

**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
1.1	Software . . . . .	1
2	Systemarchitektur des Quadrocopters	2
2.1	Grundlegende Funktionsweise eines Quadrocopters . . . . .	2
2.2	Hardwareaufbau . . . . .	4
2.3	Softwarestruktur . . . . .	6
3	Grundlagen	9
3.1	Das Robot Operation System <i>Robot Operation System (ROS)</i> . . . . .	9
3.2	Einführung in die Koordinatensysteme und Koordinatentransformationen . .	11
3.2.1	Koordinatensysteme . . . . .	11
3.2.2	Koordinatentransformationen . . . . .	14
4	2D Positionsbestimmung	18
4.1	Projektion der Laserdaten in das o-frame auf der xy-Ebene des n-frames („laser_ortho_projector“) . . . . .	18
4.2	Positonsbestimmung über Scanmatching . . . . .	22
5	Positionsregelung	29
5.1	Struktur der Positionsregelung . . . . .	29
5.2	Modellbildung . . . . .	31
	Literaturverzeichnis	32
	Abbildungsverzeichnis	33



Tabellenverzeichnis	34
A Anhang	35

# KAPITEL 1

---

## Einleitung

---

### 1.1 Software

# 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.

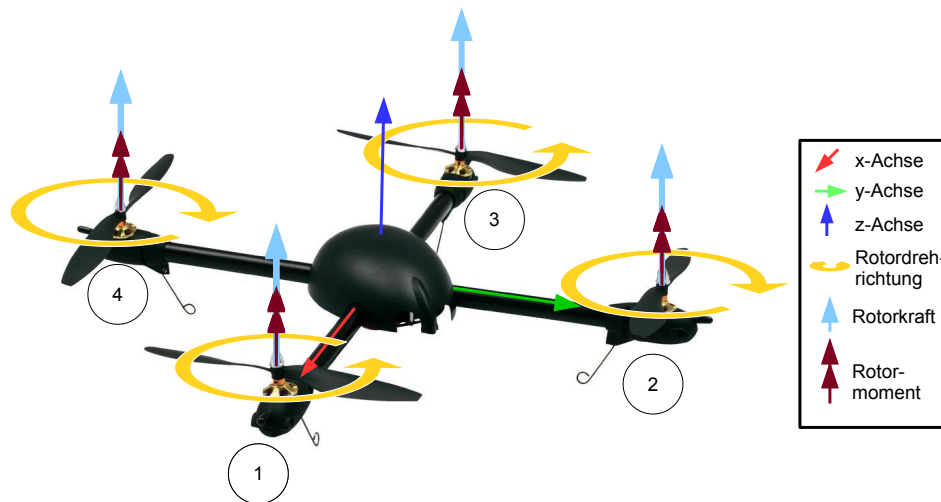
### 2.1 Grundlegende Funktionsweise eines Quadrocopters

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 Quadrocopter 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 Drehzahlenunterschied zwischen  $n_2$  und  $n_4$  verantwortlich für die Rotation.

Eine Änderung der Orientierung um die Hochachse  $z$ , sprich Änderung des Yaw-Winkels ist lässt sich ebenfalls über Variation der Rotordrehzahlen hervorrufen. Dabei kommt der Effekt zum tragen, das 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  $M^b$ .

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

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

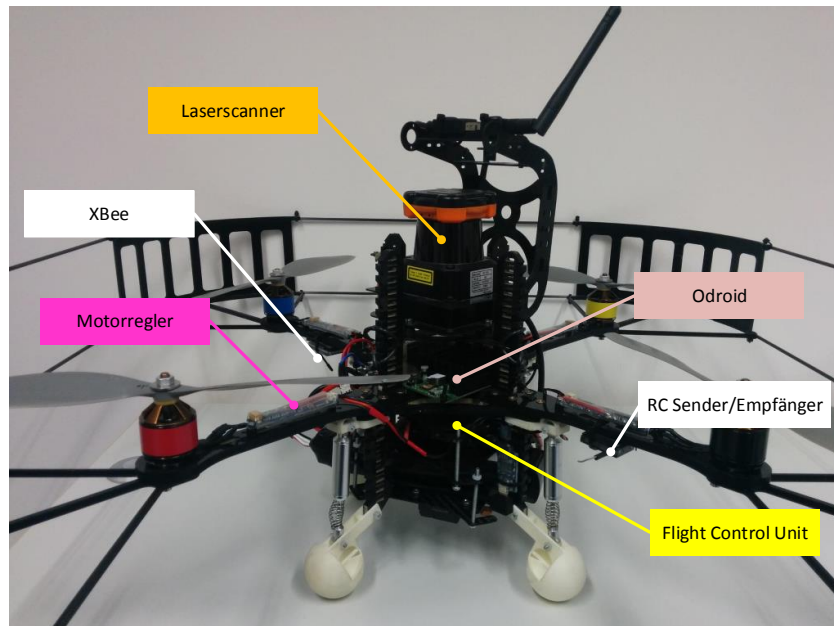
## 2.2 Hardwareaufbau

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 separate 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 inertielle 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 (Koordinatensysteme 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).

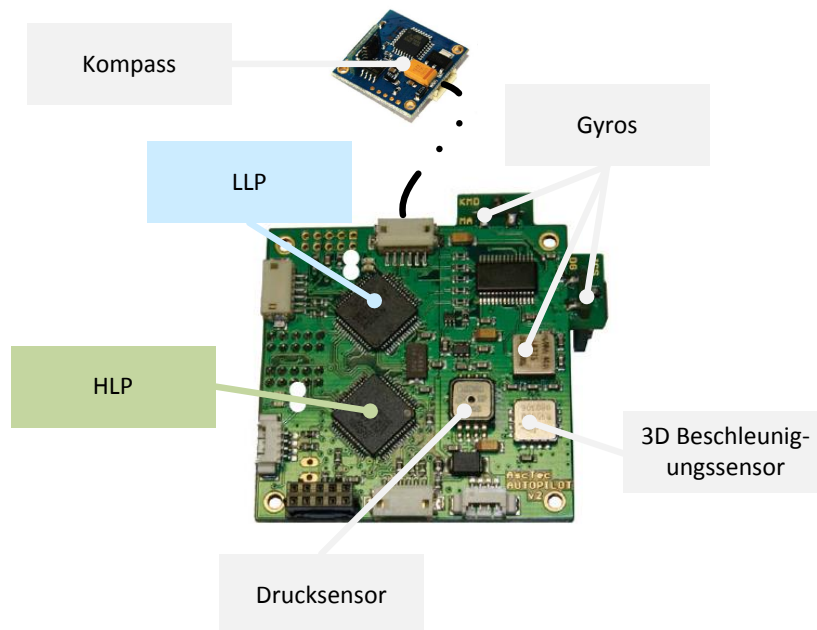
Da der Einsatzbereich im Indoorbereich liegt, ist Drucksensor zur Höhenbestimmung in geschlossenen Räumen nicht eignet, da er erst ab einer Höhe von 5m zuverlässige Werte



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

liefert. Daher wurde in einer vorangegangenen 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 Entfernungen 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 Inertialsensorik 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  $270^\circ$ . Die Umlaufdauer beträgt dabei 40 Hz, d.h. alle 25 ms steht ein neuer Umgebungssan 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 vorhergegangenen 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

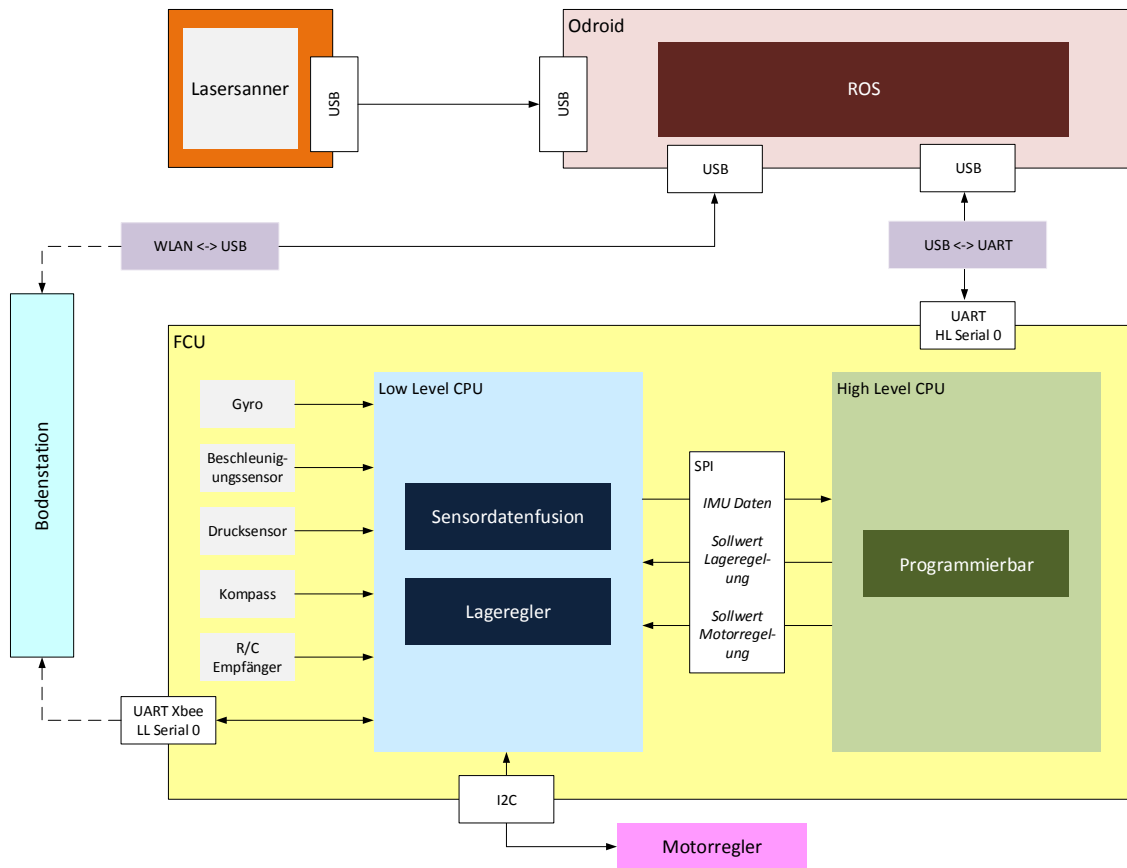
Protokolle die einzelnen Komponenten mit einander kommunizieren.

Beginnend mit der *FCU*, deren beiden Prozessoren *LLP* und *HLP* die mit einer Frequenz 1kHz getakten 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 benötigen 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

---

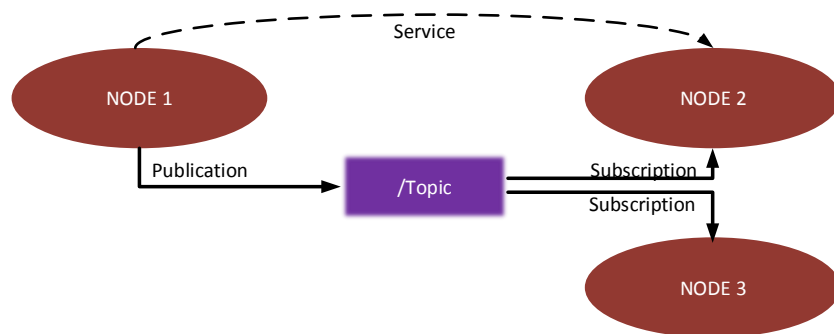
Das Kapitel Grundlagen behandelt die Themen, die in mehreren Abschnitten dieser Arbeit relevant sind. Dabei handelt es sich um das Robot Operation System, die verwendeten Koordinatensysteme und die Transformation zwischen ihnen.

### 3.1 Das Robot Operation System *ROS*

Ziel dieses Unterkapitel ist es das Opensource Betriebssystem *ROS* vorzustellen. Wie es aufgebaut ist und welche Vorzüge 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, das Komponenten unterschiedlicher Hersteller miteinander verbunden werden können. In unserem Fall Hokuyo Lasersanner 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. Knoten bezeichnet und läuft als eigener Prozess.

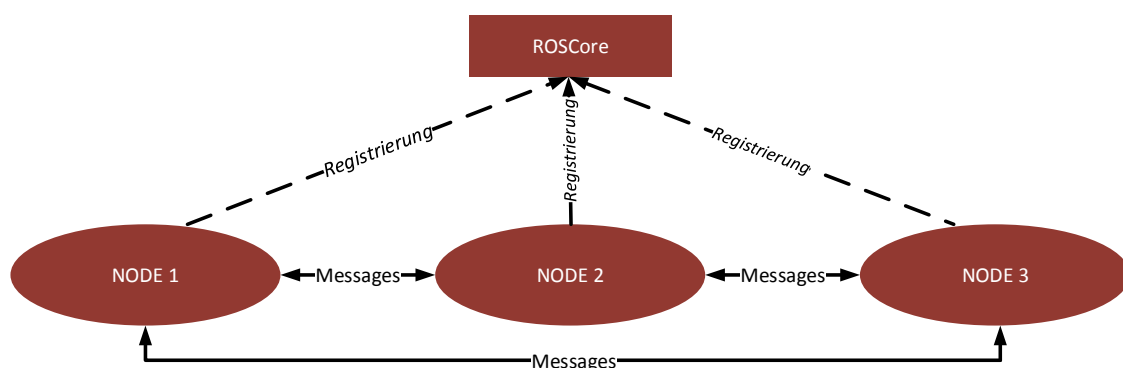
Der Austausch von Daten zwischen den Nodes erfolgt über so genannte Topics (Abbildung 3.1]. Dabei werden von den Knoten Nachrichten (engl. Messages) in Topics gepostet und somit veröffentlicht (publication). Benötigt ein weiterer Knoten den Inhalt dieses Topic



**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, bietet ROS die Möglichkeit einer Synchronen 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, dass durch das verwendete IP-Protokoll keine deterministische Versendung der Nachrichten nicht gewährleistet ist, da es sein kann, dass 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 zurückzugreifen, 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.

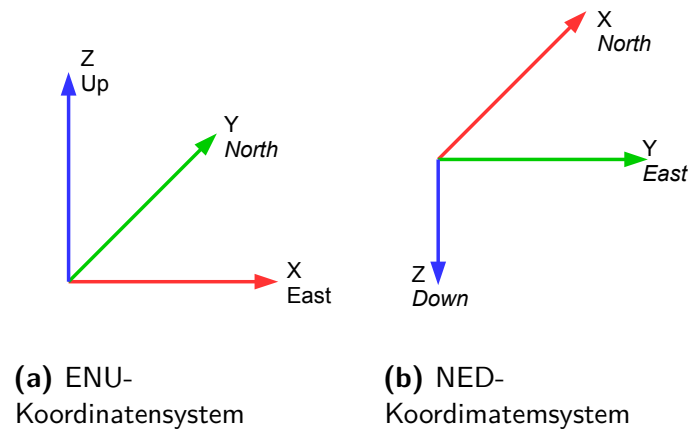
## 3.2 Einführung in die Koordinatensysteme und Koordinatentransformationen

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 dieses 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 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

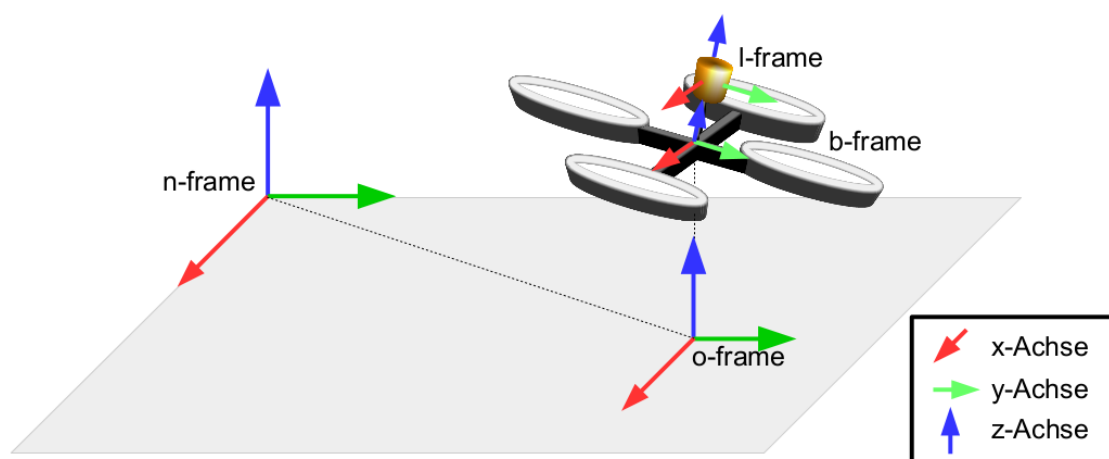


**Abbildung 3.3:** test

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 Koordinatensystem 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 Quadcopters verbunden. Man spricht dabei von einem 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 Quadrocoters 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.

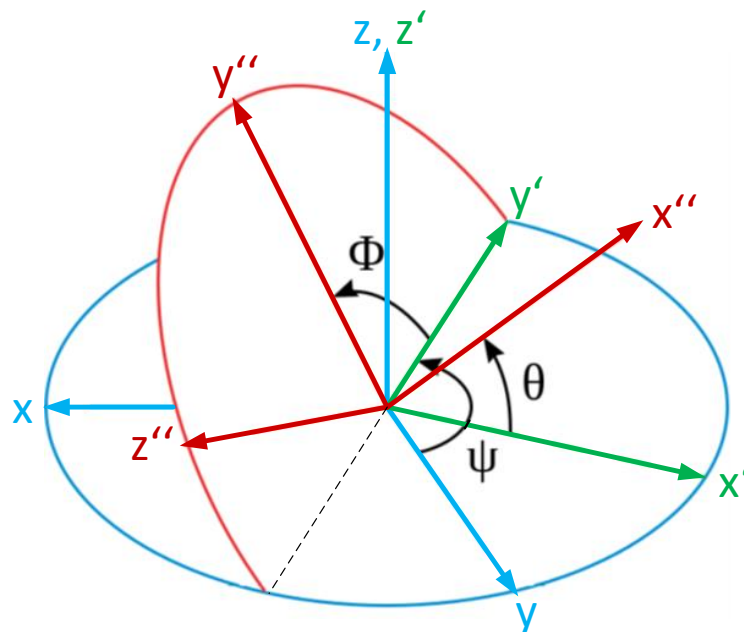
Da sich Messwerte wie zum Beispiel die *IMU*-Daten oder die Laserdaten auf unterschiedlichen Koordinatensysteme beziehen, benötigt man Koordinatentransformation(Kapitel 3.2.2).

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

### 3.2.2 Koordinatentransformationen

Damit Daten eines Referenzsystem in einen anderes 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 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 eingezeichnet 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 sein nochmal darauf hinzuweisen, das 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 ergibt sich:



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

Mit Hilfe dieser Transformationmatrix 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 Gesamttransformationmatrix 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, dass diese auf Grund der trigonometrischen Funktionen nur für Winkel  $\phi, \theta, \psi = \{x \in \mathbb{R} \mid -\pi \leq x \leq \pi\}$  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 ein vielfaches von  $2\pi$  eindeutig charakterisiert. Die genaue Definition findet sich in der Literatur [THIELECKE] und [YOUTUBE]. Da der 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, dass 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 Entfernungen sind dabei im l-frame definiert. Die Rohdaten enthalten somit keine Information über die Position des Quadrocopters im Navigationskoordinatensystem, sondern jегentlich 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).

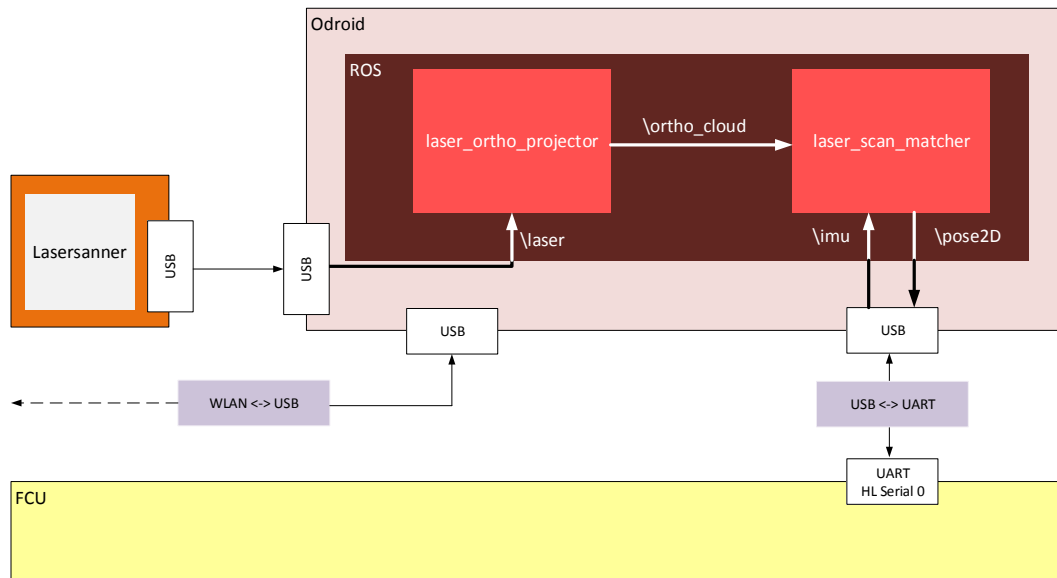
Realisiert sind diese Vorgänge in den von ROS zu Verfügung gestellte scan\_tools. Genauer gesagt handelt es sich dabei um dem „laser\_ortho\_projector“- und dem „laser\_scan\_matcher“-Knoten (Abbildung 4.1). Ziel der folgenden Kapitel ist es die Mathematik sowie die Funktionsweise die hinter diesen Algorithmen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

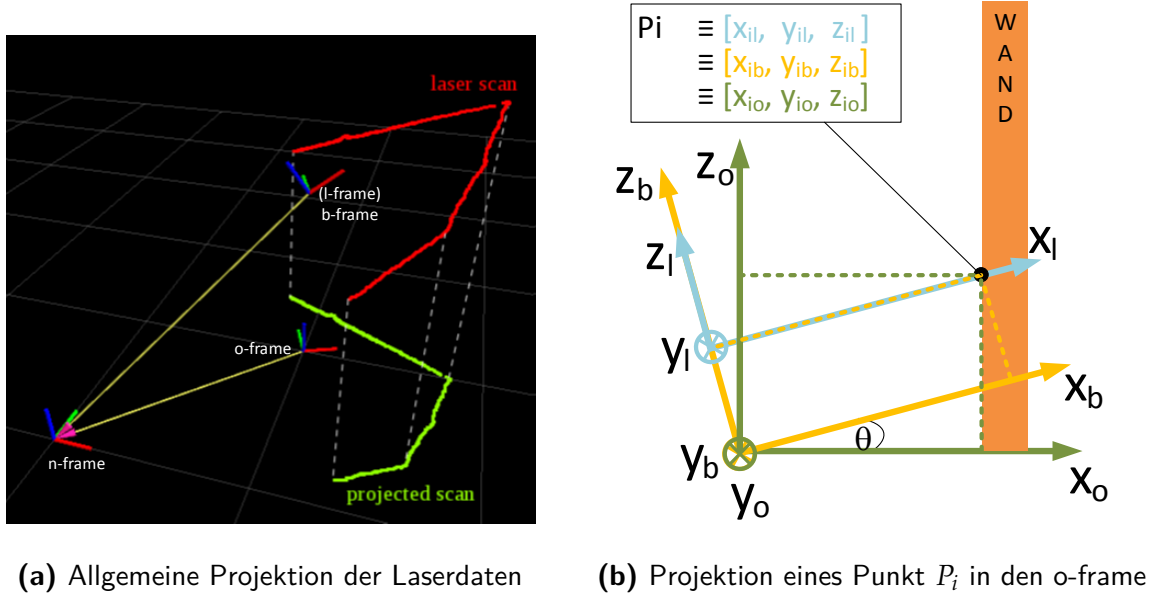
#### 4.1 Projektion der Laserdaten in das o-frame auf der xy-Ebene des n-frames („laser\_ortho\_projector“)



**Abbildung 4.1:** Verknüpfung des „laser\_ortho\_projector“- und des „laser\_scan\_matcher“-Knoten GRAFIK EVENTUELL 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 orthogonale 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 geschlossene Räume 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 Koordinatenursprungs des b-frames auf der z-Achse des o-frames hat demzufolge keinen Einfluss auf die Projektion. Folglich kann für beide Koordinatensysteme der identische Ursprung angenommen werden. Unter Beachtung dieser Annahmen kann der Einfluss des Roll- $(\phi)$  und Nickwinkels  $(\theta)$  auf die Entfernungsmessung eliminiert werden. Die Winkelgrößen liefert die IMU. Der mathematische Ablauf der orthogonalen Transformation wird basierend auf Literatur [6] 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

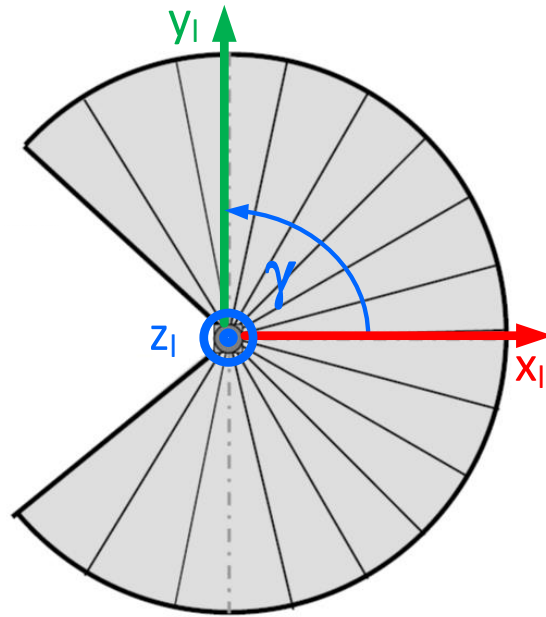
Entfernungsdaten eines Laserumlaufs besteht aus mehreren diskreten Abtastungen (Abbildung 4.3). Übergeben werden sie in Form von Entfernung  $r_i$  in einem Array (Topic \laser Abbildung 4.1). Mittels der Schrittweite von  $0.25^\circ$  lässt sich anhand des Indizes  $i$  für jeden Messpunkt einen Winkel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

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

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

$$M_{no} = \begin{bmatrix} \cos \theta \cdot r_i, & \sin \phi \sin \theta & \cos \phi \sin \theta \\ 0 & \cos \phi & -\sin \phi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix} \quad (4.4)$$

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

$$p_i^o = M_{no} \cdot P_i^b \quad (4.5)$$

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

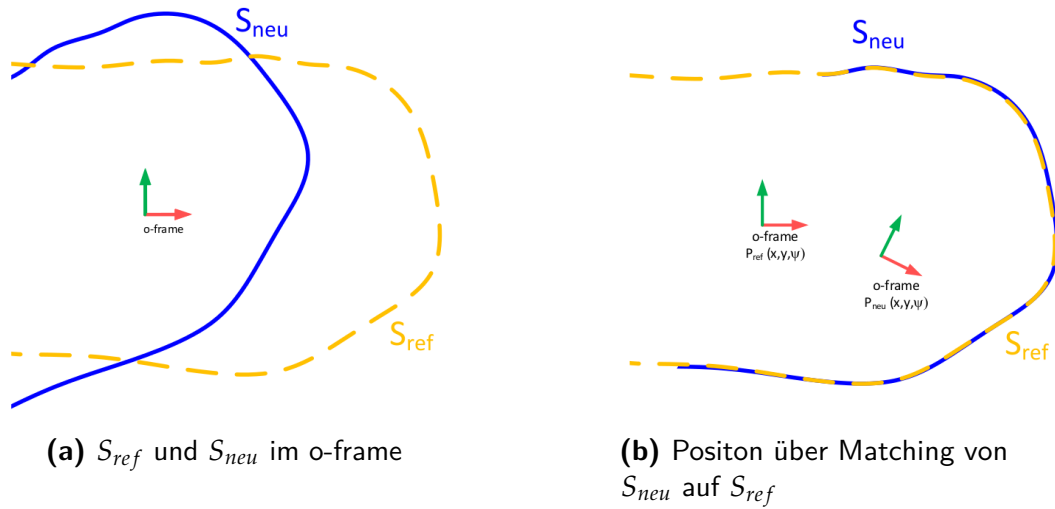
Aufbauend auf den im Kapitel 4.1 vorgestellten Laser\_ortho\_projektor, ist es Aufgabe des Folgenden Teilkapitels zu erläutern wie anhand eines Referenzscans  $S_{ref}$  und einem weiteren Scan  $S_{neu}$  die Position in der euklidischen xy-Ebene des n-frames bestimmt werden kann. Zur Anwendung kommt hier die Methode des Scanmatching. Dabei gilt die Annahme, dass 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  $M_z^0$  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.

- Vorgang 1, Punktekorespondenz.

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, d.h. jeden Punkt werden alle Punktestände 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.



▪ Vorgang 2, Bestimmung des Rotationswinkel  $\Delta\psi^o$  und der Translation  $T$

Wie schon zu Beginn angekündigt, soll eine rotatorische und translatorische Transformation zur Überlappung von  $S_{neu}$  mit  $S_{ref}$  führen. Idealerweise 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 Rotationsmatrix  $M_z^o$  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  $S_{neu}$  Indizes für Indizes vergleichbar. Der Grund warum in Vorgang 1 zu dem Scan  $S_{neu}$  ein korrespondierender 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 keinem eindeutigen Ergebnis führt. Laut [5] kann aus dieser Formel (4.11) als Fehlergleichung herangezogen werden. Mit dieser lässt sich über die least-squares 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 [1] 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 Fehlergleichung 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 Fehlergleichung 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 [1] 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.

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

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

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

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

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

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

draus 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.

- Vorgang 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.

- Vorgang 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

# PLATZHALTER GRAFIK

**Abbildung 4.5:** Flussdiagramm *ICP*-Algorithmus

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 Inertialsensorik 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 [6].

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

# 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 Fusionsalgorithmus sind im Paper [2] beschrieben. Aufgabe dieses Kapitels ist es die dort veröffentlichten Annahmen und Formeln zu verifizieren. Dies erfolgt über Literaturrecherchen und Herleitung der publizierten Gleichungen. Unter Berücksichtigung, dass die Position nun mehr vom Laser über die in Kapitel 4 vorgestellten Algorithmen bestimmt wird, ist Abschnitt 5.1 darauf ausgelegt eine Übersicht über die Bestandteile der Regelung zu geben. Herleitung der Komponenten erfolgt in den anschließenden Unterkapiteln.

### 5.1 Struktur der Positionsregelung

Das Hauptaugenmerk dieses Unterkapitels 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.

# PLATZHALTER GRAFIK

**Abbildung 5.1:** Reduzierte Kaskadenstruktur der Positonsregelung.

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  $O_s$  und Ist-Orientierung  $O$ , in Verbindung mit der Sollschiebungsvorgabe  $T_s$  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 Gyroscopes 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 [4] beschrieben. Die Ungewissheit über den Regleraufbau der Lageregelung stellt für den Entwurf der überlagerten Positionsregelung kein Problem dar. Von Interesse ist lediglich, das die Annahme einer sehr hohen Dynamik des Reglers zur einer Vereinfachung des für die Positionsregelung benötigten Modells (Kapitel 5.2) führt. Hohe Dynamik bedeutet, das der die Sollwinkel in einer sehr kurzen Zeit  $t \rightarrow 0$  s erreicht wird.

# PLATZHALTER GRAFIK

**Abbildung 5.2:** Struktur der Regelung

## 5.2 Modellbildung



---

## Literaturverzeichnis

---

- [1] *Skriptum AUT4 Mobile Robotik* (Zitiert auf Seiten 24 und 25)
- [2] 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 Seite 29)
- [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 28)
- [4] HOFFMANN, Frank ; GODDEMEIER, Niklas ; BERTRAM, Torsten: Attitude estimation and control of a quadrocopter. In: *IROS'10*, 2010, S. 1072–1077 (Zitiert auf Seiten )
- [5] 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)
- [6] 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 28)

---

## 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 . . . . .	27
5.1	Kaskadenstruktur . . . . .	30
5.2	Gesamtstruktur Regelung . . . . .	31

---

## Tabellenverzeichnis

---

# ANHANG A

---

## Anhang

---

Hier können weiterführende Grafiken, Codefragmente oder Ähnliches, das den Rahmen der Ausführung der eigentlichen Arbeit sprengen würde, hinzugefügt werden.