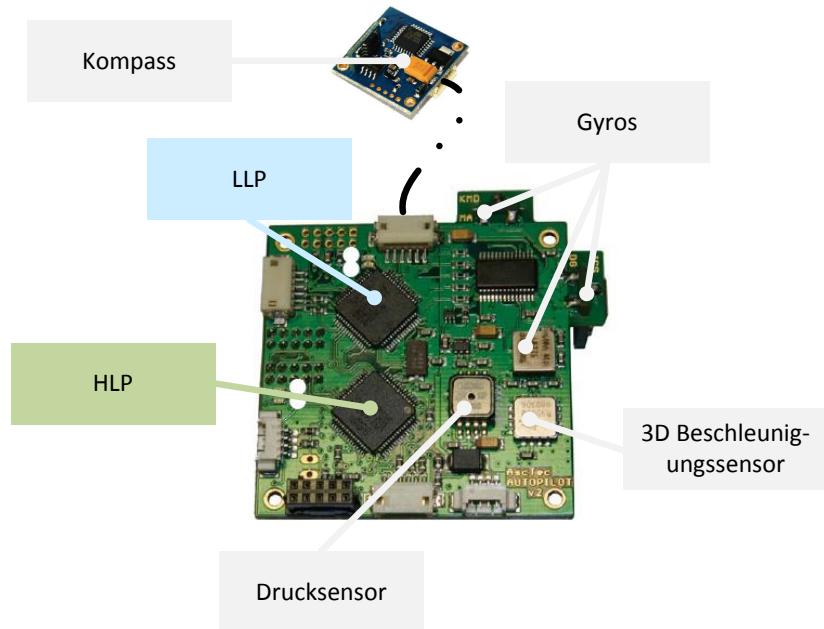
Da der Einsatzbereich im Indoorbereich liegt, ist Drucksensor ist zur Höhenbestimmung in geschlossenen Räumen nicht eignet, da er erst ab einer Höhe von 5m zuverlässige Werte



**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 Entferungen 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 2.3 eingegangen.

## 2.3 Softwarestruktur

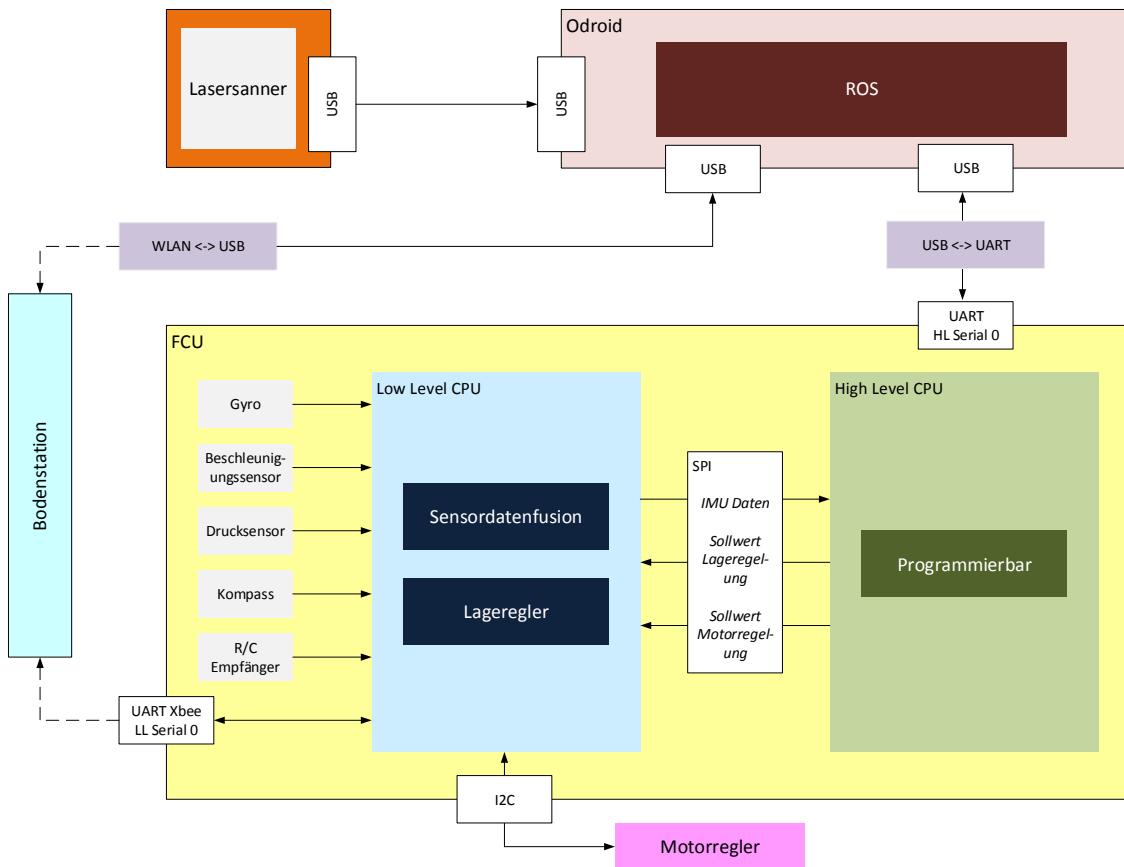
Nachdem im vorhergegangen Kapitel 2.2 die verbaute Hardware vorgestellt wurde, geht es in diesem Abschnitt um die Softwarestruktur(Abbildung 2.4). Es wird aufgezeigt welche Software bereits fest implementiert ist und wo adaptive Applikationen integriert werden können. Des weiteren wird die Kommunikationsstruktur 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 2.3 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 2.2 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

---

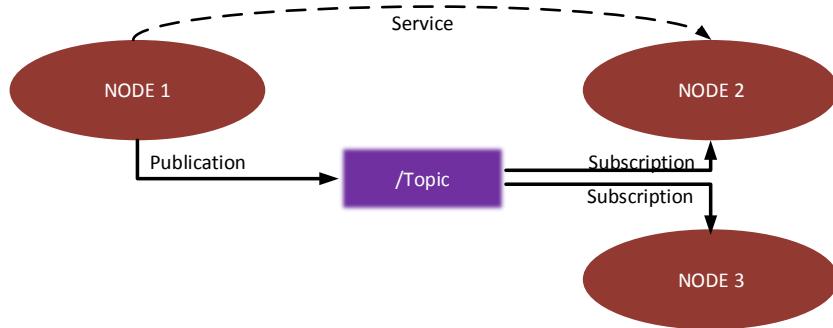
Das Kapitel Grundlagen behandelt die Themen, die in mehreren Abschnitten dieser Arbeit relevant sind. Dabei handelt es sich um das Robot Operation System, die verwendeten Koordinatensysteme und die Transformation zwischen ihnen.

### 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.

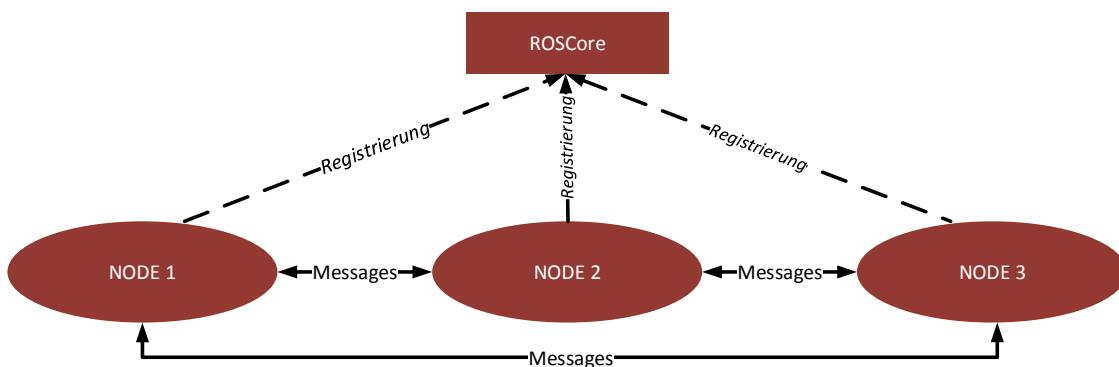
*ROS* stellt dem Softwareentwickler Bibliotheken und Werkzeuge zur Verfügung, die helfen Roboteranwendungen zu erstellen. Das auf einem *Internet Protocol (IP)*-basierende modulare Kommunikationsframework ermöglicht die Verknüpfung von Anwendungssoftware, Sensoren und Aktoren sogar unter mehreren Robotern. 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

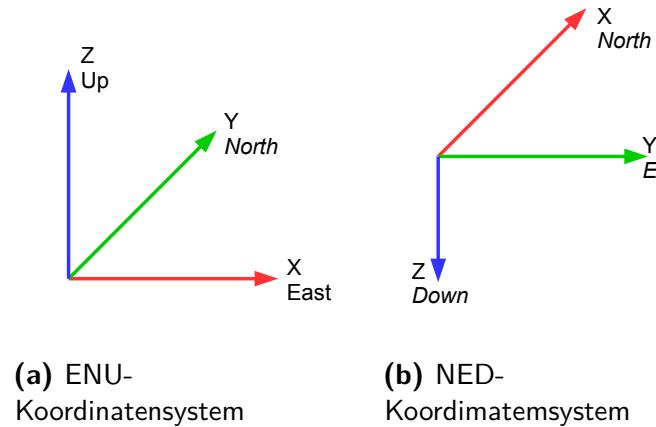


Abbildung 3.3: test

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.

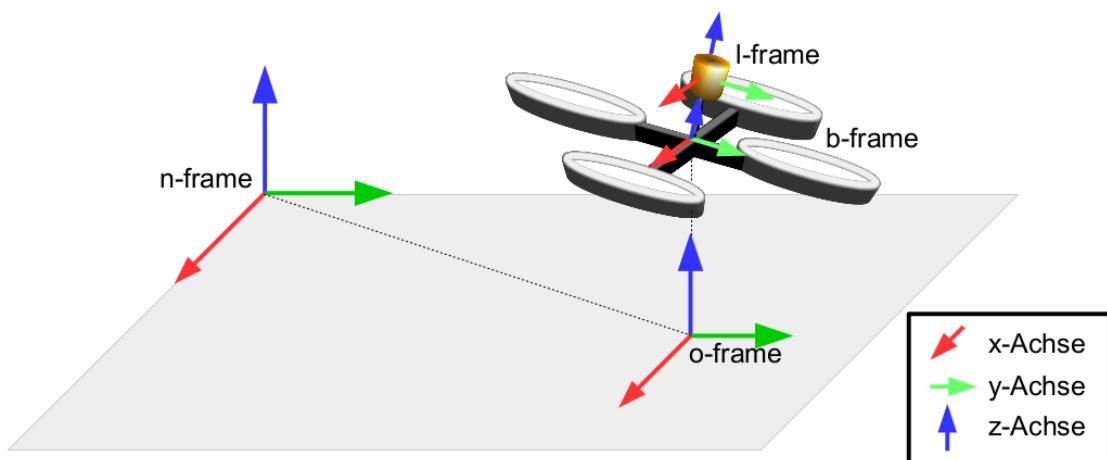


Abbildung 3.4: In der Arbeit angewandte Koordinatensysteme

- **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.

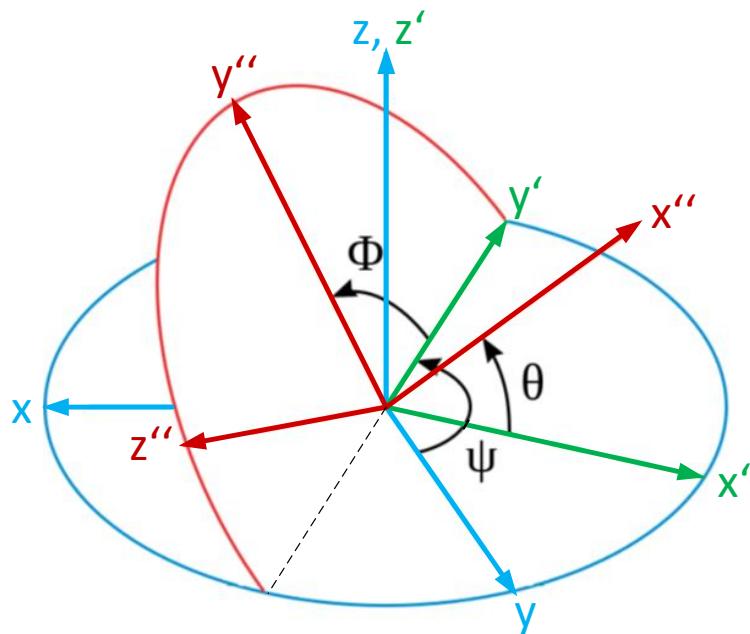
Da sich Messwerte wie zum Beispiel die *IMU*-Daten oder die Laserdaten auf unterschiedlichen Koordinatensysteme beziehen, benötigt man Koordinatentransformation(Kapitel 3.2.2).

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:

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

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

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

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,

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

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 [14] 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

---

Wie schon in der Einleitung (Kapitel 1) sowie der Aufgabenstellung beschrieben, erfolgt die Positionsbestimmung über den auf dem Quadrocopter montierten Lasersanner. Man spricht hierbei von einem Onboard-Lokalisierungssystem. Die aufgenommen Entferungen sind dabei im l-frame definiert. Die Rohdaten enthalten somit keine Information über die Position des Quadrocopters im Navigationskoordinatensystem, sondern eigentlich die Entfernung von umgebenden Objekten, bzw. Wänden. Anhand derer lässt sich jedoch über die Methode des „scanmatching“ die Position in einer zweidimensionalen Ebene bestimmt werden (Kapitel 4.2). Da diese Ebene der xy-Ebene des n-frames entsprechen soll, müssen die Laserdaten zunächst in das o-frame überführt werden (Kapitel 4.1).

Realisiert sind diese Vorgänge in den von ROS zu Verfügung gestellte scan\_tools. Genauer gesagt handelt es sich dabei um dem „laser\_ortho\_projector“- und dem „laser\_scan\_matcher“-Knoten (Abbildung 4.1). Ziel der folgenden Kapitel ist es die Mathematik sowie die Funktionsweise die hinter diesen Algorithmen steht zu erläutern.

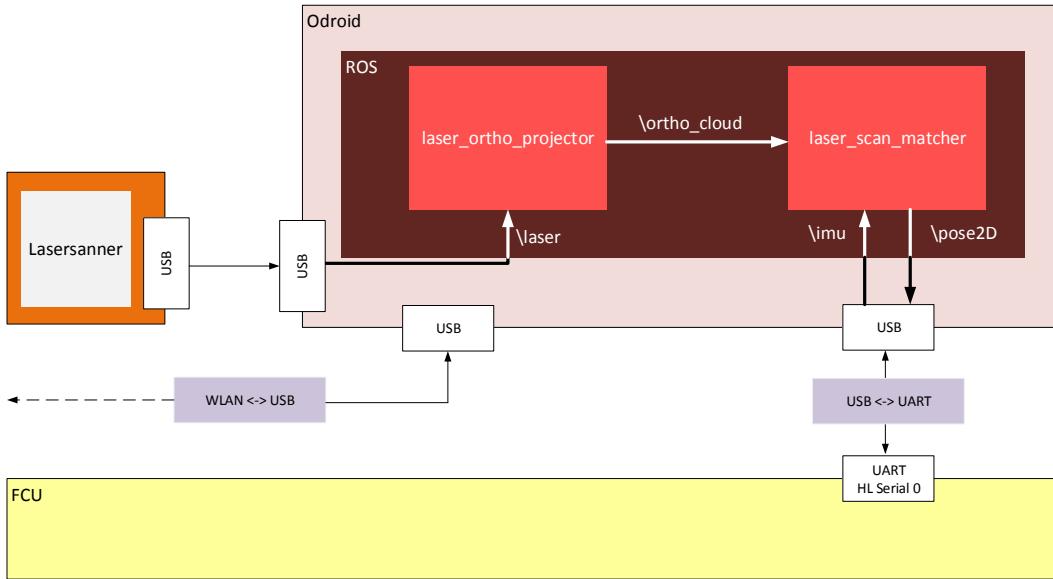
ANMERKUNG ES FEHLT NOCH EIN LITERATURVERZEICHNIS!

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

Bei der Laserprojektion werden die Laserdaten des l-frame orthogonal zur xy-Ebene des n-frames in die des o-frame transformiert. In allgemeiner Form ist dies in Abbildung 4.2a

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

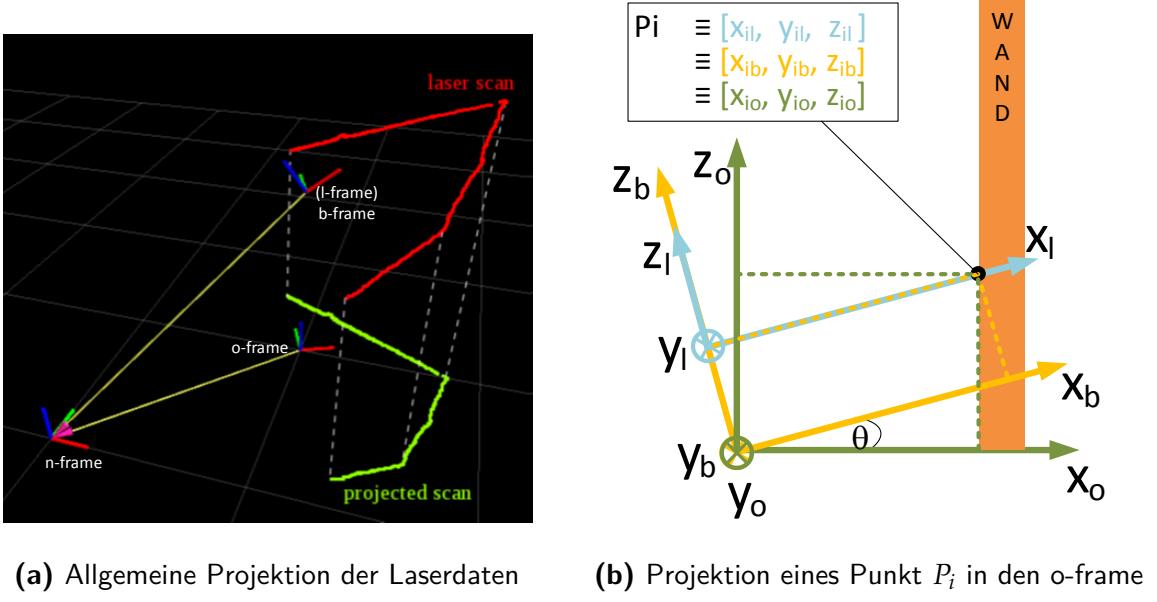
---



**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

dargestellt. Das o-frame definiert später in Kapitel 4.2 die Position und Orientierung des Quadrocopters in der zweidimensionalen Ebene des n-frame. Möglich ist die orthogonalen Projektion nur unter der Annahme, dass es sich bei den erfassten Objekten um Gegenstände mit rechtwinkligen Eigenschaften handelt. Das bedeutet sie weisen unabhängig der Höhe in der sie erfasst werden die gleichen Formen auf. Für geschlossenen 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 eliminiert werden. Die Winkelgrößen liefern die IMU. Der mathematische Ablauf der orthogonalen Transformation wird basierend auf Literatur [13] im Folgenden dargelegt.

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



**Abbildung 4.2:** Projektion der Laserdaten

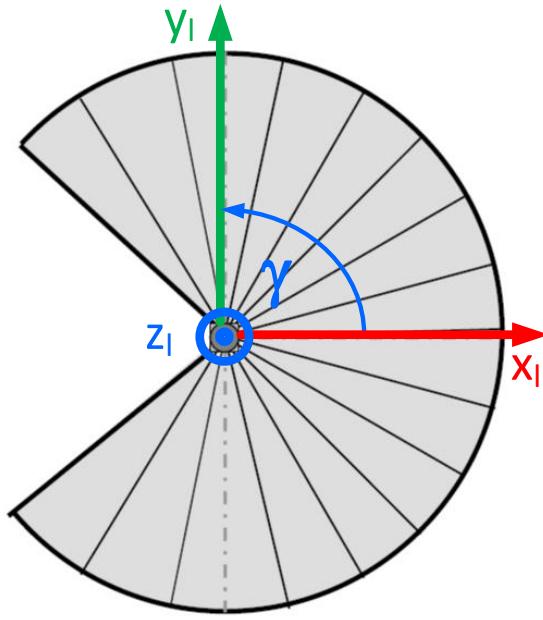
Entfernungsdaten eines Laserumlaufs besteht aus mehreren diskreten Abtastungen (Abbildung 4.3). Übergeben werden sie in Form von Entfernung  $r_i$  in einem Array (Topic \laser Abbildung 4.1). Mittels der Schrittweite von  $0.25^\circ$  lässt sich anhand des Indizes  $i$  für jeden Messpunkt einen Winkel  $\gamma_i$  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 nöten die Laserdaten vom l-frame ins 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  $10cm$  auf der  $z^b$ -Achse, da der Laser oberhalb des Quadrocopterschwerpunktes montiert ist.



**Abbildung 4.3:** Draufsicht I-frame

$$p_i^b = [\cos(\gamma_i) \cdot r_i, \sin(\gamma_i) \cdot r_i, 0.1]^T \quad (4.3)$$

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.

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

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

Dargestellt in Abbildung 4.4a.

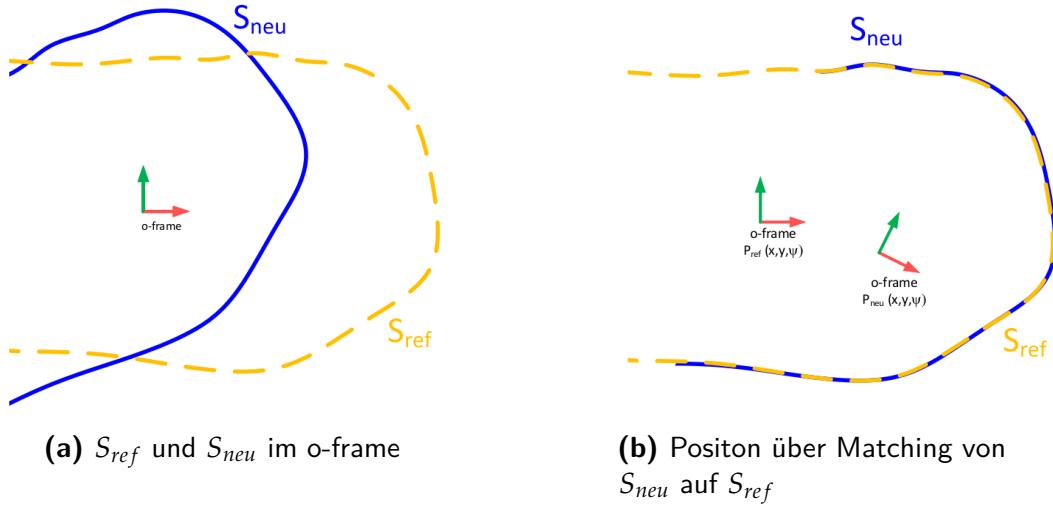


Abbildung 4.4: Prinzip Scanmatching

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 [10] 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 [12] 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 [12] 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.

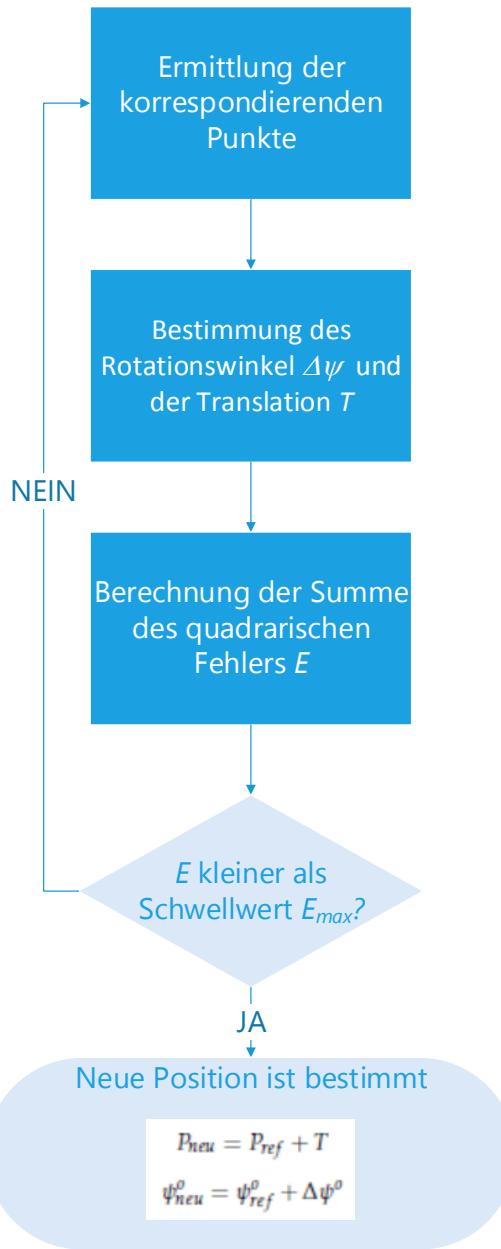
$$P_{neu} = P_{ref} + T \quad (4.27)$$

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

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 [13].

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

---

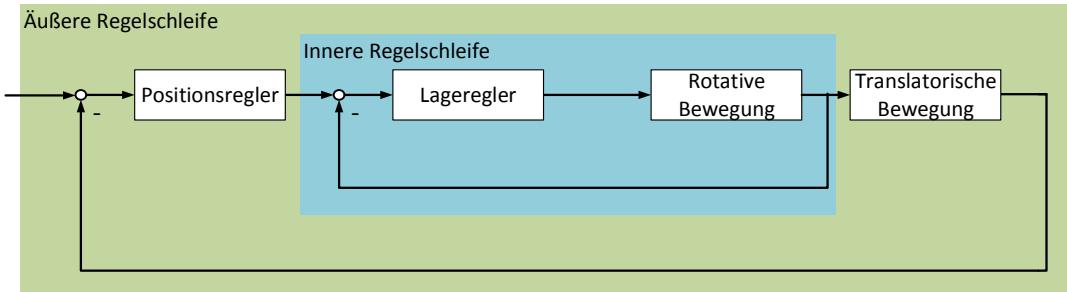
## Verifizierung der Positionsregelung der ETH-Zürich in Verbindung mit einem Laserscanner

---

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 Simultaneous Localization and Mapping (VLSAM)*-Algorithmus und der Fusion der *IMU* die Position des Flugobjekts in der unbekannten Umgebung ermittelt wird. Regelung- und Fusionsalgorihmus sind im Paper [1] beschrieben. Aufgabe dieses Kapitel ist es die dort veröffentlichten Annahmen und Formel zu verifizieren. Dies erfolgt über Literaturrecherchen und Herleitung der publizierten Gleichungen. Unter Berücksichtigung, das die Position nun mehr vom Laser über die Kapitel 4 vorgestellten Algorihmen 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

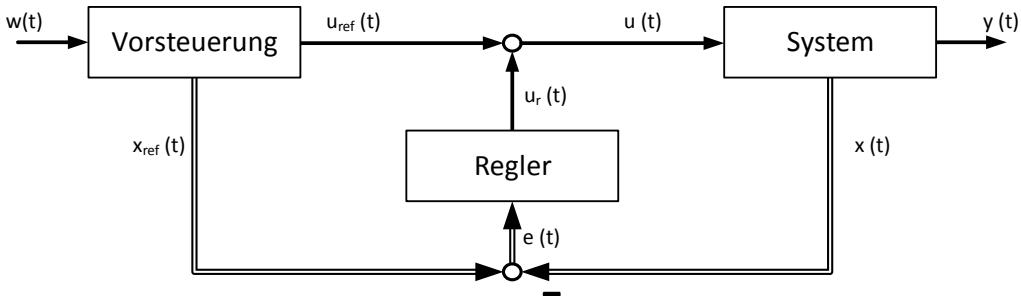
Das Hauptaugenmerk diese Unterkapitel liegt darauf, wie sich die Positionsregelung in die in Kapitel 2.3 vorgestellte Systemstruktur einfügt. Welche Softwarekomponenten dafür auf dem *HLP* integriert wurden. Wie die Kommunikation zwischen ihnen und der Umgebung aussieht. Mit dem Ziel ein grundlegendes Verständnis für die Funktionsweise der Regelung zu generieren.



**Abbildung 5.1:** Kaskadenstruktur der vereinfachten Positionsregelung

Die Positionsregelung wird auf die Lageregelung (engl. Attitudecontrol) aufgesetzt. Daraus resultiert eine Kaskadenstruktur (Abbildung 5.1). 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 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 [6] beschrieben. Die Ungewissheit über den Regleraufbau der Lageregelung stellt für den Entwurf der überlagerten Positionsregelung kein Problem dar. Von Interesse ist lediglich, dass die Annahme einer sehr hohen Dynamik dieses Reglers zur Vereinfachung des für die Positionsregelung benötigten Modells (Kapitel 5.2) führt. Hohe Dynamik bedeutet, dass der Sollwinkel in einer sehr kurzen Zeit  $t \rightarrow 0$  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 lineare zwei Freiheitsgrade Regelung angewandt werden. Diese besteht aus einer Vorsteuerung und einem Folgeregler. Die Vorsteuerung auf dem *HLP* ist in Form eines Referenzmodells (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

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



**Abbildung 5.2:** Struktur zwei Freiheitsgraderegelung

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 gewährleiste. Deshalb ist zusätzlich der Folgeregel (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 5.3 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[8] 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.

---

<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.

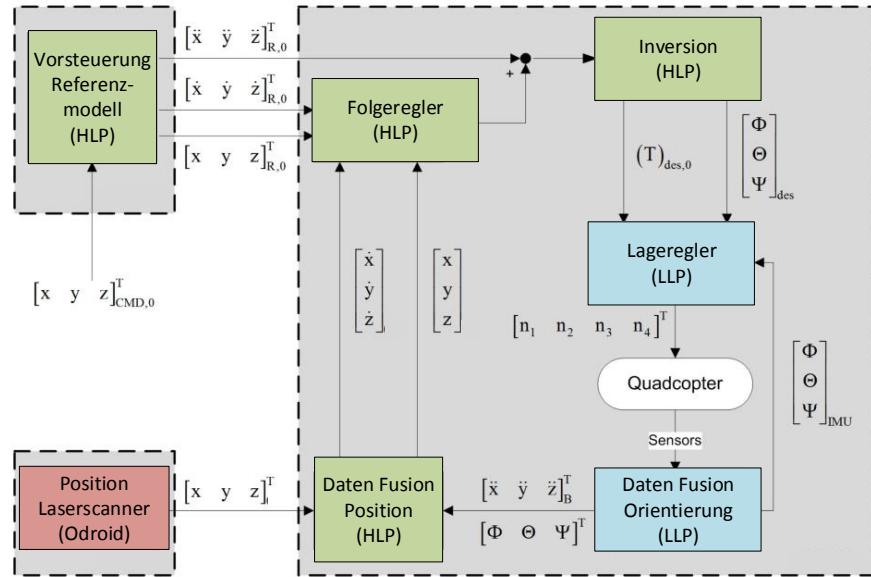


Abbildung 5.3: Struktur der Regelung

## 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.

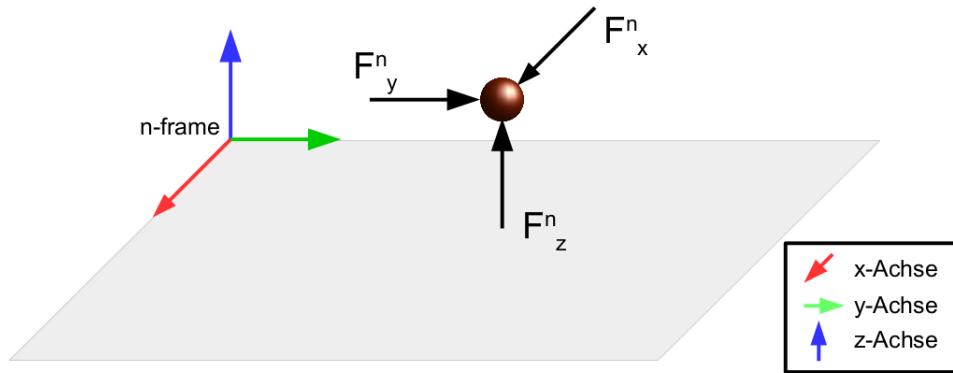
Bevor jedoch die Bewegungsgleichungen des Quadrocopter aufgestellt werden können, ist es notwendig die Freiheitsgrade des dynamischen Systems gegenüber des Bezugssystems zu bestimmen. Da zwischen dem n-frame(Bezugssystem) und dem b-frame(Quadrocopter) keine mechanische Verbindung oder konstante Koordinatentransformation 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

einzu kalkulieren 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 regelnde Verhalten so einfach wie möglich, aber so detailliert wie nötig darstellen.“ Betrachtet nun die Struktur des implementierten Flugregelung (Abbildung 5.3) so lässt sich das Modell aus Sicht der Positionsregelung stark reduzieren.

Grund hierfür ist die in Bild 5.1 dargestellte Kaskadenstruktur der Regelung. In Verbindung mit Abbildung 5.3 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  $T \approx 0.1 \text{ s}$ . Mit der Gewissheit, das 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 2.1) der Rotoren. Letzt genannte Kraft muss zur Anwendung der Newtonischen Gesetze in Kraftkomponenten des n-frames zerlegt werden (Abbildung 5.4). Da die Dynamik der Lageregelung



**Abbildung 5.4:** Auf den Quadrocopter wirkende Kraft. Definiert im n-frame

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,

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

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

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

Für die konstante Masse  $m (= 1.863 \text{ kg})$  des Quadrocopters, 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.

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

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.

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

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

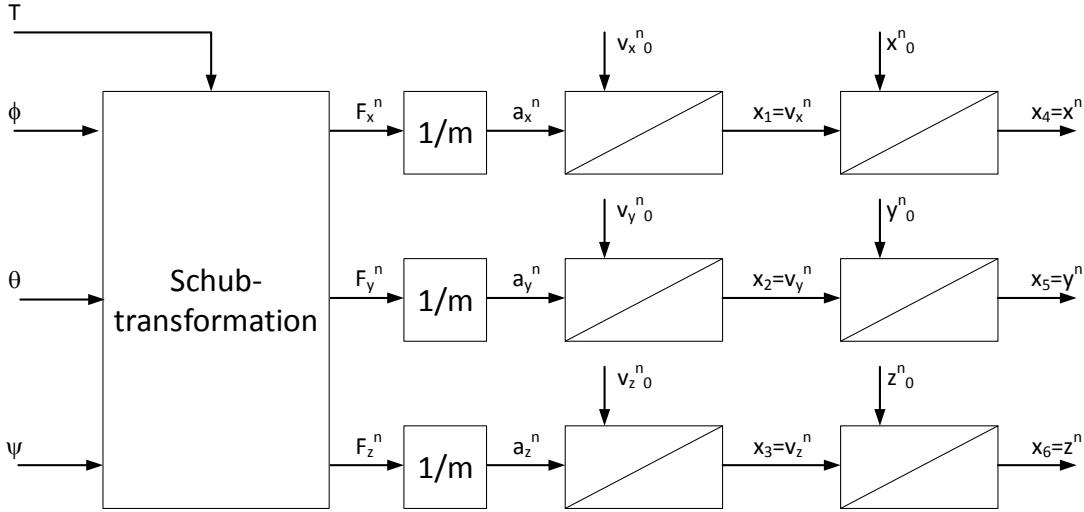


Abbildung 5.5: Modellstruktur des Quadrocopters

### 5.3 Exakte Zustandslinearisierung

Betrachtet man die Modellstruktur für die translatorische Bewegung (Abbildung 5.5) so kann man erkennen, dass für fiktive Eingänge  $u_f = [a_x^n \ a_y^n \ a_z^n]^T$  das Modell aus drei unabhängigen entkoppelten Integriererketten 2. Ordnung bestehen würde. Die per Definition der Linearität [11] ein lineares Zustandsverhalten aufweisen. Gesucht ist also ein Stellgesetz, das für die fiktiven Eingangen  $u_f$  die Stellgrößen der realen Eingänge  $u = [\phi \ \theta \ \psi \ T]$  generiert. Eine sogenannte Inversion.

Damit aus dem nichtlinearen Modell Kapitel 5.2 ein Exakt zustandslinearisierendes Stellgesetz (Inversion) generiert werden kann, muss es sich bei dem Ausgang  $y = [x \ y \ z]$  um einen sogenannten flachen Ausgang  $y_f = [y_{f1} \ y_{f2} \ y_{f3}]$  handeln. Per Definition (siehe ANHANG DEFINITION FLACHHEIT NICHTLINEARER MEHRGRÖ?ENSYSTEME) kann es sich bei dem gewünschten flachen Ausgang  $y_f = [x \ y \ z]$  nur um einen solchen Handeln wenn die Anzahl der flachen Ausgänge  $p$  derer der realen Eingänge  $u$  entspricht. Dies ist nicht der Fall. Damit dennoch eine Inversion durchgeführt werden kann, ist es möglich ein flacher Ausgang mit der selben Anzahl an Ausgängen wie Eingängen zu suchen. Dieser würde nicht mehr mit der Position  $P^n$  übereinstimmen. Deshalb wurde in Paper [1] ein anderer Weg eingeschlagen. Um die Position als flachen Ausgang zu erhalten, wurde

ein Koordinatensystem eingefügt welches im Kapitel 3.2.1 als o-frame beschrieben ist. So können die Beschleunigungsvorgaben für den gewünschten fiktiven Eingang  $u_f$  über eine einfache Transformation (Gleichung 3.1) um den Winkel  $\psi$  ins o-frame übertragen werden. Durch diesen Trick reduziert sich die Anzahl der Eingänge der für Inversion um einen. Somit entspricht die Anzahl der verbliebenen Eingängen  $\tilde{u} = [\phi \ \theta \ T]$  denen des gewünschten flachen Ausgangs. Damit ist die Flachheit des Ausgangs  $y_f$  nicht erwiesen. Es fehlt per Definition (vgl. ANHANG) noch der Beweis, dass sich alle Zustände  $x$  und die reduzierten Eingänge  $\tilde{u}$  durch die flachen Ausgänge und deren Ableitungen darstellen lassen. Für die Zustände ist dies ohne großen Rechenaufwand möglich.

$$\begin{aligned} x_4 &= x = y_f 1 \\ x_5 &= y = y_f 2 \\ x_6 &= z = y_f 3 \\ x_1 &= \dot{x} = \dot{y}_f 1 \\ x_2 &= \dot{y} = \dot{y}_f 2 \\ x_3 &= \dot{z} = \dot{y}_f 3 \end{aligned} \tag{5.7}$$

Für die Eingangsgrößen  $\tilde{u}$  müssen die Zustandsgleichungen für  $\dot{x}_1, \dot{x}_2$  und  $\dot{x}_3$  nach  $\phi, \theta$  und  $T$  aufgelöst werden. Dabei entspricht  $\dot{x}_1 = a_x = \ddot{y}_f 1 = u_{f1}$ ,  $\dot{x}_1 = a_y = \ddot{y}_f 2 = u_{f2}$  sowie  $\dot{x}_1 = a_z = \ddot{y}_f 3 = u_{f3}$ . Damit kann der Schub  $T$  als Betrag der wirkenden, bzw. geforderten Beschleunigungen multipliziert mit der Masse dargestellt werden.

$$\begin{aligned} T &= m \cdot \sqrt{a_x^2 + a_y^2 + (a_z + g)^2} \\ &= m \cdot \sqrt{\ddot{y}_{f1}^2 + \ddot{y}_{f2}^2 + (\ddot{y}_{f3} + g)^2} \end{aligned} \tag{5.8}$$

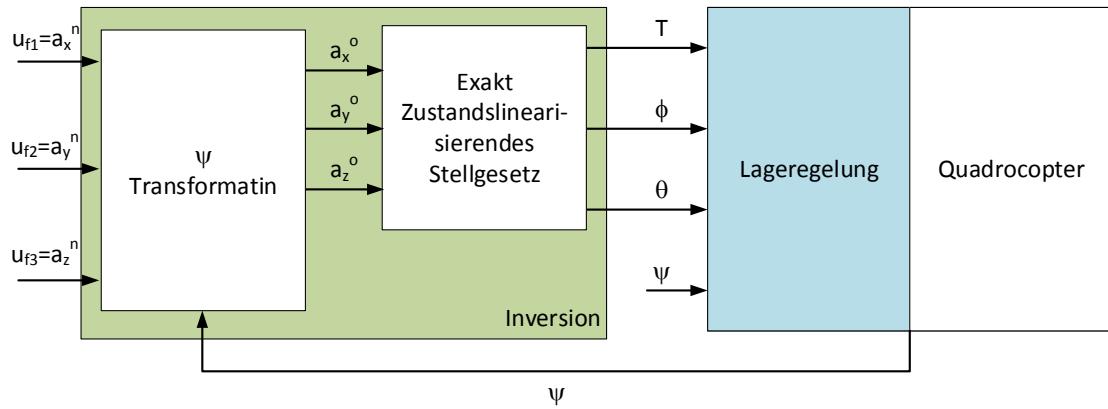
Nun da  $T$  bestimmt ist, kann unter Beachtung das  $\psi = 0$  durch die vorgelagerte Transformation  $\dot{x}_2$  aus Gleichung 5.6 nach  $\phi$  auflösen. Damit ergibt sich,

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

θ lässt sich aufgrund der trigonometrischen Beziehung  $\tan\alpha = \frac{\sin\alpha}{\cos\alpha}$  aus den Zustandsgleichungen für  $x_1$  und  $x_3$  berechnen.

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

Die Stellgesetze der Inversion sind damit Gleichung (5.8), (5.9) und (5.10) mathematisch beschrieben. In Verbindung mit der Koordinatentransformation von  $\psi$  ergibt sich die in Abbildung 5.6 dargestellte Gesamtinversion. Aussicht der noch zu implementieren Positionsregelung besteht das zu regelnde System nun aus drei entkoppelten Integrierketten (Abbildung 5.7). Die nun ein lineares Ein-/Ausgangsverhalten aufweisen, welches jedoch instabil ist. Deshalb muss der Folgeregelung (Kapitel 5.5) eine stabile E/A-Dynamik vorgegeben werden. Dafür zuständig ist die im folgenden Kapitel 5.4 beschriebene Vorsteuerung.



**Abbildung 5.6:** Gesamtinversion

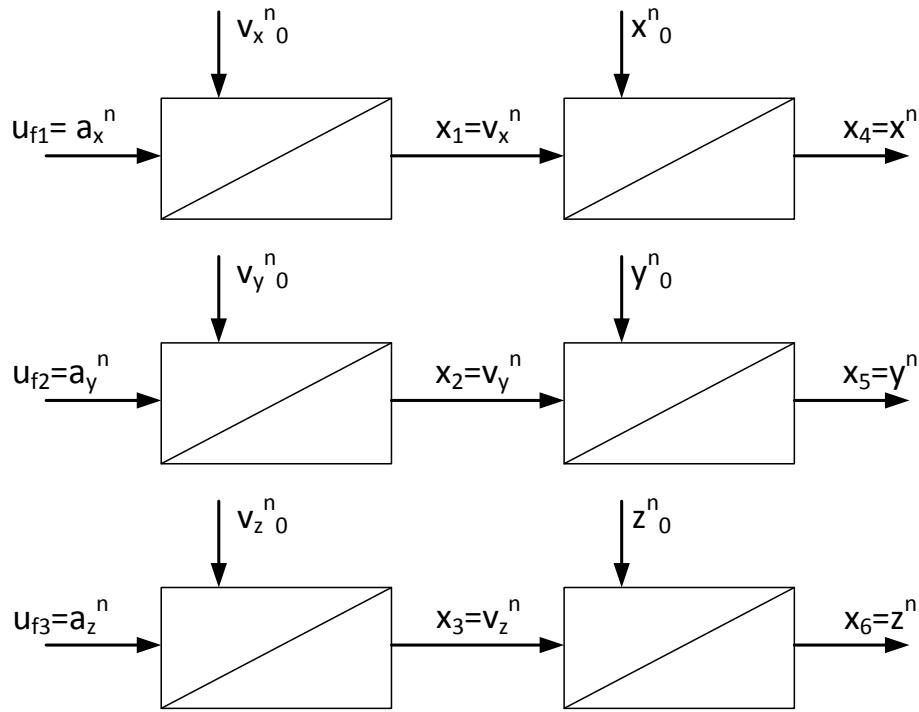


Abbildung 5.7: Modell aus Sicht der Positionsregelung dank Inversion

## 5.4 Vorsteuerung/Referenzmodell

Zur Vorgabe des gewünschten E/A-Verhalten benötigt man als Vorsteuerung ein Referenzmodell. Diesem werden die anzufliegenden Positionen  $P_{cmd}^n$  übergeben. Aufgabe der Vorsteuerung ist es daraus eine Trajektorie zu generieren. Diese beschreibt den Pfad über den der Quadrocopter in diese Koordinate überführt wird. Da die neuen Eingangsgrößen  $u_f$  des zustandslinearisierten Modells Beschleunigungsvorgaben sind, muss der Verlauf der Positionsvorgabe der Vorsteuerung  $P_{ref}^n$  einmal stetig differenzierbar sein, sodass die zweite Ableitung existiert. Damit dies gewährleistet kann nach [4] die E/A Dynamik für dieses System über folgende Formel vorgegeben werden.

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

Stellt man hierfür 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 dem einem Übertragungsglied eines 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 nun (5.13) mit (5.12) gleich kann man die Koeffizienten  $c_0$  und  $c_1$  abhängig der Eigenfrequenz  $\omega_0$  und der Dämpfung  $D$  bestimmen.

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

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

Somit lässt ist der Verlauf der Trajektorie für  $u_{ref} = \ddot{P}_{ref}^n$  (Gleichung 5.11) folgendermaßen festgelegt.

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

Vorteil dieser Darstellung ist, das die Dynamik der Vorsteuerung über die in den Grundlagen der Regelungstechnik [11] vermittelten Entwurfskriterien für PT2-Glieder bestimmen lässt. So ist die Schwingungsfähigkeit über die Dämpfung  $D$  beeinflussbar ( $D < 1$ , keine Schwingung). Mit  $\omega_0$  kann schließlich eingestellt werden wie schnell die Zielkoordinaten erreicht werden sollen.  $\omega$  ist da bei durch die maximal zulässige Stellgröße beschränkt.

Das Strukturbild des resultierten Referenzmodells ist in Abbildung 5.8 visualisiert. Es ist anzumerken, das aufgrund der entkoppelten Integriererketten eine separate Dynamikvorgabe für jeden Zweig des zustandslinearisierten Modells möglich ist. Weiterhin besteht die Möglich die Vorsteuerung so auszubauen, das Geschwindigkeiten vorgegeben werden können mit den er sich aus Sicht des Quadrocoptes im Raum bewegen soll. Dabei erfolgt die Vorgabe der Soll-Geschwindigkeiten im o-frame. Diese wird Integriert und die anschließend ins n-frame transformiert.

$$P_{cmd}^n = \int (M_z^T \cdot v^o) dt + P_{cmd0}^n \quad (5.17)$$

# PLATZHALTER GRAFIK

**Abbildung 5.8:** Sturkturbild Vorsteuerung

Mit wird dem sich stetig Verändernde  $P_{cmd}^n$  wird das Referenzmodell (5.16) gespeist. Somit lassen sich zum Beispiel über die Fernbedienungen Geschwindigkeiten vorgeben, was die Handhabbarkeit des Quadrocopters deutlich vereinfacht.

Die Vorsteuerung ist somit für alle Anwendungszwecke entworfen. Simuliert man das Modell (Abbildung 5.6) inklusive des Stellgesetzes der Vorsteuerung Gleichung (5.16) für konsistente Anfangsbedingungen  $P_{ref0}^n = P_0^n$  so ergibt sich die gewünschte Flugkurve. In der Realität ist die Konsistenz der Anfangsbedingungen jedoch nicht immer gegeben. Außerdem können äußere Krafteinwirkungen wie zum Beispiel Winde wirken, die den Quadrocopter von der Trajektorie auslenken. Um diesen Effekten entgegen zuwirken benötigt man nun einen Folgeregler. Dessen Aufgabe besteht darin äußere Einflüsse zu eliminieren.

## 5.5 Folgeregler

Inkonsistente Anfangsbedingungen, externe Störungen sowie Modellunsicherheiten bewirken einen Ausgangsfolgefehler  $e = P_{ref}^n - P^n$ . Dieser ergibt sich durch die Fortpflanzung des für das reale System nicht ausreichenden Stellsignals  $\ddot{e} = u_{f,ref} - u_f$ . Das bedeutet, die Fehlerdynamik kann ebenfalls über eine zweifache Integratorkette modelliert werden. Zweifache Integratorkette ergo instabiles System. Damit der Quadrocopter auf die Soll-Trajektorie konvergiert, muss die Fehlerdynamik stabilisiert werden. Dafür ist es von nötig, die Dynamik der Folgefehler vorgeben zu können. Zu diesem Zweck werden die Zustandsgrößen des

Ausgangsfolgefehlermodells zurückgeführt (Abbildung 5.9). Daraus ergibt sich folgende Differentialgleichung.

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

mit

$$\begin{aligned} e &= P_{ref}^n - P^n \\ \dot{e} &= \dot{P}_{ref}^n - \dot{P}^n \\ \ddot{e} &= u_{fref} - u_f \end{aligned} \quad (5.19)$$

ergibt sich folgendes Stellgesetz für die Eingangsgröße  $u_f$  des zustandslinearsiererenten Systems das den Folgefehler ausregelt.

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

Die Annäherungsverhalten ist Abhängig von den Koeffizienten  $\tilde{c}_0$  und  $\tilde{c}_1$ . Anhand von 5.21 lassen sich für diese mittels der Polvorgabe die Dynamik vorgeben.

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

# PLATZHALTER GRAFIK

**Abbildung 5.9:** Ausgangsfolgefehlermodell mit Zustandsrückführung

Mit einer konstanten Kraft wirken diese auf das Flugsystem. Beachtet man die diese Kraft hervorgerufenen Beschleunigung  $a_{st}$  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 die Erweiterung des Folgeregeler mit einer Integrierenden Komponente [4].

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

Der I-Anteil liefert jetzt einen Signalanteil zur Kompensation der Störung im stationären Zustand. Den Beweis dafür erhält man in dem man die Integro-Differentialgleichung (5.23) nach der Zeit ableitet ergibt sich Folgenden 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 eingeschwungenem 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 5.10).

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

Wie auch schon zuvor kann die Dynamik Polvorgabe festgelegt werden. Für den Folgeregler mit I-Anteil ist hierzu die Differentialgleichung (5.24) zu verwenden.

### 5.5.1 Einstellung der Dynamik mittels Polvorgabe

Anhand der Lage der Polstellen einer Übertragungsfunktion, können Rückschlüsse über das Einschwingverhalten und die Stabilität getroffen werden. Ziel einer Zustandsregelung, wie

# PLATZHALTER GRAFIK

**Abbildung 5.10:** Folgeregler

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 so ist das System asymptotisch stabil GLEICHUNG ansonsten instabil(Abbildung 5.11).

# PLATZHALTER GRAFIK

**Abbildung 5.11:** 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 Reellenachse befinden desto schneller ist das System.
- Bei mehreren Polstellen wird das Verhalten hauptsächlich über die Pole mit dem größten Realteil bestimmt. Man nennt sie deshalb auch dominante Pole.

Betrachtet man eine Übertragungsfunktion im Bildbereich.

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

Entsprechen die Polstellen  $\lambda_i$  vom System, denn Nullstellen  $s_i$  des Nennerpolynom.

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

Das dynamische Verhalten kann jetzt anhand der Pollage analysiert werden. Im Umkehrschluss können für frei wählbare Koeffizienten des Nennerpolynoms die Dynamik über Vorgabe von Soll-Polestellen  $\lambda_{si}$  erfolgen. Möglich ist dies mit Hilfe einen Koeffizientenvergleichs.

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

Somit ist es möglich mittels (5.21) oder (5.24) das Einschwingverhalten der Fehlerdifferenzialgleichung vorzugeben. Wie man jedoch die Polstellen optimal bestimmt, dafür gibt es keine generelle Vorgehensweise. In der Regel werden sie empirisch über Simulationen festgelegt und zum Schluss am realen Modell getestet bzw. gegebenenfalls optimiert. Dies geschieht meist händisch. Allerdings gibt es hier Algorithmen die ausgehend von einem Startwert, die Parameter automatisch an das System anpassen.

### 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. Diesen Vorgang bezeichnet man als selftuning.

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

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

Das bedeutet um ihn zu optimieren, können Tuningalgorithmen angewandt werden die für einen PID-Regler entwickelt worden sind. So wurde der unter [9] vorgestellte Algorithmus der die Reglerparameter über eine adaptive Interaktion tunt, in ROS integriert. Die Interaktion besteht dabei aus drei Differentialgleichungen.

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

Dabei entspricht  $\gamma$  dem Anpassungskoeffizient. In [9] ist  $\gamma = 10$  empfohlen. Die Struktur des sich daraus ergebenden Folgeregler mit selbstoptimierung ist in Abbildung dargestellt. Anzumerken ist das die Verbesserung der Verstärkungen gedacht ist. Das bedeutet, die Qualität der Optimierung ist Abhängig von den zu Beginn übergebenen Parameter, die zum Beispiel über die Polvorgabe bestimmt worden sind.

## 5.6 Zustandsschätzung

Bei der Folgeregelung handelt es sich um einen Zustandsregler. Das bedeutet die Werte aller Zustände des realen Systems müssen bekannt sein. Betrachtet man das Stellgesetz der Regelung inklusive Vorsteuerung, bedeutet dies, das Position  $P$  und die Geschwindigkeit  $v$ , mit der sich der Quadrokobert im n-frame bewegt benötigt werden. Die Position  $p$  ist bereits über den Laser (Kaptiel) bestimmt. Gesucht ist noch die Geschwindigkeit  $v$ . Messtechnisch lässt sich diese nur sehr schwer ermitteln zum Beispiel über ein Dopplerradar. Dies würde

# PLATZHALTER GRAFIK

**Abbildung 5.12:** Folgeregler inklusive Selftuning

eine zusätzliche Sensor bedeutet, die wiederum mit kosten verbunden sind. Es sind also Methoden gefragt, die über die vorhandene Sensorik (IMU und Laser) die Geschwindigkeit ermitteln.

## 5.6.1 Geschwindigkeitbestimmung über die Inertialsenorik

Die *IMU* beinhaltet einen 3D-Bewegungsseonsor. Dieser nimmt die auf Beschleunigung in x, y und z- Achse des Quadrocopters auf und stellt mit einer Frequenz von 1kHz ( $T_{imu} = 1ms$ ) die Messwerte 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 gutem Ergebnis. Grund dafür ist das Sensorrauschen  $r_{a_k}$  sowie der Bias b, der trotz Intialisierung nicht vollständig zu eliminiert ist. Somit 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 daran, die Fehler werden mit integriert. Damit driften geschätzte Geschwindigkeit und reelle Geschwindigkeit auseinander. Somit ist eine Geschwindigkeitsbestimmung rein über die Inertialsensorik nicht möglich.

### 5.6.2 Geschwindigkeit als Ableitung der Position

Ebenfalls lässt sich die Geschwindigkeit aufgrund des Weg-Zeit-Gesetzes (5.5) über Ableitung der Quadrocopterpositionen bestimmen. Da es bei den Positionsdaten um diskrete Werte handelt, lässt sich die Ableitung und somit die Geschwindigkeit (5.35) anhand des Euler-Verfahren (5.34) approximieren.

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

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

Die Abtastzeit  $T_l = 25ms$  ( $f_l = 40Hz$ ). Dies entspricht der Updaterate mit der die Algorithmen zur Positionsbestimmung mit neuen Umgebungsscans des Lasers 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 Positionswert  $P_{k_{tat}}^n$  aufweist. Diese Abweichung ist nicht konstant und ist als Positionsräuschen  $r_{P_k}$  darzustellen.

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

Dieses Positionsräuschen  $r_P$  überträgt sich auf die Geschwindigkeit.

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

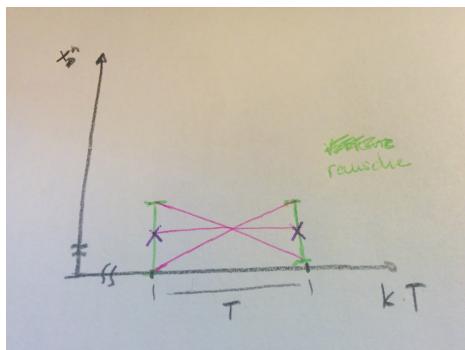
Diese führt jedoch nicht dazu, dass wie bei der Geschwindigkeitsbestimmung mittels der Beschleunigungssensoren, das Schätzwert und Realwert auseinander divergiert. Nichtsdestotrotz hat das Rauschen einen Einfluss auf die Qualität der Geschwindigkeitsberechnung. So fällt bei niedrigen Geschwindigkeiten der Abstand zwischen zwei Positionswerten 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 5.13a). Zwar zeigt Grafik 5.13b 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 mir geringer Geschwindigkeit den Einfluss des Positionsrauschen zu minimieren. Geht man davon aus, das 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 [5]. Die gefilterte Geschwindigkeit  $\hat{v}_k^n$  hängt dabei nicht nur von vorherigen Messwerten  $v_k^n$  sondern auch von den vorherigen Filterergebnissen.

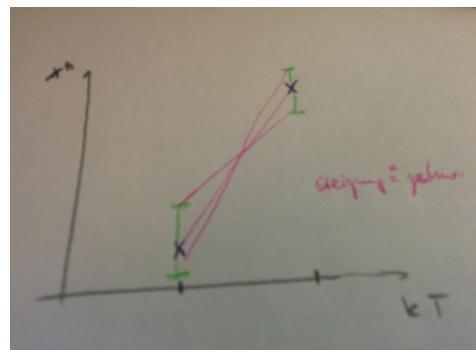
$$\hat{v}_k^n = \sum_{j=0}^n a_j \cdot v_{k-j}^n + \sum_{i=1}^n b_i \cdot \hat{v}_{k-i}^n \quad (5.38)$$

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 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 Stopbandeigenschaften 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



(a) Keine/kleine Positionsverschiebung, hohe Varianz



(b) Große Positionsverschiebung, geringe Varianz

**Abbildung 5.13:** Auswirkung der Größe der Positionsverschiebung innerhalb eines Schätzwertes auf die Varianz der Geschwindigkeitsschätzung

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 folge leisten kann.

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

Wie schon in Kaptiel 5.6.2 erkannt, nimmt die Varianz der Geschwindigkeitsschätzung ab je weiter die Positionsverteile auseinander liegen. Hierfür sei auf Grafik 5.13b 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 5.14. 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, 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 [7] wird dafür die Methode des *FOAW* vorgestellt. Diese Besagt, existiert eine Grade zwischen den Positionsverteilen  $p_k^n$  und  $p_{k-n}^n$  einer Achse, die alle  $n$  Punkte innerhalb

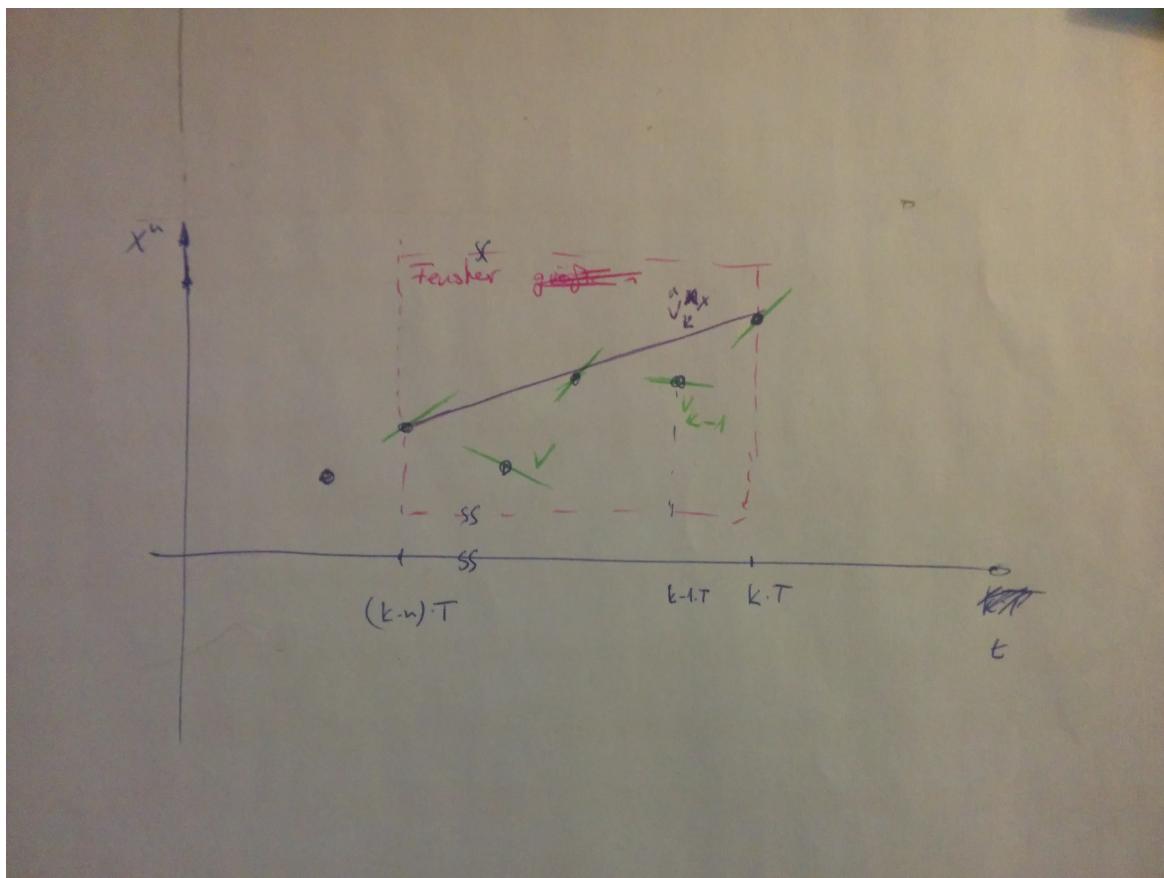


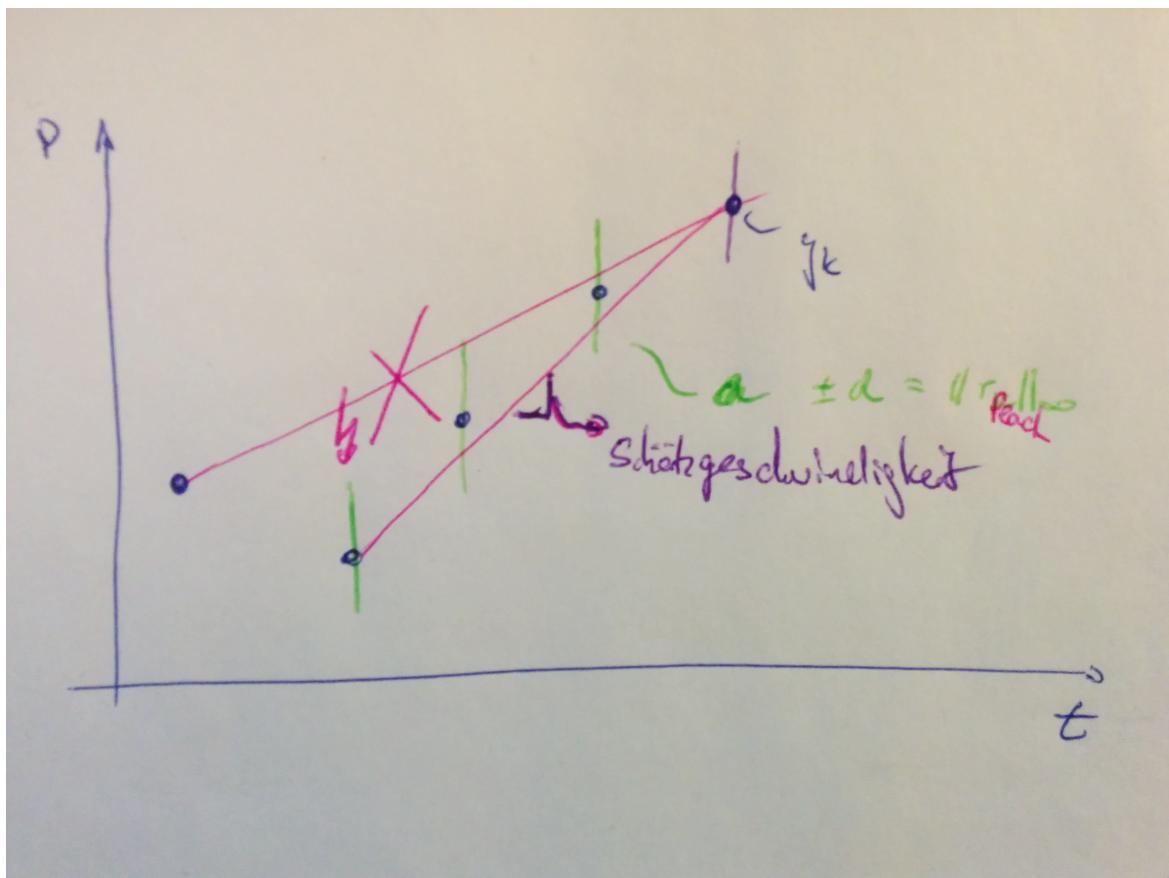
Abbildung 5.14: Fensterung

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.

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

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

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



**Abbildung 5.15:** End-fit First-Order Adaptive Windowing (FOAW)

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 [7] 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 5.16) dargestellt.

Die End-fit-FOAW verwendet für den Schätzvorgang die zwei Endpunkte des Fensters. In [7] 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 5.16:** Flussdiagramm FOAW

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

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

Nun ist  $b_n$  allgemeingültig für variable Fenstergrößenbestimmt.

---

## 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 29 und 36)
- [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: *Regelung nichtlinearer Systeme*. 2013  
(Zitiert auf Seiten 39 und 43)
- [5] GARDILL, Markus ; WEIGEL, Robert: *DES Digitale Elktrische Systeme*. 2014  
(Zitiert auf Seite 49)
- [6] HOFFMANN, Frank ; GODDEMEIER, Niklas ; BERTRAM, Torsten: Attitude estimation and control of a quadrocopter. In: *IROS'10*, 2010, S. 1072–1077 (Zitiert auf Seite 30)
- [7] JANABI-SHARIFI, Farrokh ; HAYWARD, Vincent ; J. CHEN, Chung shin: *Discrete-Time Adaptive Windowing for Velocity Estimation*. 2000  
(Zitiert auf Seiten 50, 53 und 54)
- [8] KALLWIES, Jan: *Höhenbestimmung und Höhenregelung bei einem Quadrocopter*, Diplomarbeit, 2013 (Zitiert auf Seite 31)
- [9] LIN, Feng ; BRANDT, Robert ; SAIKALIS, George: Self-Tuning of PID Controllers by Interaction. (2000) (Zitiert auf Seite 46)

- [10] 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)
- [11] LUNZE, Jan: *Regelungstechnik 1*. Springer London, Limited, 2008 (Springer-Lehrbuch Bd. 1). – ISBN 9783540689096 (Zitiert auf Seiten 36 und 40)
- [12] MAY, Stefan: *Skriptum AUT4 Mobile Robotik*. 2013 (Zitiert auf Seiten 24 und 25)
- [13] 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 19 und 27)
- [14] THIELECKE, Jörn: *Eingebettete Navigationssysteme*. 2014 (Zitiert auf Seite 16)

---

## 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  | Kaskadenstruktur   | 30 |
| 5.2  | Gesamtstruktur Regelung  | 31 |
| 5.3  | Gesamtstruktur Regelung  | 32 |
| 5.10 | Folgeregler  | 44 |
| 5.11 | Stabilitätsgebiet  | 44 |
| 5.12 | Selftuning Folgeregler   | 47 |
| 5.13 | Auswirkung der Positionsverschiebung auf Varianz der Geschwindigkeit | 49 |
| 5.14 | Fensterung   | 51 |
| 5.15 | End-fit FOAW   | 52 |

|                                     |    |
|-------------------------------------|----|
| 5.16 Flussdiagramm $FOAW$ . . . . . | 54 |
|-------------------------------------|----|

---

## Tabellenverzeichnis

---

# ANHANG A

---

## Anhang

---

FLACHHEIT NICHT LINEAER MEHRGRÖ?ENSYSTEME VON HERR DEUTSCHER