Bachelorarbeit
BSC-132

Mechanik und Meerestechnik

Bewegungsplanung und Regelung für autonomes Greifen mit einem Unterwasserroboter

Motion Planning and Control for Autonomous Grasping with an Underwater Robot

von
Jannik Jorge Grothusen

Betreuer: Prof. Dr.-Ing. R. Seifried
D.-A. Dücker, M.Sc.

Technische Universität Hamburg (TUHH)
Institut für Mechanik und Meerestechnik
Prof. Dr.-Ing. R. Seifried

Hamburg, Januar 2022

TUHH
Technische Universität Hamburg

Inhaltsverzeichnis

Kapitel 1

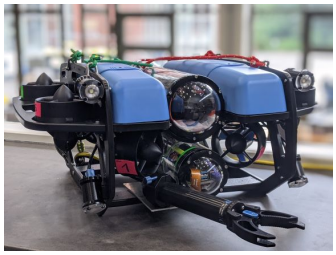
Einleitung

In den vergangenen Jahrzehnten haben sich die Einsatzbereiche von Robotern stark ausgeweitet. Die möglichen Anwendungen sind vielfältig und reichen von der Erforschung des Weltraums bis hin zur Untersuchung der Tiefsee. Vor allem in lebensgefährlichen Umgebungen wie unter Wasser werden Roboter vermehrt eingesetzt, um unter der Abwesenheit von atembarer Luft oder unter hohem Druck Aufgaben auszuführen. Sie werden beispielsweise zur Inspektion, Reparatur und Wartung von Unterwasserstrukturen zur Öl- oder Gasförderung aber auch bei Offshore-Windparks eingesetzt [SchjølbergEtAl16]. Andere Aufgaben sind auch die Untersuchung der Meeresbiologie, die Umweltüberwachung und die Kartierung des Meeresbodens [Antonelli14].

Missionen wie diese werden überwiegend von Menschen durch ferngesteuerte Unterwasserfahrzeuge (engl. Remotely Operated Vehicle, ROV) bzw. eine Untergruppe namens UVMS (von engl. Underwater Vehicle Manipulation System) durchgeführt. Der Begriff UVMS bezeichnet hierbei im Allgemeinen ein Unterwasserfahrzeug, welches mit Schnittstellen zur Interaktion mit der Umgebung, z. B. einem Greifarm, ausgestattet ist. Abbildung 1.1 zeigt eine Auswahl verschiedener UVMS. Sie sind in der Regel mit einer Kontrollstation an der Wasseroberfläche verbunden und werden manuell gesteuert. Automatische Steuerfunktionen oder Autonomie sind hierbei sehr selten [SchjølbergEtAl16]. Die Effizienz dieser Einsätze hängt von der Erfahrung und dem Können des Steuernden ab.

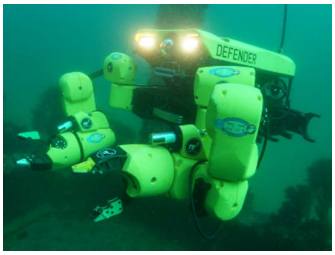
Autonomie bei Einsätzen mit Unterwasserrobotern ist ein wichtiger Schritt zur Steigerung der Effizienz und Sicherheit [SchjølbergEtAl16]. Hierbei müssen Problemstellungen der Navigation und Führung gelöst werden.

Der Beitrag dieser Arbeit besteht in der Entwicklung und Untersuchung eines Systems zur Bewegungsplanung und -regelung für autonome Greifprozesse mit einem Unterwasserroboter und kann in die Bereiche Mechanik und Regelungstechnik eingeordnet werden.



(a) BlueROV2 mit Greifarm.

(b) TRIDENT FP7 Projekt [RibasEtAl15].



(c) RE2 Sapien Sea Class [RE2Robotics21].

Abbildung 1.1: Auswahl verschiedener UVMS.

1.1 Zielsetzung

Diese Arbeit behandelt eins der ultimativen Ziele der Robotik: die Gestaltung von autonomen Robotern. Ein solches System führt Aufgaben ohne menschlichen Eingriff aus. In [Huang08] werden Stufen von Autonomität definiert. Volle Autonomität ist hier definiert als ein Operationsmodus, in dem ein unbemanntes System die ihm zugewiesene Aufgabe innerhalb eines festgelegten Bereichs ohne menschliches Eingreifen erfüllt und sich dabei an die Betriebs- und Umweltbedingungen anpasst. Die Aufgabe spezifiziert hierbei *was* der Roboter tut und weniger *wie* er dies tut [Latombe91].

Mit dieser Definition von Autonomität besteht die Zielsetzung dieser Arbeit darin, ein vollautonomes System zur Bewegungsplanung und -regelung für Greifprozesse mit einer vollaktuierten Unterwasserdrohne zu entwickeln. Hierfür soll ein vorhandenes Lokalisierungssystem genutzt werden, um das Objekt zu erkennen und die Fahrzeugpostion relativ zu ihm zu ermitteln. Abbildung 1.2 skizziert das zu lösende Problem.

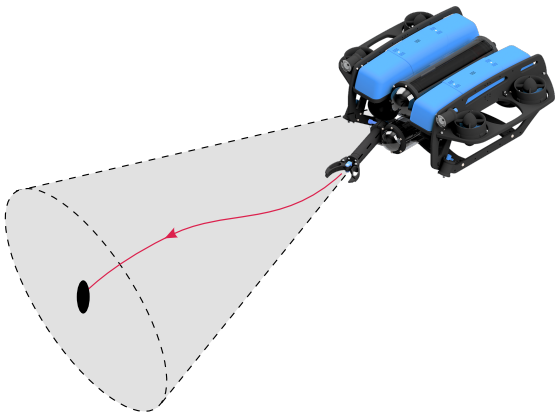


Abbildung 1.2: Vereinfachte Darstellung des Problems.

1.2 Gliederung der Arbeit

Zunächst wird in Kap. 2 ein Überblick über die Problemstellung des Greifens gegeben, sowie die benötigten Grundlagen zur Bewegungsplanung und -regelung eines Roboters erläutert. Anschließend wird in Kap. 3 das in dieser Arbeit untersuchte System, das BlueROV2 (siehe Abb. 1.1a), beschrieben. Die Architektur zur Bewegungsplanung und -regelung wird in Kapitel 4 entwickelt. Hierbei wird zu Beginn das zu lösende Problem abstrahiert und eine vorhandene Lokalisierungsmethode erklärt. Danach folgt die Erarbeitung der Pfadplanungs-, Pfadverfolgungs- und Regelungsalgorithmen. Die entwickelte Methode wird in Kap. 5 in das BlueROV2 integriert und untersucht. Kapitel 6 fasst die Ergebnisse dieser Arbeit zusammen und zeigt mögliche Weiterentwicklungen auf.

Kapitel 2

Grundlagen

Ein Roboter ist ein vielseitiges mechanisches System, ausgestattet mit Aktuatoren und Sensoren, welche von einem Rechensystem kontrolliert werden. Er kann verschiedenste Aufgaben lösen, welche meist Interaktion mit der Umgebung beinhalten. Dafür muss sich der Roboter in seiner Arbeitsumgebung bewegen. Seine Umgebung unterliegt den Gesetzen der Natur und kann andere physikalische Objekte beinhalten [Latombe91].

Dieses Kapitel gibt zunächst einen Überblick, wie ein Roboter mit seiner Umgebung interagieren kann. Anschließend wird die Bewegungsplanung eines Roboters behandelt. Hierbei wird die Darstellung im Konfigurationsraum eingeführt und das Pfadplanungsproblem erläutert. Abschließend wird eine allgemeine Systemarchitektur zur Bewegungsplanung und -regelung von Unterwasserfahrzeugen dargestellt.

2.1 Greifen und Manipulation

Menschen sind außerordentlich gut darin, Objekte mit verschiedenen Eigenschaften in ihrer Umgebung zu greifen und zu bewegen. Eine Maschine mit dieser Fähigkeit zu bauen ist schon lange ein Ziel in der Robotik. Die ersten Manipulatoren wurden bereits in den 1960er Jahren gebaut und haben sich seitdem stark weiterentwickelt. Zu Beginn führten sie sorgfältig vorgeschriebene Bewegungsabläufe aus, ohne sich an die Umgebung anzupassen. Mit der Zeit wurden Roboter dazu fähig Bewegungsabläufe automatisch zu generieren und feinfühlig zu handeln. Mittlerweile erledigen Roboter in vielen Bereichen verschiedenste Aufgaben. Dennoch ist die Fingerfertigkeit der menschlichen Hand bis jetzt unübertroffen [BillardKragic19].

Der Begriff Manipulation beschreibt in der Technik generell das physikalische Eingreifen in die Umgebung. Im Folgenden wird Manipulation auf das Bewegen oder Umorientieren eines Objekts in der Umgebung beschränkt [KuffnerXiao16]. Nach [Latombe91] kann bei klassischer Objekt-Manipulation zwischen dem Greifen, Transferieren und Positionieren unterschieden werden. Um ein Objekt zu manipulieren, nimmt ein Roboter physischen Kontakt mit dem Objekt auf und bewegt dieses anschließend durch die Ausübung von Kräften und Momenten. Diese Arbeit beschäftigt sich ausschließlich mit dem Greifen eines Objekts.

Greifen ist die Fähigkeit, ein Objekt in einer Pose (d. h. Position und Ausrichtung) relativ zu einer Hand zu fixieren [BicchiKumar00]. In der Robotik wird das Greifen durch eine Untergruppe der Endeffektoren realisiert. Sie werden typischerweise Hand oder Greifer genannt und beschreiben die Schnittstelle zwischen dem Roboter und der Umgebung [MurrayLiSastry94]. Greifer lassen sich anhand verschiedenster Merkmale klassifizieren. Nachfolgend wird basierend auf der Greifer-Konfiguration unterschieden. Greifer mit zwei oder drei Greifbacken sind oft speziell für eine Aufgabe bzw. ein Objekt entwickelt. Andere Greifer werden mit dem Ziel entwickelt, die menschliche Hand und ihre vielseitigen Fähigkeiten nachzuahmen. Abbildung 2.1a zeigt den in dieser Arbeit verwendeten Greifer und Abb. 2.1b einen für den Weltraumeinsatz entwickelten Greifer, welcher der menschlichen Hand ähnelt.

Zum Greifen muss zunächst eine zulässige Position des Greifers am Objekt gewählt werden. Dies bedingt, dass dem Roboter die Pose des Objekts bekannt ist. Die Greifposition muss zugänglich und stabil bzw. robust genug sein, um den externen Kräften des Greifers zu widerstehen [Latombe91]. Anschließend muss ein Pfad zu dieser Position generiert werden, was zu einer Problemstellung der Bewegungsplanung führt. Der Prozess der Bewegungsplanung wird in Abschnitt 2.2 näher erläutert. Der letzte Schritt beim Greifen ist die Fixierung des Objekts durch den Greifer mittels Form- oder Kraftschluss [KuffnerXiao16].

(a) Newton Subsea Gripper [BlueRobotics21].

(b) Dexhand, entwickelt am DLR [ChalonEtAl11].

Abbildung 2.1: Zwei verschiedene Greifer mit unterschiedlicher Komplexität.

2.2 Bewegungsplanung

Es ist eine fundamentale Aufgabe der Robotik, einen kollisionsfreien Pfad für den Roboter in seiner Umgebung zu planen. Obwohl dieses geometrische Problem simpel erscheint, ist seine Lösung sehr rechenaufwendig [KavrakiLaValle16]. Um die zentralen Fragestellungen der Bewegungsplanung zu beantworten, wird zunächst ein vereinfachtes Grundproblem nach [Latombe91] formuliert.

Mit dem Ziel, das physikalische Problem der Bewegungsplanung auf ein geometrisches Problem der Pfadplanung zu übertragen, werden nachfolgend einige Vereinfachungen eingeführt. Es wird angenommen, dass der Roboter das einzige Objekt ist, welches sich in seiner Arbeitsumgebung bewegt. Seine dynamischen Eigenschaften werden vernachlässigt, um temporale Einschränkungen zu umgehen. Des Weiteren wird der Roboter als Starrkörper angenommen und Kontaktkräfte mit der Umgebung werden zunächst vernachlässigt. Das Grundproblem der Bewegungsplanung lässt sich nun (ohne Berücksichtigung von Hindernissen) in der Arbeitsumgebung wie folgt formulieren:

> Der Roboter sei ein einzelner Starrkörper $\mathcal{R}$, welcher sich in seiner Arbeitsumgebung $\mathcal{W}$, dem Euklidischen Raum bewegt. $\mathcal{W}$ ist statisch und wird repräsentiert als $\mathbb{R}^n$ mit $n = 2$ oder $n = 3$. Die Geometrie von $\mathcal{R}$ und $\mathcal{W}$ ist vollständig bekannt und $\mathcal{R}$ ist nicht kinematisch eingeschränkt.
>
> Mit einer Anfangsposition und -orientierung und einer Zielposition und -orientierung von $\mathcal{R}$ in $\mathcal{W}$ soll ein Pfad generiert werden. Der Pfad spezifiziert eine kontinuierliche Sequenz an Positionen und Orientierungen von $\mathcal{R}$ beginnend an der Startposition und -orientierung und endend an der Zielposition und -orientierung.

Im Laufe dieser Arbeit wird gezeigt, dass diese Problemformulierung, obwohl sie stark vereinfacht ist, für das zu lösende Problem, ein Objekt mit dem verwendeten System zu greifen, ausreicht. Nachfolgend wird in diesem Abschnitt das Konzept des Konfigurationsraums eingeführt, um die verschiedenen Facetten der Bewegungsplanung zu abstrahieren. Anschließend wird ein Überblick über verschiedene Pfadplanungsmethoden gegeben.

2.2.1 Konfigurationsraum

Der Konfigurationsraum (engl. Configuration Space, C-Space) ist ein hilfreiches Werkzeug, um Problemstellungen der Bewegungsplanung einheitlich und präzise zu formulieren. Die zugrundeliegende Idee dieser Formulierung besteht darin, den

Roboter als Punkt in seinem Konfigurationsraum zu betrachten. Die Geometrie der Aufgabe des Roboters kann ebenfalls in diesem Konfigurationsraum dargestellt werden [Latombe91].

Eine der grundlegendsten Fragen, die zu einem Roboter gestellt werden kann, ist die nach seiner Position. Die Antwort auf diese Frage gibt seine Konfiguration $\boldsymbol{q}$. Sie spezifiziert die Position jedes Punktes der Robotergeometrie [KavrakiLaValle16]. Der Konfigurationsraum $\mathcal{C}$ enthält alle möglichen Konfigurationen $\boldsymbol{q} \in \mathcal{C}$. Seine Dimension entspricht der Anzahl an Freiheitsgraden des Roboters.

Zur Veranschaulichung wird das, zu Beginn dieses Kapitels eingeführte, Grundproblem der Bewegungsplanung im Konfigurationsraum formuliert. Als Arbeitsraum wird $\mathcal{W} = \mathbb{R}^2$ gewählt und somit ein Starrkörper in der Ebene betrachtet. Per Definition sind alle Punkte eines Starrkörpers bekannt, wenn seine Position und seine Orientierung bekannt sind. Eine mögliche Konfigurationsdarstellung wäre somit $\mathcal{C} = \mathbb{R}^2 \times \mathcal{S}^1$. Hierbei ist die Konfiguration durch eine Verkettung von Koordinaten in $\mathbb{R}^2$ und einem Winkel aus $\mathcal{S}^1$ gegeben.

Eine Bewegung des Roboters kann im Konfigurationsraum eindeutig über Pfade oder Trajektorien beschrieben werden. Dies wäre bei dem vorliegenden Beispiel im Arbeitsraum aufgrund einer fehlenden Dimension der Orientierung nicht möglich.

Ein Pfad $\Theta(s)$ bildet einen skalaren Pfadparameter s, welcher zu Beginn 0 und am Ziel 1 ist, auf einen Punkt beziehungsweise eine Konfiguration $\boldsymbol{q}$ im Konfigurationsraum $\mathcal{C}$ ab, $\Theta : [0, 1] \to \mathcal{C}$.

Bei einer Trajektorie $\Theta(s(t))$ ist der Parameter $s(t)$ zeitabhängig. Dies wird auch Zeitskalierung genannt, da jeder Zeit $t \in [0, T]$ ein Wert s zugeordnet wird, $s : [0, T] \to [0, 1]$. Hierbei ist T die gesamte Dauer der Bewegung.

2.2.2 Pfadplanung

Das Ziel der Bewegungsplanung ist es, unter Berücksichtigung von Randbedingungen, einen Pfad $\Theta(s)$ in dem Konfigurationsraum $\mathcal{C}$ zu finden, welcher die Zielkonfiguration des Roboters erreicht. Hierbei können verschiedenste Randbedingungen für die Arbeitsumgebung oder den Roboter gelten. Der Arbeitsraum kann durch Hindernisse und die Bewegungsmöglichkeiten des Roboters durch holonome und nicht-holonome Zwangsbedingungen eingeschränkt sein.

Es gibt verschiedene Planungsansätze, die unter diesen Randbedingungen und anderen Kriterien, wie der Zeit, einen validen Pfad generieren. Sie sind je nach Randbedingungen unterschiedlich komplex. Für Arbeitsumgebungen mit Hindernissen kann bspw. ein Potentialfeldverfahren angewendet werden [Latombe91].

Hierbei werden die Zielkonfiguration als attraktives und die Startkonfiguration, sowie Hindernisse als abstoßendes Potential definiert. Der Pfad wird generiert, indem der Roboter als punktförmige Masse behandelt wird, welche unter dem Einfluss des Potentialfelds steht. Für weitere Erklärungen zu Pfadplanungsmethoden in einem Arbeitsraum mit Hindernissen sei auf [KavrakiLaValle16] und [Latombe91] verwiesen.

Pfadplanungsverfahren sind einfacher, wenn keine Hindernisse in der definierten Arbeitsumgebung vorhanden sind. Das Ziel der Pfadplanung besteht dann darin, die Start- und Endkonfiguration nur unter Berücksichtigung der Zwangsbedingungen des Roboters zu generieren. Zur Lösung des Problems lassen sich oft einfache mathematische Zusammenhänge formulieren. [PiazziGuarinoRomano07] schlägt hierfür Polynome siebter Ordnung vor. In [ChoiCurryElkaim08] wird ein Planungsalgorithmus basierend auf Bézierkurven präsentiert. Vor- und Nachteile dieser Verfahren werden in Abschnitt 4.3.2 aufgezeigt.

2.3 Führung, Navigation und Regelung

Damit Bewegungen von einem Roboter bzw. einem Fahrzeug ausgeführt werden können, ist ein System zur Bewegungssteuerung notwendig. Ein solches System besteht bei Unterwasserfahrzeugen üblicherweise aus drei Komponenten, welche als Führung, Navigation und Regelung (engl. Guidance, Navigation, and Control, GNC) bezeichnet werden [Fossen11]. Das in Abb. 2.2 dargestellte System nutzt zur Führung und Regelung Zustandsschätzungen und wird daher als System mit geschlossenem Kreis (engl. Closed-Loop System) bezeichnet. Die Komponenten der GNC-Architektur werden im Folgenden erklärt.

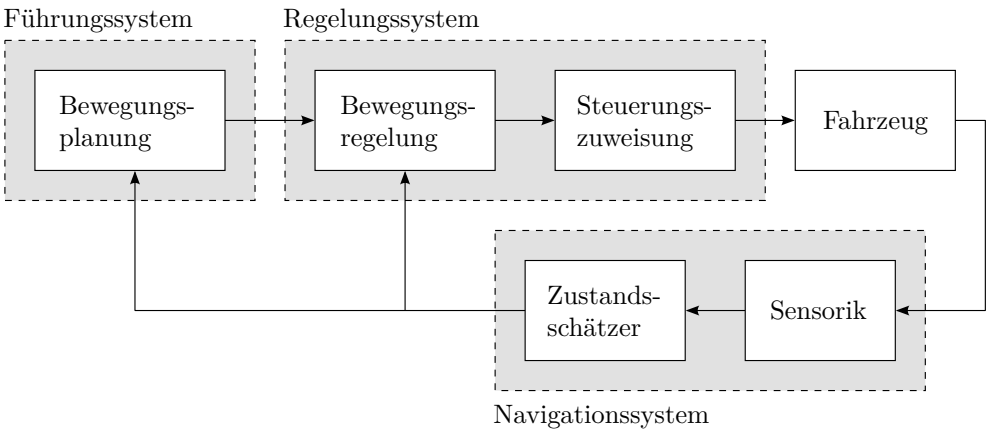


Abbildung 2.2: Vereinfachter Signalfluss einer closed-loop GNC-Architektur.

Navigation

Das Navigationssystem ist für die Bestimmung des Fahrzeugzustands in einem Referenzsystem verantwortlich. Dieser Zustand wird von [Fossen11] definiert als Position, Ausrichtung, Geschwindigkeit und Beschleunigung des Fahrzeugs. Zur Bestimmung des Zustands werden Sensordaten mit einem Zustandsschätzer gefiltert. Der verbreitetste Zustandsschätzer ist hierbei das Kalman Filter. Der ermittelte Fahrzeugzustand wird in einer closed-loop Architektur sowohl an die Führung als auch an die Regelung übergeben.

Führung

Das Führungssystem plant auf Basis seiner Aufgabe und des Fahrzeugzustands mit welcher Bewegung diese Aufgabe ausgeführt wird. Hierbei stellt es Referenzposition, -geschwindigkeit oder -beschleunigung für die Regelung bereit. Die Aufgabe wird entweder von einem Menschen oder dem Führungssystem selbst definiert. Eine typische Aufgabe der Führung ist es, einen Pfad und die zugehörigen Referenzsignale zu berechnen [Fossen11].

Regelung

Im Regelungssystem werden geeignete Stellgrößen für die Aktuatoren des Fahrzeugs berechnet. Hierzu zählt einerseits die Regelung mit Steuerungs- (engl. Feedforward Control) und Regelungsgesetzen (engl. Feedback Control) und andererseits die Zuweisung der berechneten Stellgrößen zur Aktuatorik des Fahrzeugs, um geeignete Kräfte und Momente zu verursachen. Das Ziel der Bewegungsregelung ist es, die Referenzgrößen des Führungssystems zu erreichen.

Kapitel 3

Systembeschreibung

Als Basis für die Entwicklungen in dieser Arbeit dient das BlueROV2 von der Firma BlueRobotics. Es handelt sich um eine kommerziell erwerbbare Unterwasserdrohne, welche wegen ihrer open-source Elektronik und Software eine gute Grundlage für Entwicklungen und Erweiterungen bildet [BlueRobotics21]. In diesem Kapitel wird zunächst auf die mathematische Beschreibung von Unterwasserfahrzeugen am Beispiel des BlueROV2 eingegangen. Anschließend wird ein Überblick über die Hard- und Software-Architektur des Unterwasserroboters gegeben.

3.1 Mathematisches Modell

In diesem Abschnitt wird das mathematische Modell für Unterwasserfahrzeuge nach [Fossen94] eingeführt. Allgemein betrachtet die Bewegungslehre sechs Freiheitsgrade, da sechs unabhängige Koordinaten nötig sind, um die Position und Orientierung eines Starrkörpers im Raum zu beschreiben. Die ersten drei Koordinaten und ihre Ableitungen nach der Zeit entsprechen der Position und translatorischen Bewegung entlang der x-, y- und z-Achsen. Die letzten drei Koordinaten und ihre Ableitungen nach der Zeit werden genutzt, um die Orientierung und Rotationen um die x-, y- und z-Achsen zu beschreiben. Eine Übersicht zur Notation für Wasserfahrzeuge nach [SNAME50] bietet Tab. 3.1.

Für die Beschreibung der Bewegung eines Wasserfahrzeugs bietet es sich an, zwei rechtshändige Koordinatensysteme (KOS) zu definieren: ein raumfestes KOS $\{w\} : \{O_w, x_w, y_w, z_w\}$ und ein körperfestes KOS $\{b\} : \{O_b, x_b, y_b, z_b\}$. Obwohl in der Fachliteratur für Unterwasser-Anwendungen die z-Richtung meist nach unten definiert ist, wird sich in dieser Arbeit für die intuitive Darstellung entschieden, dass die z-Achse in positiver Richtung nach oben zeigt. Der Ursprung

Tabelle 3.1: Notation für Wasserfahrzeuge nach [SNAME50].

	Kräfte und Momente	Geschwindigkeiten	Positionen und Kardanwinkel
Bewegung in x-Richtung	X	u	x
Bewegung in y-Richtung	Y	v	y
Bewegung in z-Richtung	Z	w	z
Rotation um x-Achse	K	p	ϕ
Rotation um y-Achse	M	q	θ
Rotation um z-Achse	N	r	ψ

O_b von $\{b\}$ wird im Massenschwerpunkt des Fahrzeugs definiert und die Achsen x_b, y_b und z_b werden für Wasserfahrzeuge so gewählt, dass sie mit den Hauptträgheitsachsen übereinstimmen. Der Koordinatenursprung von $\{w\}$ liegt an der Wasseroberfläche. Die in dieser Arbeit verwendeten Koordinatensysteme und Bewegungsrichtungen sind in Abb. 3.1 dargestellt.

Da im Verlauf dieser Arbeit verschiedene Koordinatensysteme verwendet werden, wird an dieser Stelle eine Präzisierung der Vektornotation vorgenommen. Ein Vektor $\boldsymbol{v}$, welcher im Koordinatensystem $\{a\}$ ausgedrückt ist, wird als $\boldsymbol{v}^a$ geschrieben.

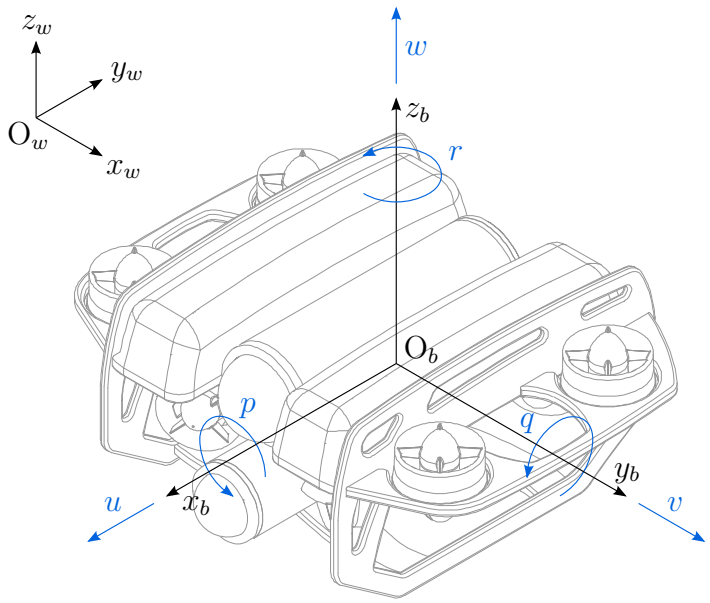


Abbildung 3.1: Kinematik am Beispiel des BlueROV2.

Generell kann die Bewegung eines Wasserfahrzeugs in 6 Freiheitsgraden beschrieben werden als

$$\boldsymbol{\eta} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^\top, \tag{3.1}$$

$$\boldsymbol{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^\top, \tag{3.2}$$

$$\boldsymbol{\tau} = \begin{bmatrix} X & Y & Z & K & M & N \end{bmatrix}^\top. \tag{3.3}$$

Hierbei bezeichnet $\boldsymbol{\eta}$ die Position und Orientierung im raumfesten KOS $\{w\}$. Der Vektor $\boldsymbol{\nu}$ beschreibt die Linear- und Winkelgeschwindigkeiten und $\boldsymbol{\tau}$ die auf das Fahrzeug wirkenden Kräfte und Momente im körperfesten KOS $\{b\}$.

Auf die Herleitung der kinematischen Beziehungen wird in diesem Abschnitt aufgrund der geringen Relevanz für diese Arbeit nicht explizit eingegangen. [Fossen94] zeigt, dass sich die nichtlinearen Bewegungsgleichungen in einem körperfesten Koordinatensystem bei konstanten Strömungen in sechs Freiheitsgraden formulieren lassen als

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau}, \tag{3.4}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}. \tag{3.5}$$

Hierbei ist $\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A$ die Summe aus der Masse des Starrkörpers $\boldsymbol{M}_{RB}$ und der hydrodynamischen Zusatzmasse $\boldsymbol{M}_A$. Die Zusatzmasse wird zu dem System hinzugefügt, da ein beschleunigter oder abgebremster Körper ein gewisses Volumen der ihn umgebenden Flüssigkeit bewegen muss, während er sich durch diese hindurch bewegt. $\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu})$ beschreibt die Coriolis- und Centripetalkräfte, welche durch Rotation verursacht werden. Sie sind durch die Starrkörper- und virtuelle Masse bedingt. $\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D} + \boldsymbol{D}_n(\boldsymbol{\nu})$ beschreibt die hydrodynamische Dämpfung des Systems. Hierbei steht $\boldsymbol{D}$ für lineare Dämpfung und $\boldsymbol{D}_n(\boldsymbol{\nu})$ für nichtlineare Dämpfung durch quadratische Dämpfung oder Terme höherer Ordnung. Die Kinematikmatrix $\boldsymbol{J}(\boldsymbol{\nu})$ überträgt den Vektor der Geschwindigkeiten $\boldsymbol{\nu}$ im körperfesten KOS $\{b\}$ abhängig von der Orientierung des Starrkörpers (ϕ, θ, ψ) zu $\dot{\boldsymbol{\eta}}$. Diese Transformation gilt nur für die translatorischen Geschwindigkeiten.

Der Vektor $\boldsymbol{g}(\boldsymbol{\eta})$ beschreibt Rückstellkräfte und -momente. Zu diesen Kräften gehören die Gewichtskraft $\boldsymbol{f}_g^w$, welche im Massenschwerpunkt (engl. Center of Gravity, CG) des Fahrzeugs angreift und die Auftriebskraft $\boldsymbol{f}_b^w$, welche im Formschwerpunkt bzw. Auftriebspunkt (engl. Center of Buoyancy, CB) des Fahrzeugs angreift. Die Kräfte wirken nur in vertikaler Richtung, also der z_w-Achse von $\{w\}$ und können nach [SNAME50] geschrieben werden als

$$\boldsymbol{f}_g^w = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \quad \text{und} \quad \boldsymbol{f}_b^w = \begin{bmatrix} 0 \\ 0 \\ \rho g V \end{bmatrix}. \tag{3.6}$$

Hierbei ist m die Masse des Fahrzeugs, V das durch das Fahrzeug verdrängte Flüssigkeitsvolumen, g die Erdbeschleunigung (negative z_w-Richtung) und ρ die Dichte der verdrängten Flüssigkeit.

Die Rückstellkräfte haben einen Einfluss auf die Stabilität des Fahrzeugs. Nach [SNAME50] ist ein Körper, welcher im Gleichgewicht schwimmt (entweder eingetaucht oder an der Oberfläche einer Flüssigkeit) metazentrisch stabil, wenn er bei einer Auslenkung aus dem Gleichgewicht in der horizontalen Ebene (durch Nick- oder Rollbewegung) in die Ausgangslage zurückkehrt. Liegt der Massenschwerpunkt in der aufrechten Lage unterhalb des Formschwerpunkts, so entsteht in jedem Fall (eine Ausnahme bildet die labile Gleichgewichtslage) ein rücktreibendes Moment.

Das BlueROV2 wurde so entwickelt, dass es metazentrisch stabil ist, also der Formschwerpunkt über dem Gewichtsschwerpunkt liegt. In [Mogk20] wurden mit Hilfe von CAD-Software die Positionen des Massenschwerpunktes $\boldsymbol{r}^b_{CG}$ und Formschwerpunktes $\boldsymbol{r}^b_{CB}$ bestimmt. Die Punkte liegen in $\{b\}$ bei

$$\boldsymbol{r}^b_{CG} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top \quad \text{und} \quad \boldsymbol{r}^b_{CB} = \begin{bmatrix} 0 & 0 & 27,6\text{mm} \end{bmatrix}^\top . \tag{3.7}$$

Die Schwimmkörper am Fahrzeug verlagern den Formschwerpunkt durch ihr Volumen nach oben. Gleichzeitig haben sie nur geringen Einfluss auf den Massenschwerpunkt, da sie aus Schaumstoff bestehen, dessen Dichte um den Faktor drei geringer ist, als die von Wasser [BlueRobotics21]. Abbildung 3.2 veranschaulicht die metazentrische Stabilität am Beispiel des BlueROV2.

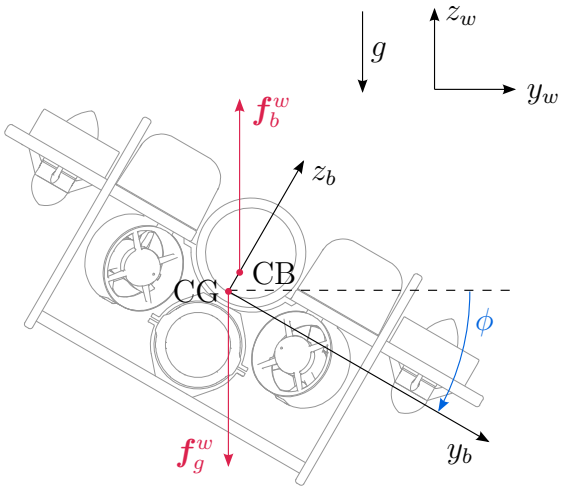


Abbildung 3.2: Darstellung der lateralen metazentrischen Stabilität des BlueROV2. Das Gleiche gilt in transversaler Richtung.

3.2 Hardware-Architektur

Das BlueROV2 wird als ROV vertrieben und ist somit primär für den teleoperativen Einsatz gedacht. Durch wenige Modifikationen kann das Fahrzeug jedoch auch autonom betrieben werden. Nachfolgend werden die wichtigsten Hardware-Komponenten des Systems beschrieben. Abbildung 3.5 stellt die Hardware-Architektur vereinfacht dar.

Computer

In dem BlueROV2 sind drei miteinander verbundene Einplatinencomputer verbaut. Zwei RaspberryPi 4B und ein Pixhawk 4. Der Pixhawk 4 dient als Flugcomputer (engl. Flight Control Unit, FCU) und erfüllt verschiedene Aufgaben. Einerseits beinhaltet er diverse Sensorik, wie eine Inertiale Messeinheit (engl. Inertial Measurement Unit, IMU), welche aus einem Beschleunigungssensor und einem Gyroskop besteht, ein Magnetometer und ein Barometer. Das Barometer ist jedoch für den Einsatz an der Luft gedacht und kann daher im Gehäuse des BlueROV2 nicht sinnvoll verwendet werden. Andererseits ist der Pixhawk 4 auch für die Ansteuerung der Antriebspropeller mittels Pulsweitenmodulation (PWM) verantwortlich. Die RaspberryPi 4B werden als begleitende Computer (engl. Companion Computer) genutzt und bilden eine Schnittstelle zu verschiedenen Hardware-Komponenten, siehe Abb. 3.5.

Aktuatorik

Das in dieser Arbeit untersuchte BlueROV2 ist auf die sogenannte *Heavy Configuration* aufgerüstet. Es sind acht Antriebspropeller vom Modell T200 der Firma BlueRobotics verbaut, mit denen sich das Fahrzeug über alle seiner sechs Freiheitsgrade kontrollieren lässt. Abbildung 3.4 zeigt die Anordnung der Propeller am Fahrzeug. Jeder Antriebspropeller wird mit Hilfe von einem elektronischen Geschwindigkeitsregler (engl. Electronic Speed Controller, ESC) über ein PWM-Signal vom Pixhawk 4 angesteuert.

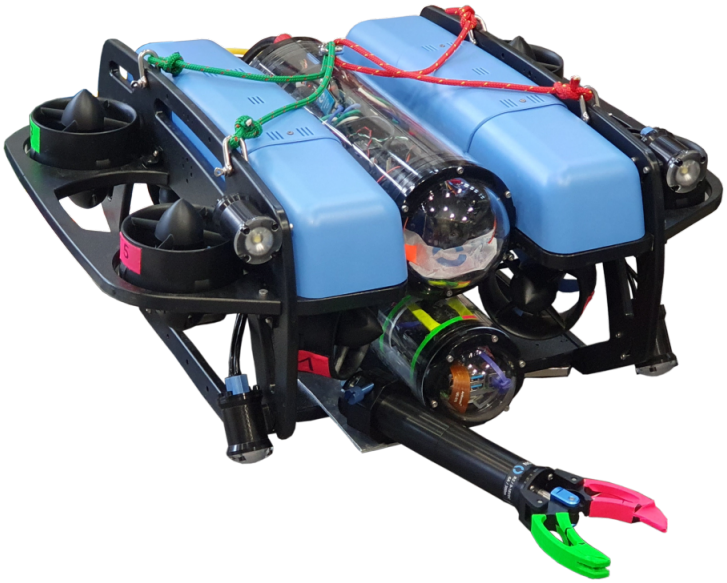


Abbildung 3.3: BlueROV2 in der verwendeten Konfiguration. Der Greifarm ist starr mit dem restlichen Fahrzeug verbunden. Die Greiferklauen sind zur leichteren Erkennbarkeit unter Wasser farblich markiert.

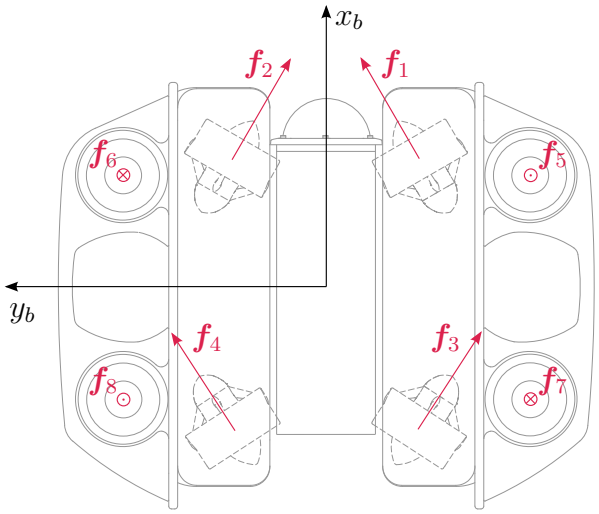


Abbildung 3.4: Konfiguration der Antriebspropeller im BlueROV2 Heavy.

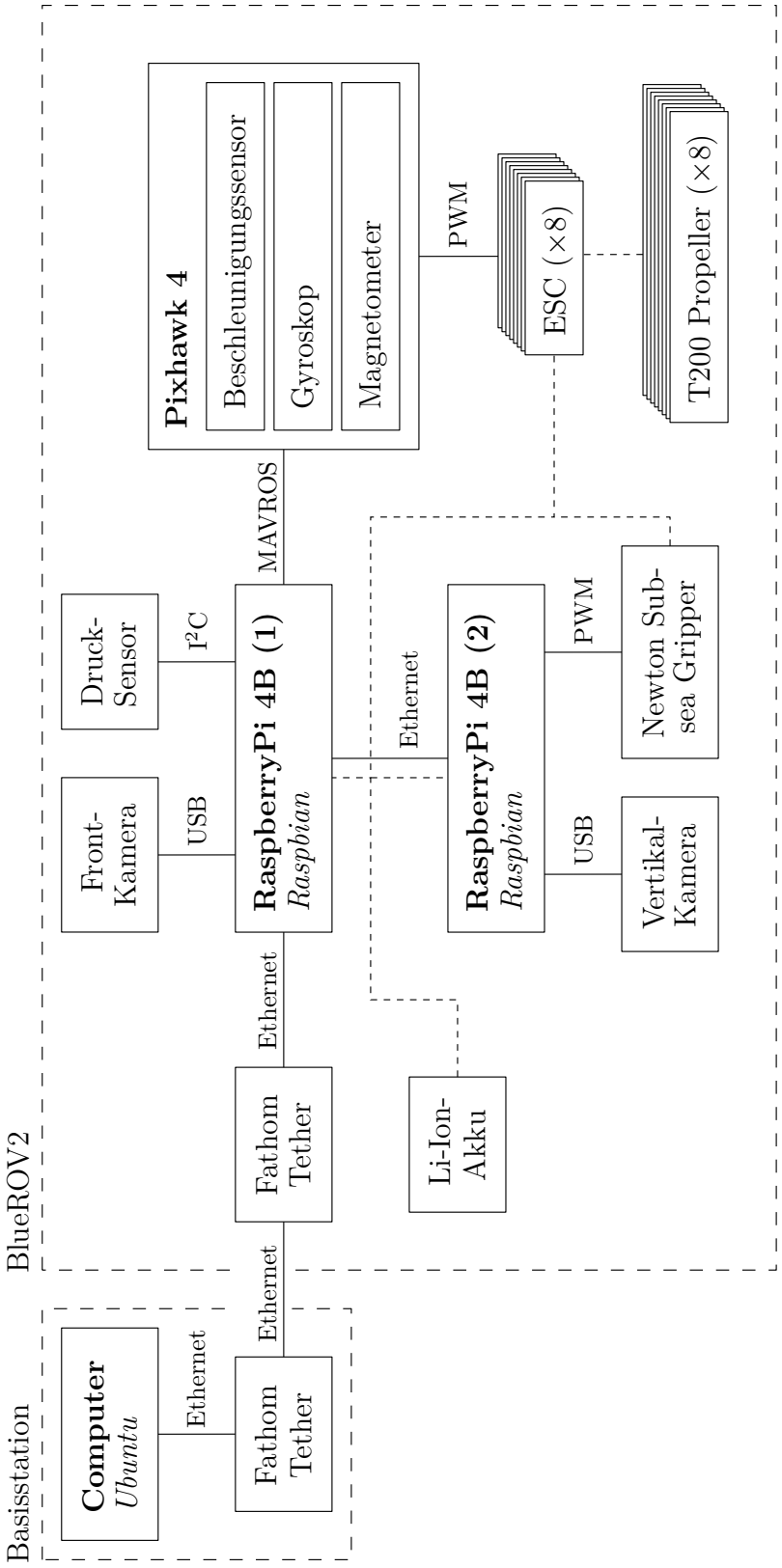


Abbildung 3.5: Übersicht der Hardware-Architektur im verwendeten System. Durchgezogene Verbindungslinien repräsentieren Informationsfluss. Gestrichelte Verbindungslinien repräsentieren Energiefluss.

Sensorik

Neben der Sensorik vom Pixhawk 4 (IMU, Magnetometer und Barometer) sind zwei Kameras verbaut. Sie sind jeweils an einen der RaspberryPi 4B angeschlossen. Eine ist mit dem Bildwinkel in positiver x_b-Richtung und eine mit dem Bildwinkel in negative z_b-Richtung orientiert. Sie werden nachfolgend als Frontkamera C_f bzw. Vertikal-Kamera C_v bezeichnet. Zusätzlich ist ein zweites Barometer an einen der RaspberryPi Computer angeschlossen. Die Positionen und Orientierungen der Kameras und des Drucksensors können Tab. 3.2 entnommen werden und sind in Abb. 3.6 dargestellt.

Greifarm

Der verbaute Greifarm *Newton Subsea Gripper* ist ebenfalls von der Firma BlueRobotics. Er wird mit Hilfe eines Adapters starr mit dem Rahmen des BlueROV2

Tabelle 3.2: Positionen und Orientierungen von Sensoren und Endeffektor.

Positionen und Eulerwinkel	Greifer	Front-kamera	Vertikal-Kamera	Barometer
x^b [m]	$0,4$	$0,16$	$0,15$	$-0,17$
y^b [m]	0	0	0	0
z^b [m]	$-0,08$	$0,05$	$-0,08$	$0,09$
ϕ^b [°]	0	0	0	0
θ^b [°]	0	0	0	0
ψ^b [°]	0	$11,9$	90	0

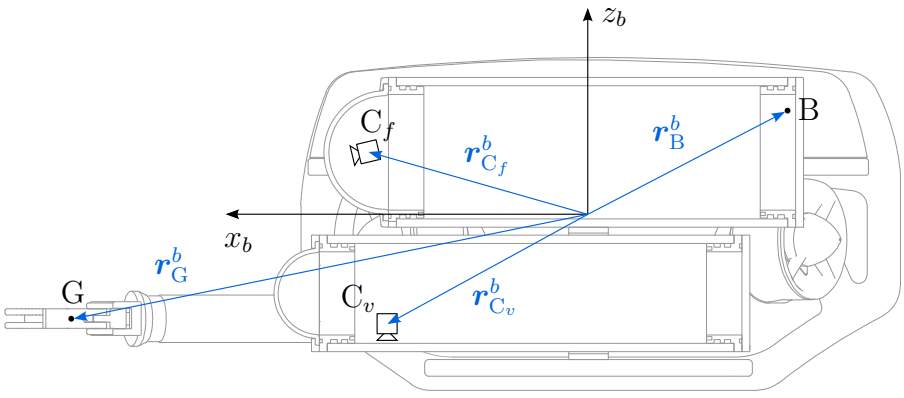


Abbildung 3.6: Positionen von Sensoren und Greifer. (G - Greifermittelpunkt, B - Barometer, C_f - Frontkamera, C_v - Vertikalkamera).

verbunden, siehe Abb 3.3, und über ein PWM-Signal von einem der Einplatinencomputer gesteuert. Der Greifarm erfüllt eine Funktion: Greifen durch Schließen der Greifbacken. Es können Objekte mit einem Durchmesser von bis zu 62 mm gegriffen werden. Die Greifbacken werden durch einen Linearantrieb bewegt und können eine Greifkraft von 97 N an ihrer Spitze und 124 N in ihrer Mitte ausüben [BlueRobotics21]. Der Greifarm wird im Verlauf dieser Arbeit auch als Endeffektor bezeichnet. Die Position des Mittelpunkts zwischen den Greifbacken ist in Tab. 3.2 gegeben und in Abb. 3.6 dargestellt.

Weitere Komponenten

Es sind vier dimmbare LED-Strahler verbaut, welche jeweils mit einer Helligkeit von bis zu 1500 Lumen leuchten. Sie ermöglichen bessere Sicht für die Kameras unter Wasser. Alle Komponenten im System werden durch eine $14,8$V Lithium-Ionen-Batterie mit Strom versorgt. Durch ihre Kapazität von 18Ah hält diese bei moderater Nutzung bis zu 4 Stunden [BlueRobotics21].

Das BlueROV2 ist über ein Kabel (engl. Tether) mit einer Basisstation verbunden. Dies dient dem direkten Datenaustausch. Für einen rein autonomen Betrieb kann das Kabel jedoch auch vom System getrennt werden.

3.3 Software-Architektur

Das Ziel dieser Arbeit ist, ein autonomes System zu entwickeln. Hierfür muss die Software-Architektur die Kapazitäten bereitstellen. Es wird das in [QuigleyEtAl09] vorgestellte Software-Framework Robot Operation System (ROS) genutzt. ROS stellt eine flexible Plattform zur Entwicklung von komplexer und robuster Software für Roboter dar. Das Konzept von ROS basiert auf Knoten (engl. Nodes), Nachrichten (engl. Messages), Themen (engl. Topics) und Diensten (engl. Services). Knoten führen bestimmte Berechnungen aus und kommunizieren miteinander über Nachrichten. Eine Nachricht wird von einem Knoten in einem Thema veröffentlicht (engl. Publishing) und kann von anderen Knoten abonniert werden (engl. Subscribing). Dieses Themen-basierte Publisher/Subscriber Modell bildet bereits eine flexible Kommunikationsmethode, welche jedoch für synchrone Transaktionen durch Dienste erweitert wird. Ein Dienst wird von einem Knoten per Nachricht angefragt und durch eine Antwort eines anderen Knotens ausgeführt.

Der Nachrichtenaustausch zwischen Programmen wird auf diese Weise sehr übersichtlich und modular gestaltet. Der Quellcode der Knoten kann in unterschiedlichen Programmiersprachen geschrieben sein, am verbreitetsten sind die Sprachen Python und C++.

ROS ist open-source und wird von einer großen Benutzergemeinschaft geprüft und weiterentwickelt. Es existieren eine Vielzahl von umfangreichen, frei verfügbaren Bibliotheken (engl. Packages) für wiederkehrende Aufgaben.

Auf dem Pixhawk 4 wird die open-source Firmware PX4 eingesetzt. Sie basiert ebenfalls auf dem Publisher/Subscriber Prinzip und zeigt damit eine starke Verwandtschaft zu ROS. Die PX4-Firmware wurde primär für den Einsatz in kleinen unbemannten Luftfahrzeugen (engl. Micro Aerial Vehicle (MAV)) entwickelt [MeierHoneggerPollefeys15]. Sie lässt sich jedoch aufgrund ihrer Modularität auf einer Vielzahl von Robotersystemen einsetzen. Die Software-Architektur von PX4 umfasst die Treiber für die eingebettete Hardware, das Echtzeitbetriebssystem NuttX, das Nachrichtensystem uORB, welches zur Kommunikation zwischen dem Betriebssystem und den Anwendungen dient, sowie die Anwendungen selbst.

Die Anwendungen müssen hierbei nicht zwangsweise auf dem Pixhawk 4 laufen, sondern können auf begleitenden Computern ausgeführt werden. Die Kommunikation des Pixhawk 4 mit externen Systemen läuft dabei über das MAVLink-Protokoll. Für ROS-basierte Systeme bietet die MAVROS Bibliothek eine Schnittstelle zwischen MAVLink-Nachrichten und ROS-Nachrichten. Somit kann problemlos zwischen Basisstation, Companion Computern und Pixhawk 4 kommuniziert werden.

Kapitel 4

Bewegungsplanung und Regelung

In den vorangegangenen Kapiteln wurden die Grundlagen der Bewegungsplanung und das in dieser Arbeit untersuchte System eingeführt. In diesem Kapitel wird aufbauend darauf eine Architektur für die Bewegungssteuerung bei Greifprozessen mit dem BlueROV2 entwickelt. Der Systemaufbau ist stark an dem in Abschnitt 2.3 eingeführten Aufbau von [Fossen11] orientiert. In Abb. 4.1 ist eine Übersicht des Systems gegeben.

Zu Beginn dieses Kapitels wird eine Problem- bzw. Aufgabenformulierung vorgenommen. Hierzu werden Vereinfachungen und Randbedingungen eingeführt. Anschließend werden die einzelnen Subsysteme der GNC-Architektur dargestellt. Zunächst wird ein vorhandenes Lokalisierungssystem erklärt und an die Problemstellung angepasst. Danach wird die Bewegungsplanung im Führungssystem behandelt. Hierbei wird unter anderem ein Pfadplanungsalgorithmus basierend auf Bézierkurven entwickelt. Abschließend wird die Regelung mit PID-Reglern (von engl. Proportional Integral Derivative, PID) erläutert.

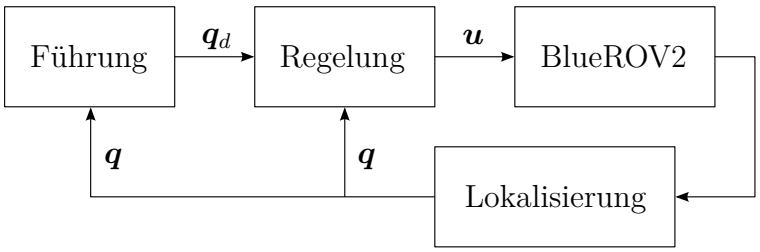


Abbildung 4.1: Vereinfachter Signalfluss der Systemarchitektur. Eine Lokalisierung bestimmt die Konfiguration $\boldsymbol{q}$. Mit ihr plant die Führung eine Bewegung und übergibt die Führungsgröße $\boldsymbol{q}_d$ an die Regelung, welche Stellgrößen $\boldsymbol{u}$ an das BlueROV2 sendet.

4.1 Problemformulierung

Das in Abschnitt 2.2 eingeführte Grundproblem der Bewegungsplanung lässt sich ohne weitere Anpassungen auf das vorliegende Problem des Greifens mit dem BlueROV2 übertragen. Der Endeffektor ist starr mit dem Unterwasserfahrzeug verbunden und somit kann der gesamte Roboter $\mathcal{R}$ als Starrkörper betrachtet werden. Seine Arbeitsumgebung $\mathcal{W}$ wird als euklidischer Raum $\mathbb{R}^3$ definiert. Grundsätzlich kann sich der Roboter im Raum bewegen und ein zu greifendes Objekt $\mathcal{O}$ kann beliebig im Raum platziert sein. Das Problem des Greifens in sechs Freiheitsgraden ist sehr komplex und würde den Umfang dieser Arbeit überschreiten. Daher werden nachfolgend einige Vereinfachungen getätigt.

Die Roll- und Nickbewegung des Roboters werden vor dem Hintergrund seiner Stabilitätseigenschaften vernachlässigt ($p = q = 0$). Die Freiheitsgrade des Roboters sind somit auf vier beschränkt: Translation (u, v) und Rotation (r) in der x-y-Ebene und Translation entlang der z-Achse (w). In der Arbeitsumgebung des Roboters gibt es keine Hindernisse oder andere Einschränkungen der Bewegungsmöglichkeit. Für die Ausgangsposition des Roboters wird festgelegt, dass das Objekt im Kamerasichtfeld der Frontkamera sein muss. Die Begründung hierfür gibt Abschnitt 4.2. Der Greifer ist wie in Abb. 4.2 dargestellt am Roboter angebracht. Er schließt sich um die z-Achse.

Das zu greifende Objekt ist statisch im Arbeitsraum platziert und hat eine bekannte Tiefe. Außerdem ist seine Größe beschränkt, sodass es zwischen die Greifbacken passt und somit durch den Endeffektor greifbar ist. Es wird ein objekt-

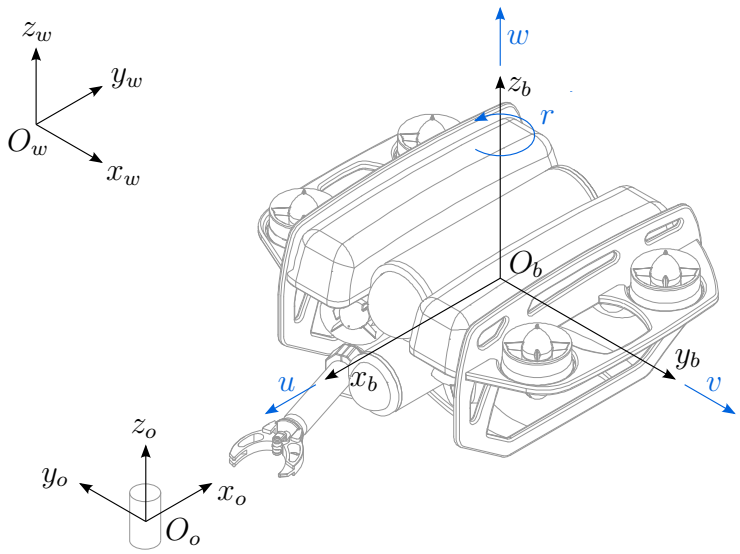


Abbildung 4.2: 3D-Darstellung der Kinematik des Problems.

festes KOS $\{o\} : \{O_o, x_o, y_o, z_o\}$ eingeführt, dessen Achsen ähnlich definiert sind, wie die des raumfesten Systems $\{w\}$. Der Koordinatenursprung O_o liegt im Zentrum des Objekts, welches zunächst vereinfacht als Zylinder dargestellt wird.

Die Aufgabe besteht darin, die Greifbacken des Endeffektors um das Objekt zu platzieren, damit ein Griff möglich ist. Die hierfür notwendigen Bedingungen werden im Laufe des Kapitels formuliert. Abbildung 4.2 zeigt schematisch die Zusammenhänge des Problems unter den eingeführten Vereinfachungen und Randbedingungen. Um den Greifer zu positionieren, muss sich der Roboter relativ zum Objekt richtig platzieren. Im Folgenden wird daher der Fahrzeugzustand bzw. die Konfiguration im Objekt-KOS $\{o\}$ ausgedrückt. Das Anfahren der Zielposition ist hierbei aus allen Richtungen der Ebene möglich. Um die Anfahrrichtung an das Objekt zu beschreiben, wird der Winkel α eingeführt. Er definiert unter welchem Winkel das Fahrzeug relativ zum Objekt in der Zielkonfiguration steht. Eine mögliche Zielkonfiguration in der Ebene ist in Abb. 4.3 dargestellt.

Für die Darstellung des Problems wird ein Konfigurationsraum $\mathcal{C} = \mathbb{R}^3 \times \mathcal{S}$ gewählt. Dieser ergibt sich aus den definierten Freiheitsgraden des Roboters. Eine Konfiguration $\boldsymbol{q} \in \mathcal{C}$ wird in $\{o\}$ ausgedrückt und definiert als

$$\boldsymbol{q}^o = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^\top . \tag{4.1}$$

Sie enthält somit die Position des Roboters und seine Ausrichtung in der x-y-Ebene. Die Zielkonfiguration wird in Abschnitt 4.3 hergeleitet. Mit der Konfiguration $\boldsymbol{q}^o$ und einer Zielkonfiguration $\boldsymbol{q}_d^o$ kann das Ziel der Pfadplanung und -regelung für das Anfahren des Objekts definiert werden als

$$\lim_{t \to \infty} [\boldsymbol{q}^o - \boldsymbol{q}_d^o] = 0. \tag{4.2}$$

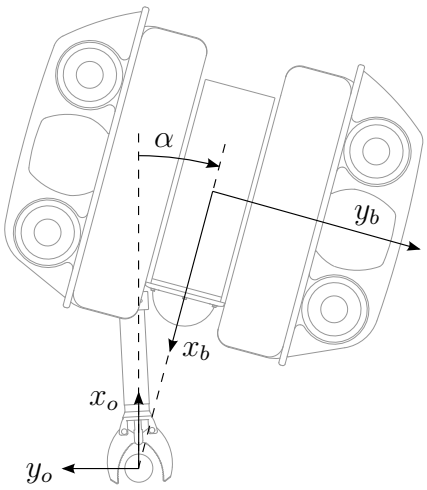


Abbildung 4.3: Geometrische Zusammenhänge einer Zielkonfiguration.

4.2 Lokalisierung

Damit die Bewegungsplanung und -regelung eines Unterwasserfahrzeugs möglich ist, muss seine Position und Ausrichtung bekannt sein. Dieser Abschnitt stellt die Funktionsweise einer visuellen Lokalisierung vor, welche genau diesen Zustand ermittelt. Das System wurde im Rahmen des Forschungsprojekts „Hydrobatic Micro Robots for Field Exploration in Hazardous Environments“ am Institut für Mechanik und Meerestechnik an der Technischen Universität Hamburg entwickelt und in [DueckerEtAl20] publiziert. Es wird in dieser Arbeit angepasst und untersucht. Die Anforderungen an das Lokalisierungssystem sind in Tab. 4.1 gegeben.

Tabelle 4.1: Anforderungen an die Lokalisierung.

A1	Stelle eine ausreichend genaue, robuste und zuverlässige Zustandsschätzung des Fahrzeugs relativ zum Objekt-KOS $\{o\}$ zur Verfügung.
A2	Stelle dem Führungssystem ununterbrochen die aktuelle Konfiguration des Fahrzeugs bereit.

Visuelle Landmarken werden statisch in der Umgebung des Objekts angebracht und mittels der Frontkamera vom BlueROV2 aufgezeichnet. Im Kamerabild $\mathcal{I}$ werden die Marker zunächst durch den AprilTag-Erkennungsalgorithmus erkannt, danach kann die Kamera- bzw. Fahrzeugposition $\hat{\boldsymbol{\eta}}$ relativ zu den Markern rekonstruiert werden. Es folgt eine Filterung dieser Daten durch ein erweitertes Kalman Filter, welches den Zustand $\boldsymbol{\eta}$ schätzt. Abschließend wird eine Transformation dieser Zustandsschätzung in eine, für die Planung und Regelung relevante, Konfigurationdarstellung $\boldsymbol{q}^o$ vorgenommen. Eine Übersicht über den Signalfluss der Lokalisierung bietet Abb. 4.4.

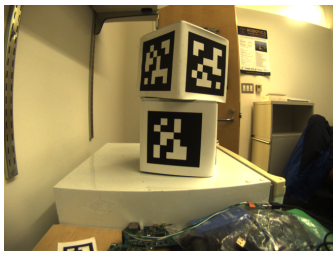
Abbildung 4.4: Vereinfachter Signalfluss der visuellen Lokalisierung.

4.2.1 AprilTag-Algorithmus

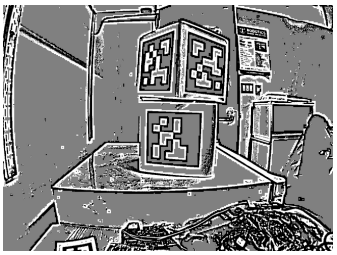
In diesem Abschnitt wird die Erkennung von künstlichen Landmarken mit dem AprilTag-Erkennungsalgorithmus erläutert. Der AprilTag-Algorithmus wurde erstmals in [Olson11] vorgestellt und in [WangOlson16] bzgl. der Effizienz und Robustheit angepasst und verbessert. Für einen Vergleich verschiedener Marker- und Erkennungssysteme und die Vorteile von AprilTag wird auf [Bauschmann18] verwiesen.

Ein AprilTag-Marker besteht aus einer schwarzen Umrandung und schwarzen und weißen Zellen, genannt Bits, im Inneren. Jede solcher unterschiedlichen AprilTag-Codierungen besitzt eine eindeutige ID, die durch den Erkennungsalgorithmus jedem erkannten Marker zugeordnet wird. Diese AprilTag-Marker werden nachfolgend als Tags bezeichnet. Mit AprilTag lassen sich verschiedene Familien an Tags generieren. Die Tag-Familien werden jeweils nach der Anzahl der verwendeten Bits und dem minimal gewährleisteten Hamming-Abstand, welcher Auskunft über die Unterscheidbarkeit der Tags gibt, benannt. In dieser Arbeit werden die Tags der Familie `36h11` verwendet. Sie sind in Abb. 4.5a gezeigt.

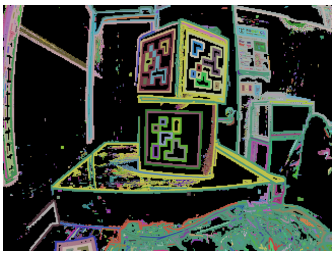
Der erste Schritt der Erkennung ist eine Umwandlung des Originalbildes, siehe Abb. 4.5a, in ein Schwarz-Weiß-Bild, in dem Pixel einen Wert für Schwarz oder Weiß annehmen. Dabei wird ein adaptives Schwellenwertverfahren verwendet, welches den Schwellenwert aus einem 4×4 Pixel-Feld und dessen Umgebung aus 3×3 Feldern bestimmt. Regionen mit unzureichendem Kontrast sind in Abb. 4.5b



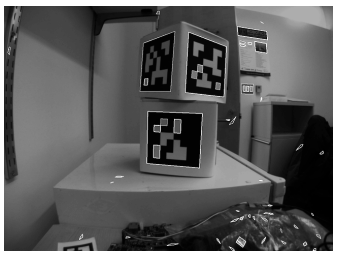
(a) Originalbild.



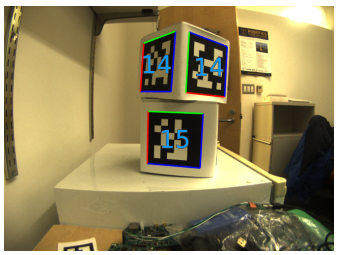
(b) Adaptives Schwellenwertverfahren.



(c) Segmentierung.



(d) Erkannte *Quads*.



(e) Erkannte Tags.

Abbildung 4.5: Schritte der AprilTag Erkennung aus [WangOlson16].

grau gefärbt und werden in der folgenden Berechnung ausgeschlossen um Rechenzeit zu sparen. Für eine detailliertere Erklärung wird auf [WangOlson16] verwiesen. Im nächsten Schritt werden an dieselbe schwarze oder weiße Region angrenzende Pixel gruppiert. Diese Segmentierung ist in Abb. 4.5c dargestellt. Anschließend werden Vierecke, so genannte *Quads*, an die ungeordneten Gruppen von Grenzpixeln zwischen schwarzen und weißen Regionen angepasst. Hierzu werden potentielle Eckpunkte gesucht und mögliche Verbindungen von diesen iteriert. Dieser Schritt gibt eine Reihe von Kandidaten-*Quads* aus, welche in Abb. 4.5d erkennbar sind. Die falschen *Quads* werden im letzten Schritt, der Tag-Codierung, herausgefiltert. Hierbei werden die detektierten Tags mit den hinterlegten Tags verglichen. Valide erkannte Tags sind mit ihrer ID in Abb. 4.5e markiert und werden in Form einer Transformation im Kamerasystem ausgegeben. Mit dieser Transformation kann die Lage des Fahrzeugs relativ zur Lage der Tags rekonstruiert werden.

4.2.2 Erweitertes Kalman Filter

Die Kamerabilder und somit auch die Zustandsmessungen des AprilTag-Algorithmus sind, wie bei jedem realen Sensor, mit Messungenauigkeiten behaftet. Die Lokalisierung beinhaltet daher eine gewisse Unsicherheit. Stochastische Zustandsschätzer bzw. Filter berücksichtigen bei der Beschreibung eines Systemzustands $\boldsymbol{x}$ diese Unsicherheiten. Der zu schätzende Zustand wird in dieser Arbeit als $\boldsymbol{x} = \boldsymbol{\eta}$, also als Position und Orientierung des Roboters definiert. Der Zustand wird in der nachfolgenden Erklärung mit $\boldsymbol{x}$ notiert, da dies die verbreitetste Notation ist.

Ein weit verbreitetes Filter ist das Kalman Filter. Es wurde erstmals in [Kalman60] vorgestellt. In diesem Abschnitt soll die Kalman Filter Theorie anhand des erweiterten Kalman Filters (EKF) erläutert werden. Die nachfolgende Erklärung basiert auf [Haykin01]. Kalman Filter basieren im Allgemeinen auf einem Zustandsraummodell mit zwei Gleichungen

$$\boldsymbol{x}_k = \boldsymbol{f}\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k\right) + \boldsymbol{w}_k, \tag{4.3}$$

$$\boldsymbol{z}_k = \boldsymbol{h}\left(\boldsymbol{x}_k\right) + \boldsymbol{v}_k. \tag{4.4}$$

Die erste Gleichung beschreibt den Übergang von Zustand $\boldsymbol{x}_{k-1}$ zu $\boldsymbol{x}_k$. Hierbei wird der aktuelle Zustand $\boldsymbol{x}_k$ durch eine Funktion $\boldsymbol{f}$ abhängig vom Vorgängerzustand $\boldsymbol{x}_{k-1}$ und einer Steuer- bzw. Störgröße $\boldsymbol{u}_k$ bestimmt. Zur Modellierung der Prozessunsicherheit wird ein mittelwertfreies und normalverteiltes Messrauschen $\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k)$ mit der Kovarianzmatrix $\boldsymbol{Q}_k$ addiert.

Die zweite Gleichung modelliert die Messung $\boldsymbol{z}_k$ des Zustands $\boldsymbol{x}_k$. Hierzu wird eine Mess- bzw. Beobachtungsfunktion $\boldsymbol{h}$ definiert, welche die Messung vom aktuellen

Zustand vorhersagt. Zur Beschreibung der Messunsicherheit wird auch hier ein mittelwertfreies und normalverteiltes Prozessrauschen $\boldsymbol{v}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_k)$ mit der Kovarianzmatrix $\boldsymbol{R}_k$ addiert.

Ein einfaches Kalman Filter ist für lineare Abbildungsfunktionen geeignet. Das in dieser Arbeit verwendetete System ist jedoch nichtlinear. Die zugrunde liegende Idee des EKF ist es, das nichtlineare Zustandsraummodell um den Punkt der aktuellen Schätzung zu linearisieren. Hierzu werden die Jacobi-Matrizen der nichtlinearen Funktionen $\boldsymbol{f}$ bzw. $\boldsymbol{h}$ berechnet.

Das Filterproblem besteht darin, aus Messwerten $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_k$ die Zustände $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ zu schätzen. Einer Zustandsschätzung $\boldsymbol{x}_k$ kann ebenfalls eine Kovarianz $\boldsymbol{P}_k$ zugeordnet werden. Der iterative und rekursive Prozess der Zustandsschätzung eines Kalman Filters basiert im wesentlichen auf zwei Schritten, der Prädiktion und der Korrektur. Für die mathematischen Hintergründe wird auf [Haykin01] verwiesen.

Zu Beginn wird das EKF mit einer a-priori Schätzung $\hat{\boldsymbol{x}}_0$ und der zugehörigen Kovarianzmatrix $\boldsymbol{P}_0$ initialisiert. Die Kovarianzmatrix $\boldsymbol{P}_0$ drückt hierbei aus, wie unsicher die initiale Schätzung $\hat{\boldsymbol{x}}_0$ ist. Anschließend folgt die regelmäßige Prädiktion. In diesem Schritt wird ein Zustand $\hat{\boldsymbol{x}}_k^-$ und seine Kovarianz $\boldsymbol{P}_k^-$ mit dem linearisierten Zustandsraummodell vorhergesagt. Die Kovarianz $\boldsymbol{P}_k^-$ wird mit jeder prädizierten Schätzung größer. Im Korrektur-Schritt wird die Schätzung $\hat{\boldsymbol{x}}_k^-$ mit jeder eingehenden Messung $\boldsymbol{z}_k$ korrigiert. Hierzu wird die Kalman-Matrix $\boldsymbol{K}_k$ abhängig von der neuesten Messung berechnet. Sie sagt aus, wie sehr die neue Messung die vorangegangene Zustandsschätzung $\hat{\boldsymbol{x}}_k^-$ beeinflusst. Eine hohe Unsicherheit im Messwert führt dazu, dass der Schätzung mehr vertraut wird als der neuen Messung. Durch die Kalman-Matrix gewichtet, wird anschließend ein neuer Zustand $\hat{\boldsymbol{x}}_k$ und seine neue Kovarianz $\boldsymbol{P}_k$ berechnet.

Für jeden weiteren Zeitschritt k wird der Systemzustand zeitlich prädiziert und durch neue Beobachtungen korrigiert. Es werden also auch Zustände ohne Messungen vorhergesagt. Diesen Schätzungen wird jedoch ohne neue Messungen zunehmend weniger vertraut.

Die grundlegende Voraussetzung dafür, dass die Lokalisierung und Zustandsschätzung funktioniert, ist, dass ausreichend viele Tags von der Kamera aufgezeichnet und durch den AprilTag-Algorithmus erkannt werden. Dafür müssen sich die Tags im Sichtfeld der Kamera befinden. Dies schränkt die Bewegungsmöglichkeiten des Roboters ein und muss bei der Bewegungsplanung berücksichtigt werden.

4.2.3 Darstellung im Konfigurationsraum

In diesem Abschnitt wird die Transformation des Zustands $\boldsymbol{\eta}$ in die Konfiguration $\boldsymbol{q}^o$ beschrieben. Aufgrund der Vereinfachung, dass Roll- und Nickbewegungen vernachlässigt werden, wurde der Konfigurationsraum in Abschnitt 4.1 bereits als $\mathcal{C} = \mathbb{R}^3 \times \mathcal{S}$ gewählt. Eine Konfiguration $\boldsymbol{q}^o = [x\ y\ z\ \psi]^\top$ wird relativ zum Objekt-KOS $\{o\}$ angegeben und beschreibt die Position in x-, y- und z-Richtung und den Gierwinkel ψ des Fahrzeugs. Da $\boldsymbol{q}^o$ sonst nicht injektiv wäre, wird der Winkel ψ auf das Intervall $[0; 2\pi)$ beschränkt.

Für den Fall, dass der Fahrzeugzustand $\boldsymbol{\eta}$ durch die Lokalisierung nicht in $\{o\}$ geschätzt wird, ist eine Transformation notwendig. Dies ist z. B. der Fall, wenn der Zustand im raumfesten System $\{w\}$ bestimmt wird. Da das Objekt statisch in $\{w\}$ platziert ist, ist die Umrechnung durch eine Koordinatentransformation mit der Position und Orientierung von $\{o\}$ in $\{w\}$ möglich.

Für die folgenden Überlegungen wird festgelegt, dass sowohl der AprilTag-Algorithmus als auch das EKF ihren Zustand $\hat{\boldsymbol{\eta}}$ bzw. $\boldsymbol{\eta}$ relativ zum Objekt bestimmen. Somit können die Einträge der Konfiguration $\boldsymbol{q}^o$ einfach aus dem Zustand $\boldsymbol{\eta}$ entnommen werden.

Es wird zusätzlich eine Unterkonfiguration $\boldsymbol{p}^o = [x\ y]^\top$ eingeführt. Sie beschreibt die Position des Roboters in der x-y-Ebene. Dies dient zum leichteren Verständnis des nächsten Abschnitts, welcher die Pfadplanung und -verfolgung in der Ebene behandelt. Die vollständige Konfiguration ergibt sich somit zu

$$\boldsymbol{q}^o = \begin{bmatrix} \boldsymbol{p}^{o\top} & z & \psi \end{bmatrix}^\top = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^\top . \tag{4.5}$$

Zusätzlich zur visuellen Lokalisierung ist die Fahrzeugtiefe z auch über den Druckmesser des Fahrzeugs bestimmbar. Mit der bekannten Tiefe des Objekts z_o^w kann z im Objekt-KOS $\{o\}$ geschrieben werden als

$$z = \frac{p - p_0}{\rho g} - z_o^w. \tag{4.6}$$

Hierbei ist die Objekttiefe z_o^w relativ zur Wasseroberfläche angegeben. Der Bruch beschreibt die berechnete Fahrzeugtiefe abhängig vom Atmosphärendruck p_0 über dem Wasser, dem gemessenen Druck p vom Barometer, der Wasserdichte ρ und der Erdbeschleunigung g. Im Vergleich zu dieser Tiefenbestimmung mittels Druckmessung bietet die visuelle Lokalisierung den Vorteil, dass die Objekttiefe bei ihr nicht bekannt sein muss. Beide Bestimmungsverfahren für die Tiefe z werden in Abschnitt 5.3.1 untersucht und verglichen.

4.3 Führung

Damit der Roboter das Objekt erreichen kann, muss zunächst die Bewegung zum Objekt geplant werden. Hierfür wird eine Führungsarchitektur nach den in Tabelle 4.2 gegebenen Anforderungen entworfen. Sie bildet die Grundlage für die anschließende Bewegungsregelung.

Tabelle 4.2: Anforderungen an die Führung.

A1	Gewährleiste eine unterbrechungsfreie Zustandsschätzung durch die Lokalisierung, indem die Tags am Objekt immer im Kamerasichtfeld sind.
A2	Generiere einen Pfad zwischen einer Anfangskonfiguration und einer Zielkonfiguration. Hierzu gehört das Anfahren einer vorgegebenen Position $\boldsymbol{p}_d^o$ unter einem vorgegebenen Winkel α in einer bestimmten Tiefe z_d^o.
A3	Stelle Führungsgrößen für die Regelung bereit.

Nach [Fossen11] wird typischerweise zwischen drei Szenarien der Bewegungssteuerung unterschieden:

- **Sollwertregelung** ist ein spezieller Fall, bei dem eine Führungsgröße wie z. B. eine gewünschte Position oder Ausrichtung als konstant gewählt wird.
- **Trajektorienverfolgung**, bei der eine Regelgröße $y(t)$ einer gewünschten Führungsgröße $y_d(t)$ folgt. Mögliche Trajektorien können unter räumlichen und temporalen Randbedingungen generiert werden.
- **Pfadverfolgung** beschreibt die Verfolgung eines Pfades unabhängig von der Zeit, siehe Abschnitt 2.2.1. Hierbei werden keine temporalen Randbedingungen entlang des Pfads festgelegt. Räumliche Bedingungen sind jedoch möglich.

In diesem Abschnitt wird ein Führungssystem entwickelt, welches unter Verwendung von Sollwertregelung und Pfadverfolgung die Führungsgrößen zur Bewegungsregelung des Fahrzeugs bereitstellt. Die zugrundeliegende Funktionsweise des Führungssystems besteht darin, das Fahrzeug auf die Tiefe des Objekts zu steuern und anschließend in der x-y-Ebene zu bewegen. Die Soll-Tiefe wird hierbei als konstante Führungsgröße z_d^o angegeben. Eine Gierwinkelregelung sorgt anschließend dafür, dass das BlueROV2 durchgehend zum Objekt ausgerichtet ist. Die Sollwertregelung für die Fahrzeugtiefe und den Gierwinkel werden in Abschnitt 4.3.1 beschrieben. Für translatorische Bewegung in der x-y-Ebene wird ein Algorithmus zur Pfadplanung in Abschnitt 4.3.2 und Pfadverfolgung in Abschnitt 4.3.3 entworfen. Abbildung 4.6 gibt einen Überblick über den Signalfluss in der Führungsarchitektur.

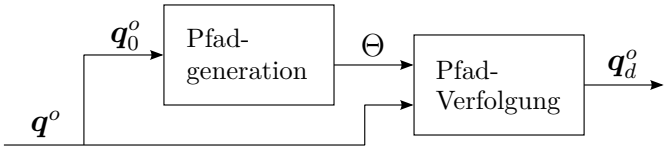


Abbildung 4.6: Signalfluss im Führungssystem. Mit einer Startkonfiguration $\boldsymbol{q}_0^o$ wird einmalig ein Pfad Θ generiert. Abhängig von der aktuellen Konfiguration $\boldsymbol{q}^o$ wird die Soll-Konfiguration $\boldsymbol{q}_d^o$ zur Verfolgung des Pfads und der anderen Führungsziele berechnet.

4.3.1 Gierwinkel- und Tiefenplanung

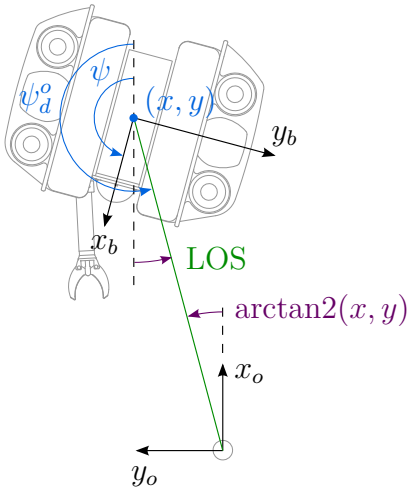
Die Anforderung **A1** besagt, dass das Objekt immer im Sichfeld der Kamera sein soll. Hierfür muss das Führungssystem sicherstellen, dass die Kamera und somit auch das Fahrzeug durchgehend in Richtung des Objekts ausgerichtet sind. Als Hilfsmittel wird zunächst die Sichtlinie LOS (von engl. Line of Sight) eingeführt. Sie ist definiert als die Strecke zwischen dem Fahrzeug und dem Objekt. Um die Anforderung zu erfüllen, wird das Ziel festgelegt, dass die Longitudinal-Achse x_b des Fahrzeugs gleich der Sichtlinie zum Objekt sein soll. Die Führungsgröße für den Gierwinkel ψ_d^o wird im Intervall $[0; 2\pi)$ definiert als

$$\psi_d^o = \pi + \arctan2(x, y), \tag{4.7}$$

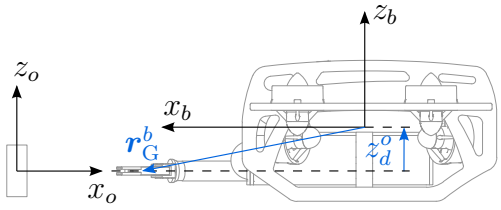
wobei x und y die x- und y-Koordinate der Fahrzeugposition $\boldsymbol{p}^o$ sind. Die arctan2-Funktion ist eine Erweiterung der inversen Winkelfunktion Arkustangens und berechnet den Polarwinkel zur Fahrzeugposition im Objekt-KOS $\{o\}$ im Intervall $[-\pi; \pi)$ eindeutig. Dieser Winkel beschreibt somit auch die Orientierung der Sichtlinie zwischen dem Objekt und dem Fahrzeug. Für den Soll-Gierwinkel im gegeben Intervall muss der Polarwinkel um π verschoben werden. Die Zusammenhänge hierfür sind in Abb. 4.7a erkennbar.

Zum Greifen muss sich der Greifarm auf der Tiefe des Objekts befinden. Da die Greifbacken bezüglich der z^b-Koordinate unter der Fahrzeugposition liegen, muss sich das fahrzeugfeste System zum Greifen um genau diese Distanz oberhalb des Objekts befinden. Abbildung 4.7b veranschaulicht diese Überlegung. Die Soll-Tiefe z_d^o für das Fahrzeug wird im Objekt-KOS $\{o\}$ mit der Greiferposition $r_{G,z}^b$ aus Tabelle 3.2 definiert als

$$z_d^o = -r_{G,z}^b. \tag{4.8}$$



(a) Soll-Gierwinkel in x-y-Ebene.



(b) Soll-Tiefe in z-Richtung.

Abbildung 4.7: Geometrische Zusammenhänge für die Führungsgrößenbestimmung des Gierwinkels ψ_d^o und der Tiefe z_d^o.

4.3.2 Pfadplanung in der xy-Ebene

Dieser Abschnitt präsentiert einen auf quadratischen Bézierkurven basierenden Pfadplanungsalgorithmus. Die nützlichen Eigenschaften von Bézierkurven für das vorliegende Pfadplanungsproblem werden im Folgenden dargelegt. Die mathematische Formulierung der Kurven in diesem Abschnitt orientiert sich an [Farin02]. Eine Anwendung der Pfadplanung mit Bézierkurven bei Landfahrzeugen ist in [ChoiCurryElkaim08] beschrieben. In [LiYang20] werden Bézierkurven genutzt, um Trajektorien für autonome Unterwasserfahrzeuge bezüglich ihrer Glattheit zu optimieren.

Eine Bézierkurve ist eine Abbildung von einem Parameter $s \in [0; 1]$ auf eine konvexe Kombination von Punkten $\boldsymbol{P}_0, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$ in einem Vektorraum. Eine Bézierkurve vom Grad n ist definiert als

$$\boldsymbol{P}(s) = \sum_{i=0}^{n} B_i^n(s) \boldsymbol{P}_i \quad \text{mit } 0 \leq s \leq 1. \tag{4.9}$$

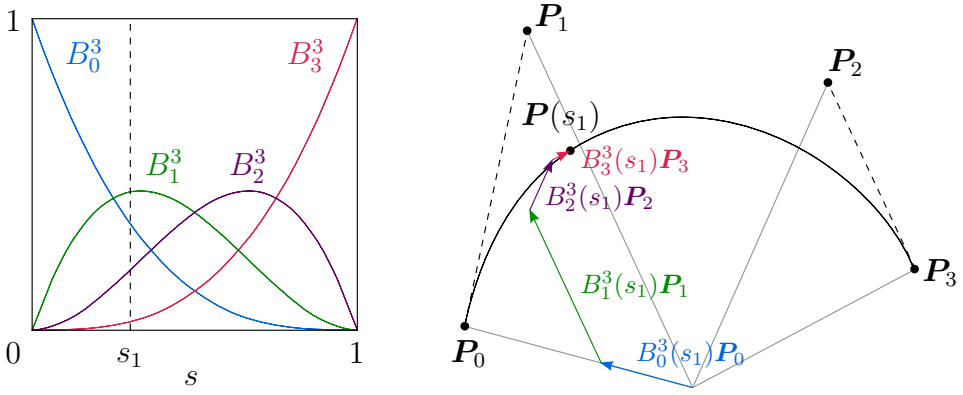
Hierbei sind $\boldsymbol{P}_i$ die sogenannten Kontrollpunkte. Die Gesamtheit der Kontrollpunkte wird auch Kontrollpolygon genannt. $\boldsymbol{P}(s = 0) = \boldsymbol{P}_0$ ist der Anfangspunkt und $\boldsymbol{P}(s = 1) = \boldsymbol{P}_n$ der Endpunkt der Kurve. $\boldsymbol{P}_1 - \boldsymbol{P}_0$ ist die Richtung der Tangente im Anfangspunkt. $\boldsymbol{P}_{n-1} - \boldsymbol{P}_n$ ist die Richtung der Tangente im Endpunkt. Der Faktor $B_i^n(s)$ stammt aus der Bernsteinbasis $\{B_0^n(s), B_1^n(s), \ldots, B_n^n(s)\}$, welche definiert ist als

$$B_i^n(s) := \binom{n}{i} s^i (1 - s)^{n-i} \quad \text{mit } 0 \leq i \leq n. \tag{4.10}$$

Die Bernsteinbasis erlaubt es, eine Kurve unter den eben genannten Randbedingungen zu berechnen. Hierbei lässt sich Gl. (4.9) als eine gewichtete Kombination aus den Kontrollpunkten interpretieren. Jeder Kontrollpunkt $\boldsymbol{P}_i$ wird mit einem Bernsteinpolynom B_i^n gewichtet. Abb. 4.8 zeigt eine grafische Interpretation der Gleichung.

Bézierkurven haben mehrere nützliche Eigenschaften für das vorliegende Pfadplanungsproblem. Einerseits verläuft die Kurve in jedem Fall durch einen definierten Startpunkt $\boldsymbol{P}_0$ und einen definierten Endpunkt $\boldsymbol{P}_n$. Diese Punkte werden mit der Start- bzw. Zielkonfiguration gleichgesetzt. Die Kontrollpunkte $\boldsymbol{P}_1$ und $\boldsymbol{P}_{n-1}$ definieren dann die Tangenten der Kurve im Anfangs- bzw. Endpunkt. Sie legen also fest, wie aus der Startkonfiguration begonnen und in der Zielkonfiguration geendet wird. Diese Eigenschaft der Kurven ist besonders nützlich für nicht-holonome Systeme, da diese meist aufgrund ihrer Aktuation nur in einer möglichen Richtung anfahren können. Die Tangente im Startpunkt kann hierbei die notwendige Anfahrrichtung modellieren.

Wenn alle Kontrollpunkte auf einer Linie liegen, wird die Bézierkurve zu einer Geraden. Dies ist ein deutlicher Vorteil gegenüber einfacher Polynominterpolation zwischen den Punkten [Farin02]. Das BlueROV2 ist vollaktuiert und kann sich somit aus dem Stand in alle Richtungen der Ebene bewegen. Die Anfahrrichtung aus der Startposition ist durch keine Randbedingungen eingeschränkt und wird somit auch durch keinen Kontrollpunkt berücksichtigt. In dieser Arbeit wird daher eine Pfadgeneration mit einer quadratischen Bézierkurve (Grad $n = 2$) gewählt. $\boldsymbol{P}_0$ ist dabei die Startposition des Fahrzeugs und $\boldsymbol{P}_2$ seine Zielposition. $\boldsymbol{P}_1$ wird primär dafür verwendet die Tangente zum Punkt $\boldsymbol{P}_2$ zu bilden und somit den Anfahrwinkel α zum Objekt zu definieren. Das Kontrollpolygon zu dem Pfadplanungsproblem ist in Abb. 4.9 dargestellt.



(a) Bernsteinpolynome. (b) Kontrollpolygon und Bézierkurve.

Abbildung 4.8: Darstellung der Bernsteinpolynome im kubischen Fall (links) und die aus ihnen resultierende Bézierkurve (rechts).

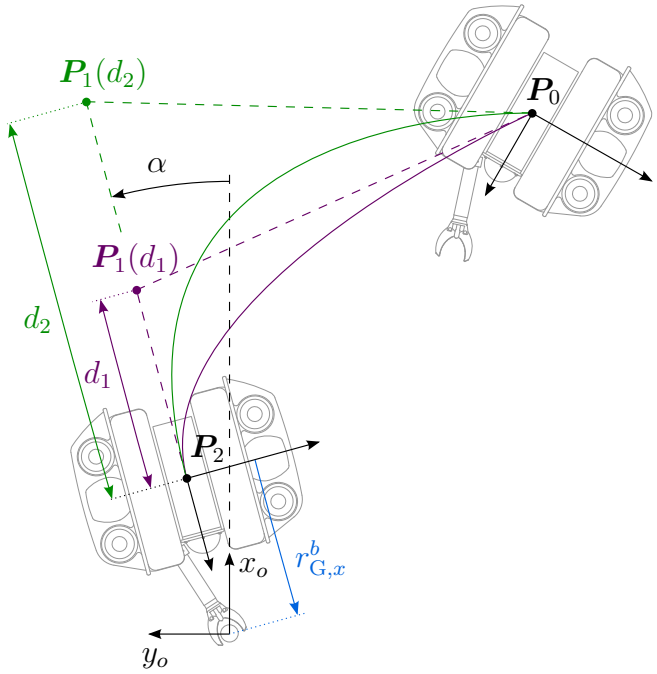


Abbildung 4.9: Exemplarisches Kontrollpolygon und resultierende Bézierkurven zwischen Anfangs- und Zielposition für zwei verschiedene Werte von d ($d_1 < d_2$).

Das Ziel ist es, die Greifbacken um das Objekt zu platzieren. Für die Definition des Kontrollpolygons wird neben dem Objekt-Anfahrwinkel α als neue Kenngröße die Distanz d eingeführt. Der Winkel definiert, wie die Tangente der Bézierkurve in den Zielpunkt läuft. Die Distanz definiert den Abstand des Kontrollpunkts $\boldsymbol{P}_1$ zum Zielpunkt und somit auch die Form der Kurve. Die Kontrollpunkte werden in $\{o\}$ definiert als

$$\boldsymbol{P}_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \quad \boldsymbol{P}_1(\alpha, d) = \begin{bmatrix} \cos(\alpha)(r^b_{\mathrm{G},x} + d) \\ \sin(\alpha)(r^b_{\mathrm{G},x} + d) \end{bmatrix}, \quad \boldsymbol{P}_2(\alpha) = \begin{bmatrix} \cos(\alpha) r^b_{\mathrm{G},x} \\ \sin(\alpha) r^b_{\mathrm{G},x} \end{bmatrix}. \tag{4.11}$$

Die Berechnung berücksichtigt, dass die Fahrzeugposition $\boldsymbol{p}^o$, die den Pfad definiert und verfolgen soll, nicht identisch der Greifarmposition ist. Die Objektposition ist im Koordinatenursprung O_o gegeben. Damit der Greifarm richtig platziert wird, muss die Zielposition bzw. der Kontrollpunkt $\boldsymbol{P}_2$ also in einem Abstand zu O_o liegen. Dieser Abstand entspricht genau der Endeffektor-Koordinate $r^b_{\mathrm{G},x}$ aus Tab. 3.2.

Die Kontrollpunkte werden abhängig von einer Startposition $\boldsymbol{p}^o_0$ und den Parametern α und d einmalig berechnet. Die Kurve wird anschließend mit Hilfe von Gl. (4.9) an m Punkten ausgewertet und somit in Wegpunkte für die Pfadverfolgung diskretisiert. Der Pfadparameter s ist somit nicht kontinuierlich, sondern

diskret. Ein Wegpunkt $\boldsymbol{t}_i^o$ wird in $\{o\}$ definiert. Die Menge aller Wegpunkte wird von der Pfadplanung als Pfad $\Theta = \{\boldsymbol{t}_1^o, \boldsymbol{t}_2^o, \ldots, \boldsymbol{t}_m^o\}$ ausgegeben.

4.3.3 Pfadverfolgung in der xy-Ebene

Als Methode für die Pfadverfolgung wird sich aufgrund der Robustheit und einfachen Implementierbarkeit an reiner Verfolgung (von engl. *Pure Pursuit*) orientiert. *Pure Pursuit* wird sowohl bei Landfahrzeugen [PadenEtAl16] als auch bei Wasserfahrzeugen [Fossen11] erfolgreich eingesetzt. Die Erklärungen in diesem Abschnitt basieren auf [Coulter92].

Der *Pure Pursuit* Algorithmus berechnet abhängig von der aktuellen Position des Fahrzeugs und einer vorausschauenden Distanz l (engl. *Lookahead Distance*) einen Zielpunkt. Zur Regelung wird dieser Punkt anschließend in das Fahrzeug-KOS transformiert und die Bewegungsrichtung des Fahrzeugs in Richtung des Zielpunktes ausgerichtet [Fossen11].

Abbildung 4.10 stellt die geometrischen Zusammenhänge der verwendeten Pfadverfolgung dar. Mittels der Laufvariable i wird über die Wegpunkte des Pfads Θ iteriert. Für jeden Wegpunkt $\boldsymbol{t}_i^o$ beginnend bei $i = 1$ wird dessen Distanz zum Fahrzeug bestimmt. Wenn die Distanz $\|\boldsymbol{t}_i^o - \boldsymbol{p}^o\|$, also die Distanz vom Wegpunkt zum Fahrzeug $\|\boldsymbol{t}_i^b\|$, kleiner als die *Lookahead Distance* l ist, wird i um 1 erhöht. Falls $\|\boldsymbol{t}_i^o - \boldsymbol{p}^o\| > l$ ist, wird dieser Punkt als nächster Zielwegpunkt festgelegt. Da sich das Fahrzeug in Richtung des Wegpunkts bewegt, wird es dazu kommen, dass die Distanz zu ihm kleiner wird und der Algorithmus einen neuen Zielweg-

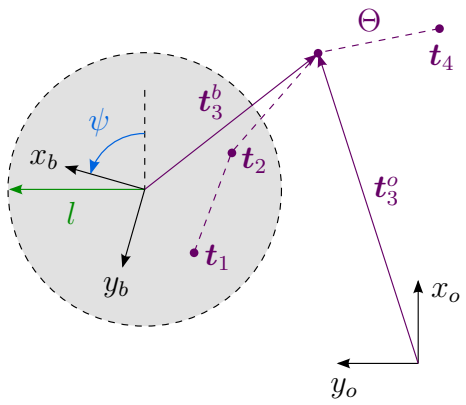


Abbildung 4.10: Geometrische Zusammenhänge der Pfadverfolgung des Pfads Θ. In diesem Fall würde $\boldsymbol{t}_3$ als nächster Zielwegpunkt identifiziert werden, da er nach der Reihenfolge der nächste außerhalb der Lookahead Distance ist.

punkt auf dem Pfad festlegt. Diese Iteration wird solange durchgeführt bis die Laufvariable $i = m$ ist, also der letzte Wegpunkt erreicht ist. Ein identifizierter Zielwegpunkt wird als Soll-Position $\boldsymbol{p}_d^o$ an die Regelung übergeben.

4.4 Regelung

Moderne Regelungssysteme basieren auf einer Vielzahl von Algorithmen wie z. B. klassischer PID-Regelung, Linear-Quadratische Regelung, neurale Netzwerken oder nicht-linearer Regelungstechnik [Fossen11]. PID-Regelung ist hierbei bei weitem der verbreitetste Regelungsalgorithmus [ÅströmHägglund95][Fossen11]. Aufgrund der Implementierbarkeit und bewiesenen Leistungfähigkeit bei Wasserfahrzeugen wird sich in dieser Arbeit für eine PID-Regelung entschieden.

Tabelle 4.3: Anforderungen an die Regelung.

A1	Gewährleiste möglichst genaue und reaktionsschnelle Verfolgung der Referenzgröße $\boldsymbol{q}_d^o$.
A2	Biete einen einfach einstellbaren und verständlichen Regleraufbau.
A3	Steuere die acht BlueROV2 Propeller, um die Führungsgrößen zu erreichen.

Dieser Abschnitt stellt zu Beginn das verwendete PID-Regelungskonzept vor. Anschließend wird die Zuordnung der Stellgrößen zu den Bewegungsrichtungen des BlueROV2 erläutert. Abbildung 4.11 gibt einen Überblick über das entwickelte Regelungskonzept.

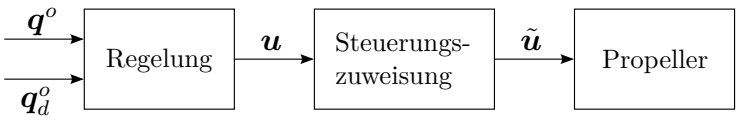


Abbildung 4.11: Signalfluss im Regelungssystem. Die Regelung berechnet anhand der aktuellen Konfiguration $\boldsymbol{q}^o$ und der Soll-Konfiguration $\boldsymbol{q}_d^o$ eine Stellgröße $\boldsymbol{u}$. Die Steuerungszuweisung kommandiert jeden Propeller individuell über die Stellgröße $\tilde{\boldsymbol{u}}$.

4.4.1 PID-Regelung

Die meisten ROV-Systeme nutzen nach [Fossen94] dezentrale PID-Regler mit einer Eingangsgröße und einer Ausgangsgröße (engl. Single-Input Single-Output, SISO). Da das BlueROV2 vollaktuiert ist, wird sich in dieser Arbeit auch für eine SISO-Architektur entschieden, welche die vier betrachteten Freiheitsgrade des Systems entkoppelt regelt. Die Erklärungen in diesem Abschnitt stützen sich auf [ÅströmHägglund95].

Berechnung der Regelabweichung

Das Ziel des Reglers ist es, die Regelgröße $\boldsymbol{q}^o$ mit Hilfe einer negativen Rückführung auf das Niveau der Führungsgröße $\boldsymbol{q}_d^o$ einzustellen. Hierzu wird eine Regelabweichung $\boldsymbol{e}(t)$ definiert als

$$\boldsymbol{e}(t) = \boldsymbol{q}_d^o - \boldsymbol{q}^o. \tag{4.12}$$

Da die z-Achsen in allen Systemen aufgrund der Vereinfachungen immer parallel verlaufen, sind Differenzen zwischen Punkten in z-Richtung in allen Koordinatensystemen gleich. Die Regelabweichung der Tauchtiefe e_z bzw. des Gierwinkels e_ψ wird über die Differenz zwischen Führungs- und Regelgröße berechnet zu

$$e_z = z_d^o - z \quad \text{und} \quad e_\psi = \psi_d^o - \psi. \tag{4.13}$$

Damit die Bewegungsrichtungen x_b und y_b des Fahrzeugs entkoppelt behandelt werden können, muss die Soll-Position $\boldsymbol{p}_d^o$ aus dem Objekt-KOS in das Fahrzeug-KOS transformiert werden. Hierzu wird eine Koordinatentransformation mit der Drehmatrix $\boldsymbol{S}_{bo}$ durchgeführt.

$$\boldsymbol{e}_p^b = \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \boldsymbol{S}_{bo}(\boldsymbol{p}_d^o - \boldsymbol{p}^o) = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} p_{d,x}^o - x \\ p_{d,y}^o - y \end{bmatrix}. \tag{4.14}$$

Die Matrix $\boldsymbol{S}_{bo}$ ist hierbei die Inverse bzw. Transponierte der Drehmatrix $\boldsymbol{S}_{ob}$, welche die Verdrehung von $\{b\}$ relativ zu $\{o\}$ beschreibt und abhängig vom Gierwinkel ψ ist. Die Regelabweichung $\boldsymbol{p}_d^o - \boldsymbol{p}^o$ wird mittels einer Multiplikation mit $\boldsymbol{S}_{bo}$ in das körperfeste System transformiert und ergibt den Positionsfehler $\boldsymbol{e}_p^b$ im Fahrzeug-KOS $\{b\}$.

Für die Regelabweichung $\boldsymbol{e}$ wird sich an der Konfigurationsdarstellung orientiert. Sie ergibt sich zu

$$\boldsymbol{e} = \begin{bmatrix} e_x & e_y & e_z & e_\psi \end{bmatrix}^\top. \tag{4.15}$$

Herleitung des Regelgesetzes

Um die Regelabweichung zu minimieren, wird eine Stellgröße $\boldsymbol{u}(t)$, welche auf das System wirkt, in der Standardform eines PID-Reglers definiert als

$$\boldsymbol{u}(t) = \boldsymbol{K}_p \boldsymbol{e}(t) + \boldsymbol{K}_i \int_0^t \boldsymbol{e}(\tau) d\tau + \boldsymbol{K}_d \frac{d\boldsymbol{e}(t)}{dt}. \tag{4.16}$$

Die Stellgröße wird als Summe aus drei gewichteten Anteilen berechnet. Das erste Glied ist eine proportionale Verstärkung der Regelabweichung mit der Diagonalmatrix $\boldsymbol{K}_p$ (P-Anteil). Das zweite Glied wirkt durch zeitliche Integration der Regelabweichung auf die Stellgröße und wird gewichtet durch $\boldsymbol{K}_i$ (I-Anteil). Das dritte Glied ist ein Differenzierer, welcher durch $\boldsymbol{K}_d$ gewichtet auf die Änderungsgeschwindigkeit der Regelabweichung reagiert (D-Anteil).

Abbildung 4.12 zeigt schematisch eine leicht modifizierte Darstellung des PID-Reglers. Zusätzlich zum P-, I- und D-Glied werden zwei Sättigungs-Operationen eingeführt. Sie wirken auf den I-Anteil und die summierte Stellgröße. Die I-Anteil-Begrenzung dient dazu, den Integrator-Windup-Effekt zu verhindern. Der Effekt kann auftreten, wenn die kommandierte Stellgröße größer als die umsetzbare Stellgröße der Regelstrecke ist. Weitere Hintergründe und Darstellungen von unerwünschtem Regelkreisverhalten durch den Integrator-Windup-Effekt können [ÅströmHägglund95] entnommen werden. Die zweite Sättigungsoperation dient dazu, die Wirkung des Reglers auf die Regelstrecke zu begrenzen. Eine Sättigungsoperation ist definiert als

$$\mathrm{sat}(u_k) = \begin{cases} \mathrm{sgn}(u_k) u_{k,max} & \text{für } |u_k| \geq u_{k,max} \\ u_k & \text{für } |u_k| < u_{k,max} \end{cases} \tag{4.17}$$

und begrenzt somit jeden Eintrag von $\boldsymbol{u}_i$ bzw. $\boldsymbol{u}$ mit $k \in \{x, y, z, \psi\}$ auf das Intervall $[-u_{k,i,max}; u_{k,i,max}]$ bzw. $[-u_{k,max}; u_{k,max}]$.

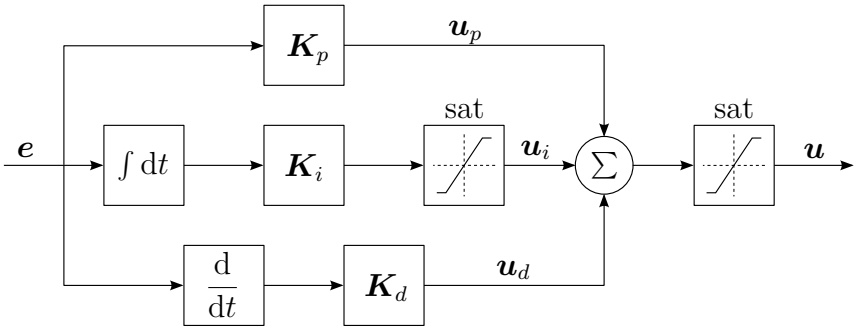


Abbildung 4.12: Schematische Darstellung des Regelgesetzes.

Wie zu Beginn des Abschnitts erwähnt, werden die Regler in einer SISO-Regelungsarchitektur verwendet, da die diskutierten Freiheitsgrade individuell aktuierbar sind. Für jede Regelabweichung (e_x, e_y, e_z und e_ψ) bzw. jede Bewegungsrichtung wird ein, von den anderen unabhängiger, PID-Regler genutzt. Dies impliziert, dass die Matrizen $\boldsymbol{K}_p$, $\boldsymbol{K}_i$ und $\boldsymbol{K}_d$ positiv und diagonal sind.

Das Ergebnis der Regelung ist die Stellgröße

$$\boldsymbol{u} = \begin{bmatrix} u_x & u_y & u_z & u_\psi \end{bmatrix}^\top . \tag{4.18}$$

4.4.2 Steuerungszuweisung

Damit die Stellgröße $\boldsymbol{u}$ eine Bewegung in der gewollten Richtung steuert, werden die Propeller verwendet. Sie wirken durch externe Kräfte $\boldsymbol{\tau}$ auf das Unterwasserfahrzeug und verursachen ein Bewegung. Da die einzelnen Propeller im BlueROV2 durch ihre Anordnung jedoch nicht unabhängig die Freiheitsgrade des Fahrzeugs aktuieren, muss eine Zuweisung der Stellgrößen zu den einzelnen Propellern geschehen. Hierzu wird ein sogenannter Mixer-Algorithmus definiert als

$$\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{B}}\boldsymbol{u} \quad \text{bzw.} \quad \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \\ \tilde{u}_5 \\ \tilde{u}_6 \\ \tilde{u}_7 \\ \tilde{u}_8 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & -1 & 0 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \\ u_\psi \end{bmatrix} . \tag{4.19}$$

Er bildet die Stellgröße $\boldsymbol{u}$, welche die angestrebten Bewegungsrichtungen enthält, mit einer Matrix $\tilde{\boldsymbol{B}}$ auf eine neue Steuervariable $\tilde{\boldsymbol{u}}$ für die Motoren ab. Die Matrix wird so gewählt, dass die gewollte Bewegungsrichtung durch eine vektorielle Kombination der einzelnen Schubkraftvektoren der Propeller erreicht wird. Die Mixer-Matrix wird anhand von Abb. 3.4 aufgestellt.

Nach [Fossen94] kann die externe Krafteinwirkung durch die Antriebspropeller in vielen praktischen Anwendungen durch das Modell

$$\boldsymbol{\tau} = \boldsymbol{B}\tilde{\boldsymbol{u}} \tag{4.20}$$

approximiert werden. Hierbei ist $\boldsymbol{B}$ eine Matrix mit passenden Dimensionen und $\tilde{\boldsymbol{u}}$ die eben eingeführte Steuervariable der Propeller. Die Matrix bildet hierbei die Stellgrößen der Propeller auf die externen Schubkräfte im System ab und modelliert somit die Zusammenhänge zwischen Ein- und Ausgangsgrößen der Propeller. Sie sieht ähnlich aus wie die Inverse von $\tilde{\boldsymbol{B}}$. Die Einträge sind hierbei abhängig von den Ausrichtungen des Propeller und ihren Schubkraftkennlinien.

Kapitel 5

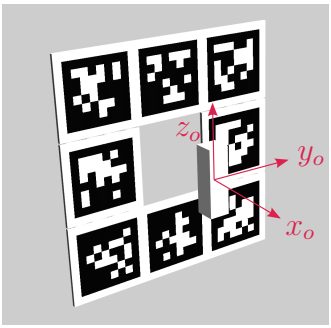
Implementierung und Validierung

In diesem Kapitel wird die im Rahmen dieser Arbeit entwickelte Architektur auf der BlueROV2-Plattform implementiert und untersucht. Zunächst gibt dieses Kapitel einen Überblick über die Validierungsumgebung, welche sowohl aus einer Simulationsumgebung als auch aus einem Experimentaufbau besteht. Anschließend wird die Implementierung der entwickelten Architektur für das Greifen mit dem BlueROV2 beschrieben. Es folgt eine getrennte Analyse der visuellen Lokalisierung auf Genauigkeit und der Pfadplanungs, -verfolgungs und Regelalgorithmen auf Funktion. Eine getrennte Untersuchung der Subsysteme verhindert Fehlzuordnung eventueller Fehler im System. Das System wird abschließend ganzheitlich auf seine Leistungsfähigkeit und Robustheit im Hardware-Experiment untersucht.

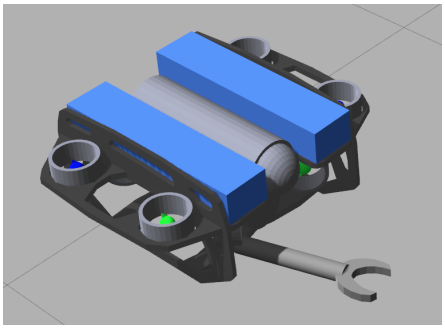
5.1 Validierungsumgebung

Zur Untersuchung und Validierung der in Kap. 4 entwickelten Architektur werden eine Software-in-the-Loop Simulationsumgebung und ein Experimentaufbau in einem Stillwasserbecken genutzt.

In beiden Umgebungen wird das zu greifende Objekt gleich aufgebaut. Es hat die Form eines Quaders mit den Abmaßen $3 \times 3\,\text{cm}$ in x- und y-Richtung und ist somit ungefähr halb so groß wie der maximal zulässige Objektdurchmesser von 6,2 cm. Es werden AprilTag Marker der Familie `36h11` in 10 cm Abstand hinter dem Objekt platziert. Diese Distanz stellt sicher, dass immer ausreichend viele Tags im Sichtfeld der Fahrzeugkamera sind, auch wenn diese nah am Objekt positioniert ist. Das verwendete Objekt-KOS $\{o\}$ ist in Abb. 5.1a dargestellt.



(a) Objektaufbau in Gazebo.



(b) BlueROV2 mit Greifer in Gazebo.

Abbildung 5.1: Bestandteile in der Simulationsumgebung Gazebo.

Simulationsumgebung

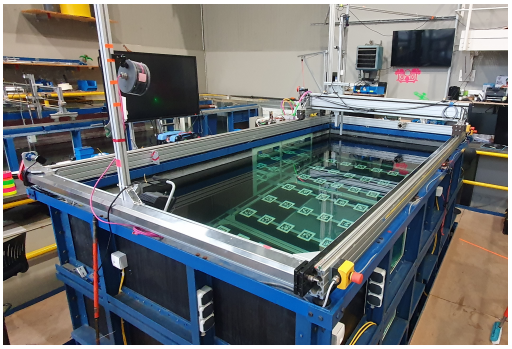
Als Simulationsumgebung wird Gazebo verwendet. Gazebo ist ein open-source 3D-Roboter-Simulator, welcher neben der Simulation von physikalischen Effekten auch eine realistische Darstellung der Umgebung bietet [KoenigHoward04]. Die Umgebung beinhaltet Beleuchtung, Schatten und Texturen und kann durch simulierte Sensoren wahrgenommen werden. Letzteres ist besonders hilfreich für die Simulation der visuellen Lokalisierung mit AprilTags.

In [Hastedt19] wurde ein Gazebo-Plugin entwickelt, welches Effekte wie hydrodynamische Zusatzmasse und hydrodynamische Dämpfung nach [Fossen11] auf Körper in Gazebo simuliert. Im Zusammenhang mit dieser Arbeit wird ein bereits existierendes BlueROV2 Modell, welches das Plugin nutzt, verwendet. Die hydrodynamischen Modellparameter werden mit Hilfe von Systemidentifikationen aus [Mogk20] und [Sandøy16] in dieser Arbeit angepasst, um realistische Simulationsergebnisse zu erhalten. Das BlueROV2-Modell wird außerdem um eine vereinfachte Nachbildung des *Newton Subsea Gripper* erweitert. Das Greifarm-Modell ist statisch und dient dazu, eine Einschränkung des Kamerasichtfelds und Kontaktkräfte mit der Umgebung zu simulieren. Abbildung 5.1b zeigt das simulierte BlueROV2-Modell mit dem Greifarm.

Experimentumgebung

Als Umgebung für Experimente in der Realität wird ein Stillwasserbecken im Institut für Mechanik und Meerestechnik der Technischen Universität Hamburg genutzt. Die Abmaße des Tanks sind ungefähr 4 m in der Länge, 2 m in der Breite und 1,4 m in der Tiefe. Abbildung 5.2b zeigt das Stillwasserbecken. Das Objekt ist analog zur Simulationsumgebung aufgebaut und ist in Abb. 5.2a dargestellt.

(a) Objektaufbau im Experiment.



(b) Stillwasserbecken.

Abbildung 5.2: Bestandteile der Hardware-Experiment-Umgebung.

5.2 Implementierung

Wie bereits in Kap. 3 beschrieben, basiert die Software des Systems überwiegend auf ROS. Alle Berechnungen, welche im Zusammenhang mit dieser Arbeit entwickelt wurden, werden in ROS-Knoten mit der Programmiersprache Python integriert. Nachfolgend wird erklärt, wie die einzelnen Subsysteme der in Kap. 4 entwickelten Architektur implementiert wurden.

Lokalisierung

Die in [DueckerEtAl20] vorgestellte und in Abschnitt 4.2 erklärte Lokalisierung ist bereits in ROS implementiert und wird in dieser Arbeit für Greifprozesse angepasst. Nachfolgend wird der Signalfluss in ROS beschrieben.

Die Bilder der Frontkamera werden von dem Knoten `/camera_node` veröffentlicht und im Knoten `/image_rectifier` entzerrt. Für die Einbindung der AprilTag-Erkennung wird der in [Malyuta17] entwickelte ROS-Wrapper `apriltag_ros` verwendet. Es wird ein sogenanntes Tag-Bundle eingeführt. In ihm sind die Positionen und Orientierungen der einzelnen Tags relativ zu einem KOS angegeben. Das Tag-Bundle wird so definiert, dass das Referenzsystem der in Abb. 5.1a eingeführten Konvention für das Objekt-KOS $\{o\}$ entspricht. Im Knoten `/apriltag_node` werden die Tags im entzerrten Kamerabild erkannt (Abb. 5.3) und das Objekt-KOS $\{o\}$ relativ zur Kamera als Transformation veröffentlicht. Diese Transformation wird umgekehrt, um die Position der Kamera relativ zum Objekt zu erhalten. Der Knoten `/ekf_node` nutzt statische Transformationen (aus Tab. 3.2) und die erweiterten Kalman Filter Gleichungen um aus der gemessenen Kameraposition die Fahrzeugposition zu schätzen. Der Knoten veröffentlicht die geschätzte Position und Orientierung des Fahrzeugs in $\{o\}$.

(a) Originalbild. (b) Erkannte Tags im Bild.

Abbildung 5.3: Experimentaufbau, dargestellt durch Bildaufnahme des Roboters.

Führung

Das Führungssystem ist in zwei ROS-Knoten realisiert. Im Knoten `/path_generator` sind die Gleichungen zur Berechnung einer Bézierkurve aus Abschnitt 4.3 implementiert. In diesem Knoten wird abhängig vom Winkel α, der Distanz d und der aktuellen Fahrzeugposition einmalig das Kontrollpolygon für die Bézierkurve berechnet, siehe Gl. (4.11). Die Gleichung zur Kurvenberechnung, siehe Gl. (4.9), wird an 200 Punkten ausgewertet und übergibt die Wegpunkte an den Knoten `/path_follower`. In ihm werden mit einer Frequenz von 30 Hz die Berechnungen zur Pfadverfolgung durchgeführt. Er iteriert über den Pfad und bestimmt den nächsten Wegpunkt anhand einer vordefinierten Lookahead Distanz l. Sie wird zunächst willkürlich als 10 cm gewählt. Neben der Wegpunktverfolgung sind in diesem Knoten auch die Berechnungen für Sollwertregelung der Fahrzeugtiefe und des Gierwinkels integriert. Eine sequentielle Aktivierung der Regler wird im Pfadverfolgungs-Knoten mit einfacher Logik realisiert. Zu Beginn wird die Tiefenregelung aktiviert. Sobald der Roboter die Soll-Tiefe erreicht hat, wird das Regelungssystem für die Bewegung in der x-y-Ebene gestartet. Führungsgrößen und Aktivierungssignale werden von dem Knoten an das Regelungssystem übergeben.

Regelung

Die SISO-Regelungsarchitektur aus Abschnitt 4.4 wird über vier einzelne Knoten realisiert: `/surge_control`, `/sway_control`, `/heave_control` und `/yaw_control`. Die Benennung ist an die englischen Begrifflichkeiten aus [SNAME50] angelehnt. Jeder Knoten ist ein unabhängiger PID-Regler, welcher nach [ChungFuKröger16] diskret implementiert ist. Alle Regler-Knoten abonnieren jeweils ihre zugehörige Führungsgröße und übergeben die berechnete Stellgröße an den Knoten `/mixer`. Der Mixer-Knoten bestimmt anhand der Mixer-Matrix, siehe Gl. (4.19), die Steu-

erzuweisung zu den einzelnen Propellern und sendet diese über MAVROS an den Pixhawk4, welcher die Motoren der Propeller steuert.

Greifer

Der Greifarm wird starr am BlueROV2 angebracht und mit der Batterie und einem RaspberryPi verbunden. Auf dem RaspberryPi muss ein ROS-Knoten laufen, um die GPIO-Pins ansteuern zu können. Der Greifarm kann nur in drei Zuständen betrieben werden: Öffnen, Neutral, Schließen. Zu Beginn eines Greifprozesses wird definiert, dass der Greifer voll geöffnet sein muss. Sobald die Führungsarchitektur den letzten Wegpunkt des Pfads erreicht hat, wird der Greifarm über ein PWM-Signal zum Schließen kommandiert.

5.3 Analyse der Lokalisierung im Experiment

Simulationsumgebungen wie Gazebo basieren auf mathematischen Modellen, welche die Wirklichkeit nie ideal abbilden. Um die realistischen Grenzen der visuellen Lokalisierung zu identifizieren, wird diese daher im Hardware-Experiment untersucht. Zusätzlich wird der Einfluss unterschiedlicher Kovarianzmatrizen auf die Ergebnisse der Positionsschätzung untersucht. Der Versuchsaufbau hierfür wird nachfolgend beschrieben.

Das BlueROV2 wird starr an einer xyz-Verfahreinheit befestigt. Mit ihr können vordefinierte Strecken mit bestimmter Geschwindigkeit abgefahren werden. Die Verfahreinheit liefert damit die sogenannte *Ground Truth*-Referenz für die Messungen, da die Position des Fahrzeugs auf wenige Millimeter genau bestimmt werden kann. Die Referenz dient in den folgenden Versuchen als Vergleichsgrundlage zur Bewertung der Genauigkeit.

Um die Richtigkeit der definierten Kamera- bzw. Greiferposition in Tab. 3.2 zu testen, wird die Zielkonfiguration mithilfe der Verfahreinheit künstlich angefahren. In dieser Position liegt das Objekt-KOS $\{o\}$ also genau im Mittelpunkt des Greifers. Dann lässt sich die tatsächliche Zielkonfiguration bestimmen zu

$$\boldsymbol{q}_d^o = \begin{bmatrix} 0{,}4\,\mathrm{m} & 0\,\mathrm{m} & 0{,}08\,\mathrm{m} & 0\,^\circ \end{bmatrix}^\top . \tag{5.1}$$

Die von der Lokalisierung geschätzten Werte in dieser Zielkonfiguration weichen nur sehr gering von den berechneten Koordinaten ab

$$\boldsymbol{q}_{d,\mathrm{exp}}^o = \begin{bmatrix} 0{,}399\,\mathrm{m} & 0{,}005\,\mathrm{m} & 0{,}086\,\mathrm{m} & 0{,}6\,^\circ \end{bmatrix}^\top . \tag{5.2}$$

Somit werden die statischen Transformationen der Sensorpositionen im Fahrzeug-KOS $\{o\}$ als ausreichend genau angesehen.

5.3.1 Tiefenbestimmung

Für die Tiefenregelung des Roboters muss ein genaue Messung des Ist-Werts der Tiefe vorhanden sein. In diesem Abschnitt wird die Bestimmung der z-Koordinate durch eine Schätzung der visuellen Lokalisierung, sowie die Tiefenmessung durch einen Drucksensor mit der Referenz verglichen. Hierzu wird eine 33 cm lange Strecke in z-Richtung zwei mal hin und zurück abgefahren. Einmal mit einer Geschwindigkeit von 5 cm/s und danach mit der halben Geschwindigkeit 2,5 cm/s. Hierbei ist die Kovarianzmatrix der Zustandsschätzung zu $\boldsymbol{R} = 0{,}1\boldsymbol{E}$ gewählt, wobei $\boldsymbol{E}$ die Einheitsmatrix mit Dimensionen der Kovarianzmatrix ist.

In Abb. 5.4 ist erkennbar, dass die Messung durch den Drucksensor den wahren Zustand besser abbildet, als die Schätzung der visuellen Lokalisierung. Das Messrauschen des Drucksensors ist jedoch höher als das, der visuellen Zu-

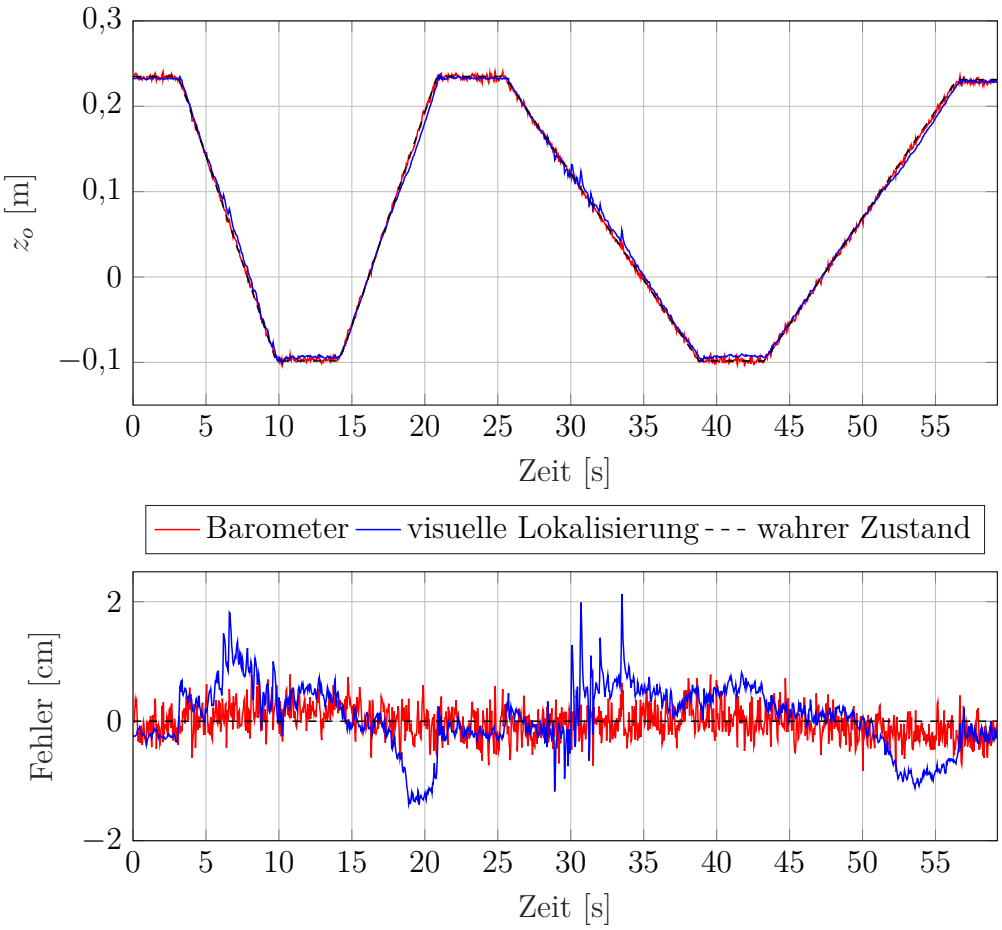


Abbildung 5.4: Zustandsschätzung für z durch visuelle Lokalisierung und Drucksensor im Vergleich zum wahren Zustand im Experiment.

standsschätzung. Hierbei sei erwähnt, dass der Fehler bei beiden Messungen im Bereich weniger Zentimeter liegt und somit beide Bestimmungsmethoden ausreichend genau erscheinen.

5.3.2 Bewegung in der xy-Ebene

In diesem Abschnitt wird die Genauigkeit der Positionsschätzung in der Ebene bei zwei verschiedenen Kovarianzmatrizen für das EKF untersucht. Mit den Informationen aus Abschnitt 4.2.2 ist bekannt, dass eine Kovarianzmatrix $\boldsymbol{R}$ mit großen Einträgen dazu führt, dass ein großes Messrauschen im Kalman Filter modelliert wird. In diesem Fall wird der Messung weniger vertraut, als bei einer geringerer Kovarianz. Um eine geeignete Kovarianzmatrix $\boldsymbol{R}$ für die Zustandsschätzung zu ermitteln, werden heuristisch zwei Fälle untersucht: $\boldsymbol{R} = R\boldsymbol{E}$ mit $R \in \{0{,}1; 1\}$, wobei $\boldsymbol{E}$ die Einheitsmatrix mit Dimensionen der Kovarianzmatrix ist.

Zur Untersuchung der Positionsschätzung mit den gewählten Kovarianzmatrizen wird ein Pfad in 0,4 m Schritten wie in Abb. 5.5 dargestellt abgefahren. Hierbei wird in der Zielkonfiguration gestartet und die Wegpunkte mit einer Geschwindigkeit von 10 cm/s angefahren. Bei dieser Geschwindigkeit hat die Trägheit des Systems unter Wasser einen merklichen Einfluss sein Bewegungsverhalten. Beim Erreichen eines Wegpunktes schwingt der Roboter an der Verfahreinheit merklich durch das abrupte Halten.

In Abb. 5.5 ist erkennbar, dass die Lokalisierung für beide Kovarianzmatrizen eine ausreichend genaue Positionsschätzung zu liefern scheint. In Abb. 5.6 wird deutlich, dass die Schätzung mit der geringeren Kovarianz das Schwingen des Fahrzeugs an der Verfahreinheit nach dem Halten an einem Wegpunkt besser erfasst. Dies ist darin begründet, dass der Messung mehr vertraut wird. Die Matrix mit $R = 0{,}1$ hat folglich eine schnellere Reaktionszeit.

Um die Unsicherheit der Schätzung für das Greifen klassifizieren zu können, wird eine weitere Messreihe aufgenommen. Bei dieser Reihe wird die Varianz der Schätzung in Abhängigkeit von der Distanz zwischen Fahrzeug und Objekt untersucht. Hierfür wird das Fahrzeug über eine Strecke von 1,2 m mit einer Geschwindigkeit von 1 cm/s vom Objekt weg bewegt. Die Ergebnisse sind in Abb. 5.7 dargestellt. Es ist erkennbar, dass die Varianz der Schätzung mit dem Abstand zunimmt. Wie erwartet ist die Schätzung durch das EKF bei einer Kovarianz-Matrix $\boldsymbol{R}$ mit $R = 0{,}1$ mit einer höheren Unsicherheit behaftet, da das Messrauschen bei dieser Schätzung größeren Einfluss hat. Die Spitzen in den Messwerten werden dadurch verursacht, dass ab einer bestimmten Distanz zum Objekt Tags sichtbar werden, welche zuvor durch den Greifarm im Kamerabild verdeckt waren. Durch ihre Erkennung verändern sich die Messwerte ruckartig, was zu einer kurzzeitig höheren Varianz der Messungen führt.

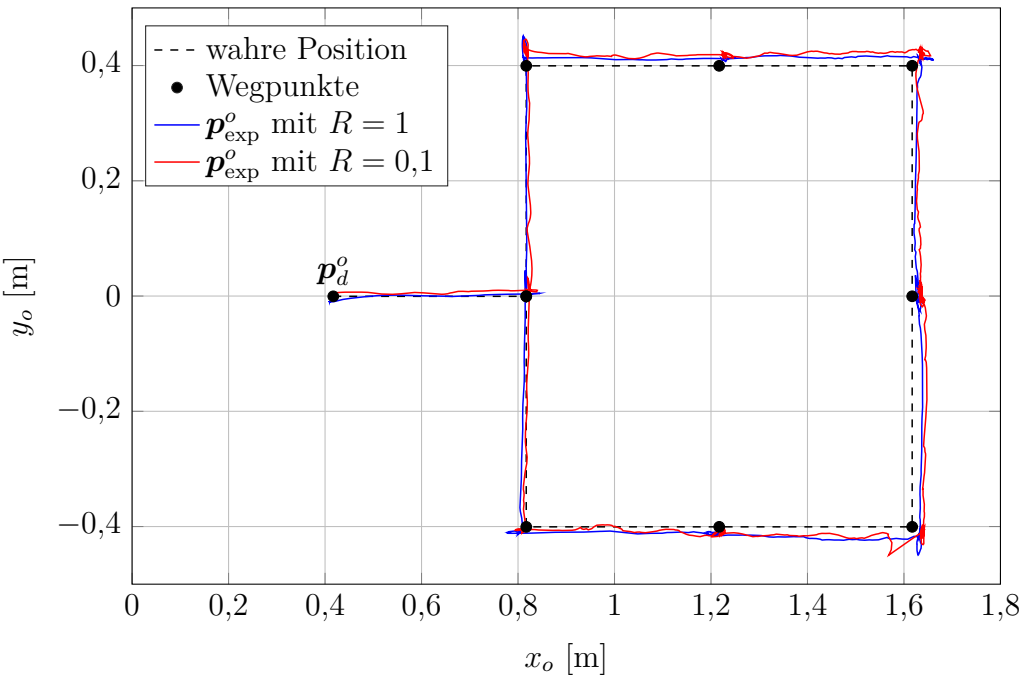


Abbildung 5.5: Vergleich der absoluten Positionsschätzung $\boldsymbol{p}^o_{\text{exp}}$ im Objekt-KOS durch die visuelle Lokalisierung bei unterschiedlichen Mess-Kovarianzmatrizen. Die Wegpunkte wurden schrittweise im Uhrzeigersinn beginnend bei $\boldsymbol{p}^o_d$ abgefahren. Das Fahrzeug war hierbei mit der Kamera zum Objekt, also in negative x_o-Richtung ausgerichtet.

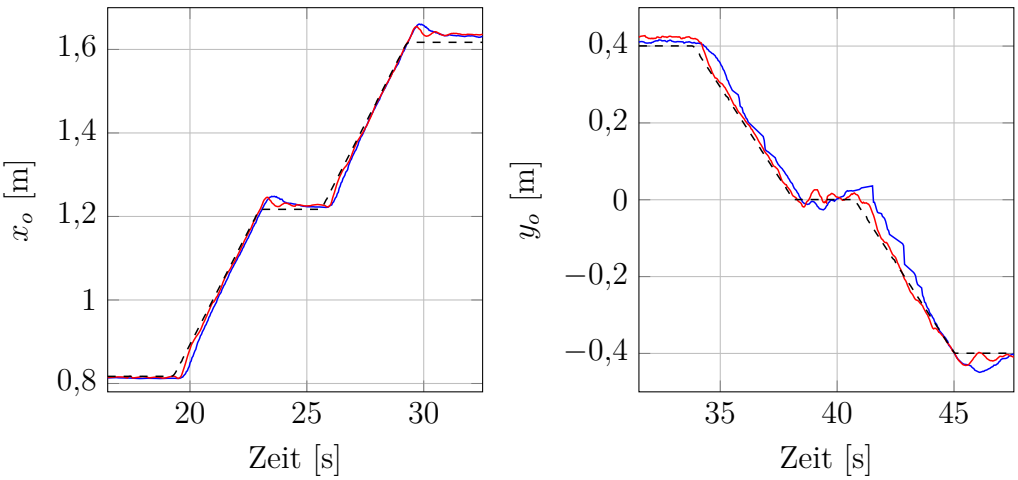


Abbildung 5.6: Geschätzte x- und y-Koordinate durch die visuelle Lokalisierung bei unterschiedlichen Mess-Kovarianzmatrizen über der Zeit. Es gilt die gleiche Farbzuordnung wie in Abb. 5.5.

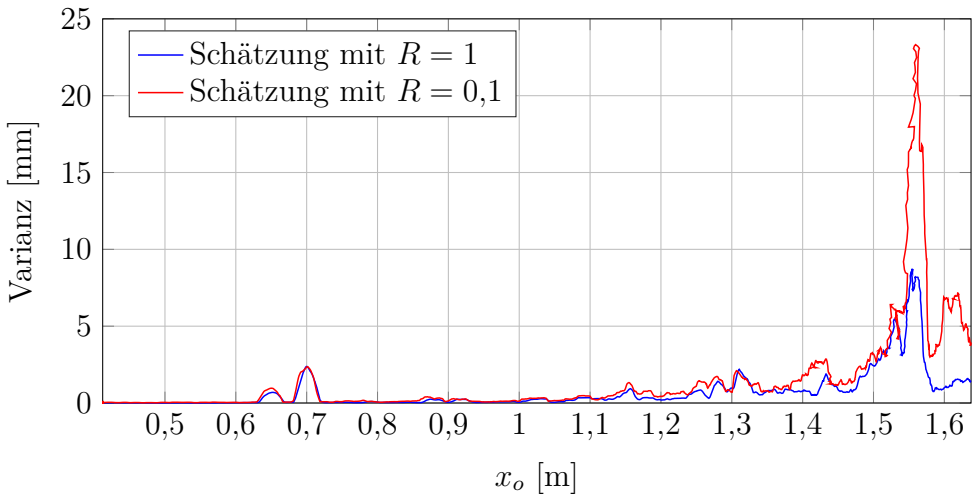


Abbildung 5.7: Varianz der Positionsschätzung $\boldsymbol{p}^o_{\text{exp}}$ bei einer Bewegung entlang der x_o-Achse für zwei verschiedene Mess-Kovarianzmatrizen. Bei $x_o = 0{,}65\,\text{m}$, $x_o = 0{,}7\,\text{m}$ und $x_o = 1{,}5\,\text{m}$ werden vorher verdeckte Tags erkannt.

5.4 Analyse der Bewegungsplanung und Regelung

In diesem Abschnitt wird zunächst die Funktionsweise der entwickelten Pfadplanungs- und Verfolgungsalgorithmen und der Regelung unabhängig von der Lokalisierung in einer Gazebo-Simulation untersucht. Danach wird das System ganzheitlich im Hardware-Experiment getestet.

Da die Funktion der Algorithmen geprüft werden soll, werden die Parameter zur Pfadberechnung und -verfolgung für die nachfolgenden Untersuchungen heuristisch dimensioniert zu

$$l = 10\,\text{cm}, \quad d = 1{,}6\,\text{m} \quad \text{und} \quad \alpha = 0\,^\circ. \tag{5.3}$$

Diese Werte für die *Lookahead Distance* l, die Distanz d und den Winkel α führen in der Simulation und der realen Umgebung, wie nachfolgend gezeigt, zu zufriedenstellenden Ergebnissen.

5.4.1 Simulation

Die in Kap. 4 entwickelten Führungs- und Regelungsalgorithmen werden zunächst in einer idealen Simulation auf ihre Funktion und Zuverlässigkeit untersucht. Die Fahrzeugkonfiguration $\boldsymbol{q}^o_{\text{sim}}$ ist hierbei durch den wahren Zustand, die *Ground*

Truth, in Gazebo gegeben. Die Konfiguration ist daher ideal und beinhaltet keine Unsicherheiten oder Ungenauigkeiten durch eine Lokalisierung.

Die Regelkreise werden mit den bekannten Auswirkungen des P-, I- und D-Anteils empirisch nach [ÅströmHägglund95] dimensioniert zu

$$\boldsymbol{K}_p = \mathrm{diag}(2{,}0 \;\; 2{,}0 \;\; 0{,}6 \;\; 1{,}4), \tag{5.4}$$
$$\boldsymbol{K}_i = \mathrm{diag}(0{,}1 \;\; 0{,}1 \;\; 0{,}5 \;\; 0{,}6), \tag{5.5}$$
$$\boldsymbol{K}_d = \mathrm{diag}(0 \;\; 0 \;\; 0{,}2 \;\; 0{,}3). \tag{5.6}$$

Die Sättigungsgrenzen der Regler werden definiert als

$$u_{x,max} = 0{,}5 \qquad u_{y,max} = 0{,}5 \qquad u_{z,max} = 1 \qquad u_{\psi,max} = 0{,}5 \tag{5.7}$$
$$u_{x,i,max} = 0{,}5 \qquad u_{y,i,max} = 0{,}5 \qquad u_{z,i,max} = 0{,}1 \qquad u_{\psi,i,max} = 0{,}2. \tag{5.8}$$

Abbildung 5.8 zeigt exemplarische Sprungantworten der Tiefen- und Gierwinkelregelung bei den genannten Reglerparametern.

Abbildung 5.9 zeigt die Pfadverfolgung bei drei beliebig aber realistisch gesetzten Anfangspositionen. Abbildung 5.10 zeigt den Regelfehler der Gierwinkelregelung während der Pfadverfolgung. Die Regelabweichung der Tiefe wird nicht dargestellt, da diese in der idealen Simulation von der Bewegung in der Ebene entkoppelt ist und somit nicht durch diese beeinflusst wird.

In Abb. 5.9 ist erkennbar, dass die Pfade nicht exakt verfolgt werden, das Fahrzeug aber trotzdem die Zielpositionen mit geringer Abweichung zuverlässig erreicht. Ein Grund für die Abweichungen in der Pfadverfolgung kann eine schlecht

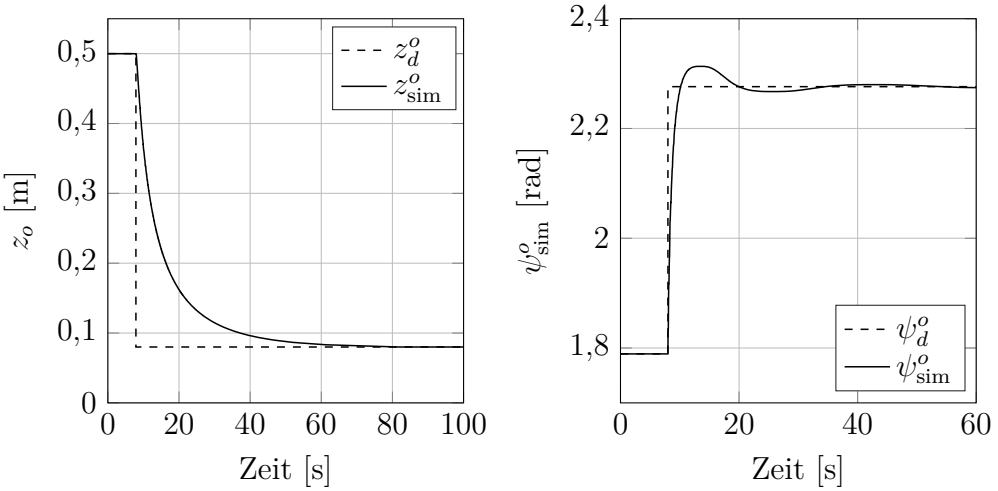


Abbildung 5.8: Exemplarische Sprungantworten für Sollwertregelung von Fahrzeugtiefe z_{sim}^o und Gierwinkel ψ_{sim}^o in Simulation.

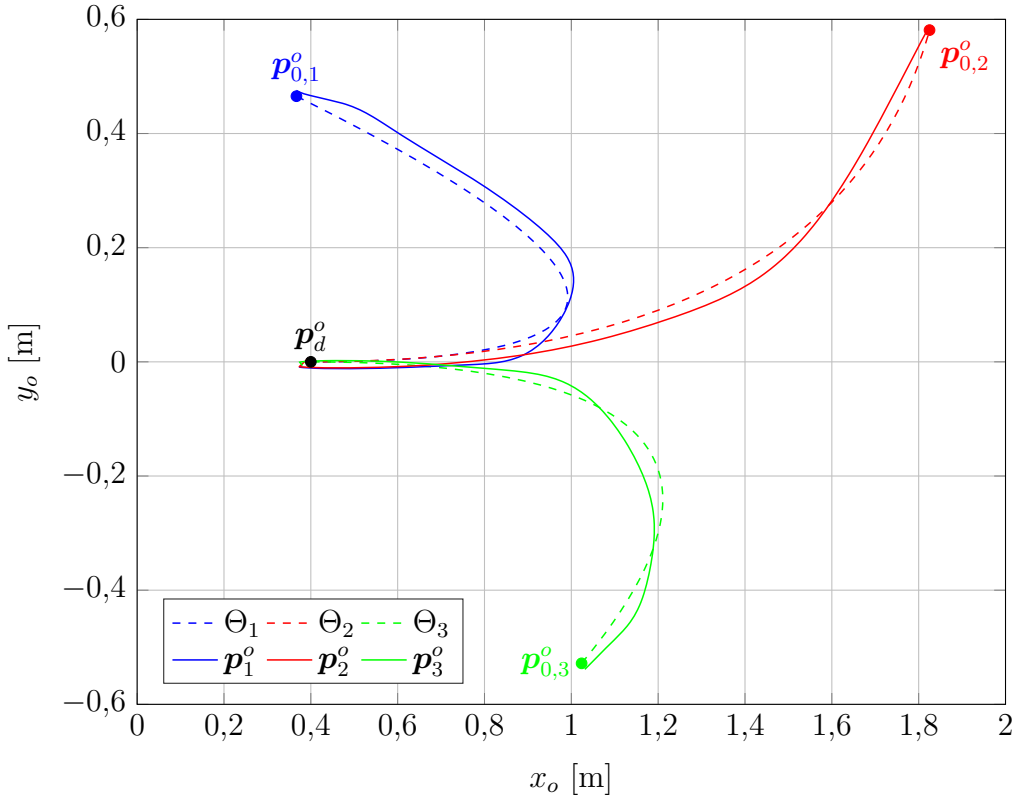


Abbildung 5.9: Generierte Pfade Θ_1, Θ_2 und Θ_3 und Pfadverfolgung $\boldsymbol{p}^o_1$, $\boldsymbol{p}^o_2$ und $\boldsymbol{p}^o_3$ in der x-y-Ebene bei drei verschiedenen Startpositionen in der Simulation.

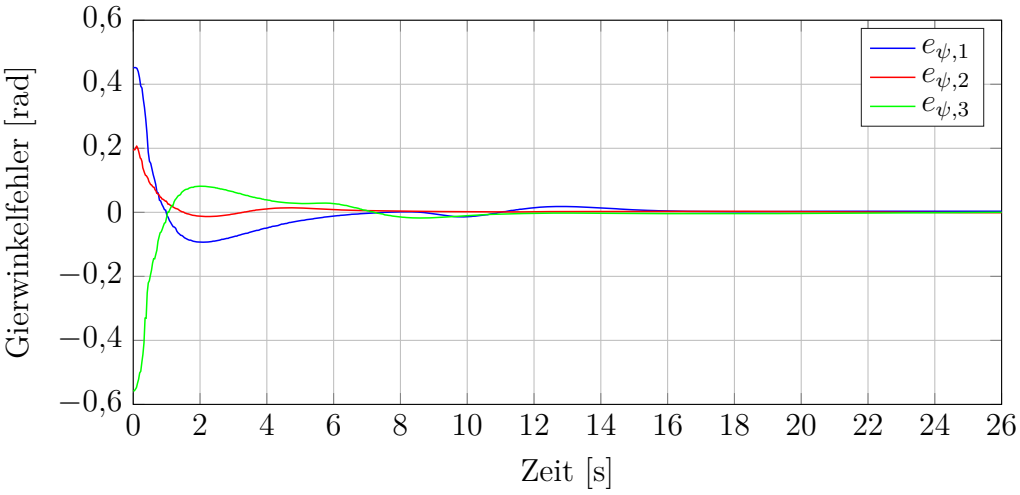


Abbildung 5.10: Gierwinkelfehler während der Pfadverfolgung von Θ_1, Θ_2 und Θ_3 in der Simulation.

gewählte *Lookahead Distance* sein. Auch zu hohe Parameter der Regler für die x_b- und y_b-Bewegungsregelung können dafür sorgen, dass der Pfad nicht optimal verfolgt wird. Abbildung 5.10 visualisiert, wie gut das der Gierwinkel-Regler während der Pfadverfolgung die Orientierung des Fahrzeugs in Richtung des Objektes halten konnte.

5.4.2 Experiment

In diesem Abschnitt wird das in Kap. 4 entwickelte System ganzheitlich im Hardware-Experiment untersucht. Für die Tiefenregelung werden Messungen des Drucksensors verwendet, da diese die Fahrzeugtiefe genauer ermitteln als die visuelle Lokalisierung, wie in Abschnitt 5.3 gezeigt. Die Kovarianzmatrix wird zu $\boldsymbol{R} = 0{,}1\boldsymbol{E}$ gewählt, da das Verhalten der Zustandsschätzung mit diesen Werten in Abschnitt 5.3 gute Leistungen erbracht hat.

Die Regler-Parameter werden erneut empirisch nach [ÅströmHägglund95] dimensioniert zu

$$\boldsymbol{K}_p = \text{diag}(2{,}0 \;\; 2{,}0 \;\; 2{,}0 \;\; 1{,}4), \tag{5.9}$$

$$\boldsymbol{K}_i = \text{diag}(0{,}1 \;\; 0{,}1 \;\; 1{,}0 \;\; 0{,}2), \tag{5.10}$$

$$\boldsymbol{K}_d = \text{diag}(0 \;\; 0 \;\; 0 \;\; 0). \tag{5.11}$$

Hierbei sind die D-Anteile gleich Null gesetzt, da die Zustandsmessungen in der Realität mit großem Messrauschen behaftet sind. Der D-Anteil des Reglers verstärkt Änderungsraten des Eingangssignals und wird somit die schnelle Änderung der Zustandsschätzung durch das Messrauschen verstärken. Dies führt zu unerwünschtem Regelkreisverhalten. Die Sättigungsgrenzen der Regler werden definiert als

$$u_{x,max} = 0{,}5 \qquad u_{y,max} = 0{,}5 \qquad u_{z,max} = 1 \qquad u_{\psi,max} = 0{,}5 \tag{5.12}$$

$$u_{x,i,max} = 0{,}5 \qquad u_{y,i,max} = 0{,}5 \qquad u_{z,i,max} = 0{,}5 \qquad u_{\psi,i,max} = 0{,}5. \tag{5.13}$$

Das Fahrzeug wurde beliebig an drei Startpositionen ($\boldsymbol{p}^o_{0,1}$, $\boldsymbol{p}^o_{0,2}$ und $\boldsymbol{p}^o_{0,3}$) im Tank platziert. Für diese Startpositionen galt nur die Bedingung, dass die Tags am Objekt durch die AprilTag-Erkennung erkannt werden. Abbildung 5.11 zeigt die generierten Pfade und realen Positionen des Fahrzeugs während der Pfadverfolgung. In Abb. 5.12 sind die Regelabweichungen für die Tiefen- und Gierwinkelregelung während der Pfadverfolgung dargestellt.

Auch im Hardware-Experiment ist erkennbar, dass die Pfade nicht exakt verfolgt werden, aber die Zielposition $\boldsymbol{p}^o_d$ dennoch zuverlässig und genau erreicht wird. Gerade zu Beginn der Pfadverfolgung sind die Abweichungen zum Pfad hoch und eine deutliche Oszillation kann bei $\boldsymbol{p}^o_2$ und $\boldsymbol{p}^o_3$ beobachtet werden. Eine mögliche

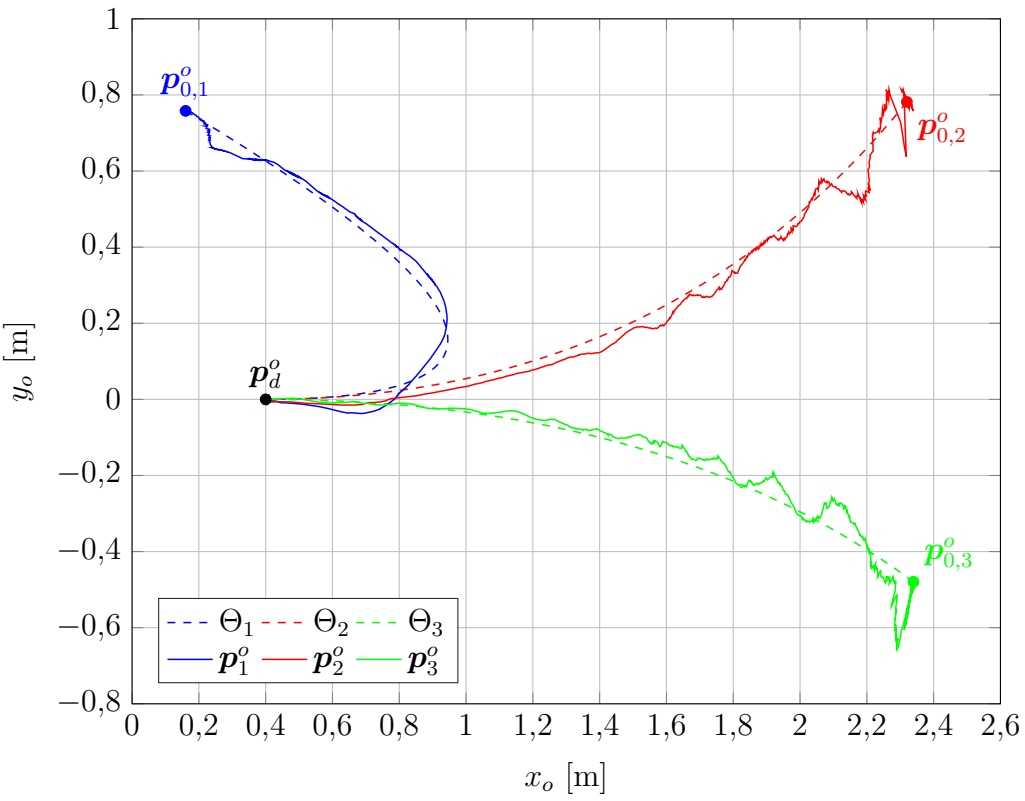


Abbildung 5.11: Generierte Pfade und Pfadverfolgung in der x-y-Ebene bei drei verschiedenen Startpositionen im Experiment.

Begründung hierfür liegt in dem Verhalten des Gierwinkelreglers. In Abb. 5.12 kann auch hier oszillierendes Verhalten beobachtet werden. Da die Pfadverfolgung mit dem Gierwinkel die Bewegungsrichtung zum nächsten Wegpunkt bestimmt, ist diese beeinflusst durch den aktuellen Gierwinkel. In Abb. 5.12 ist außerdem zu erkennen, dass bei der Verfolgung vom Pfad Θ_2 absichtlich eine Störung bzgl. der Tiefe verursacht wurde. Diese wurde bis zum Erreichen des Objekts korrigiert, was für eine Robustheit des Tiefenreglers spricht.

Nach der heuristischen Dimensionierung der Regelungs-, Pfadplanungs- und Pfadverfolgungsparameter gelang es dem System in allen drei Versuchen die Greifbacken zuverlässig um das Objekt zu platzieren und somit einen Griff zu ermöglichen. Das Anfahren dauerte im Experiment zwischen 30 und 40 Sekunden.

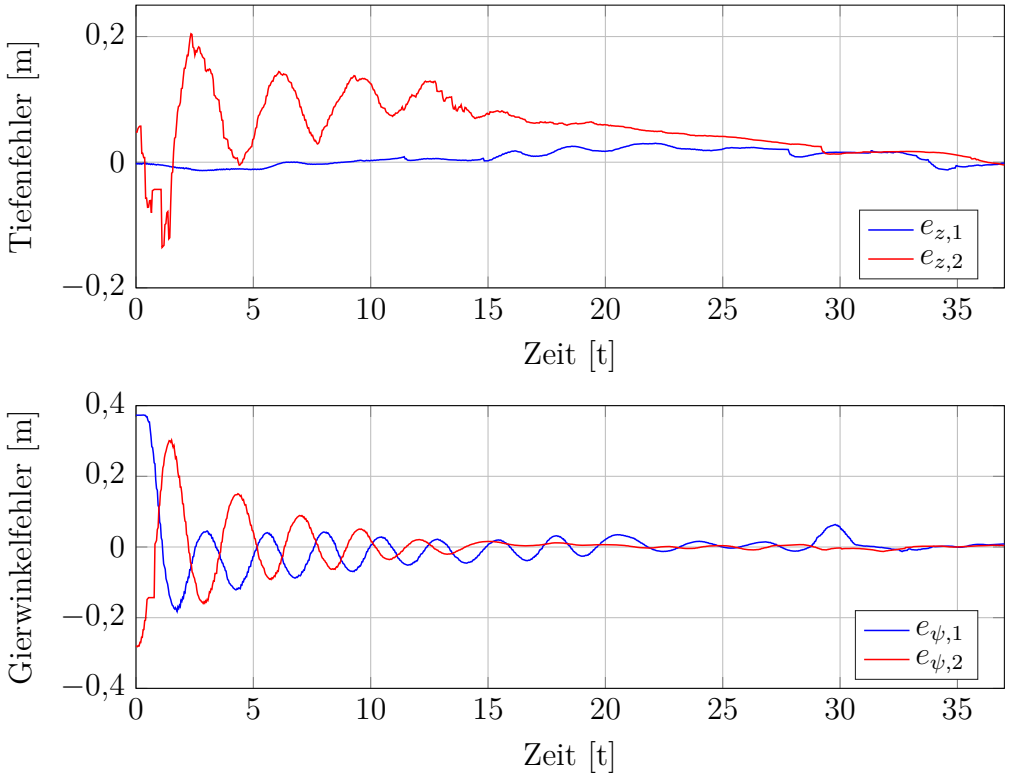


Abbildung 5.12: Tiefen- und Gierwinkelfehler während der Pfadverfolgung von den Pfaden Θ_1 und Θ_2 im Experiment.

Kapitel 6

Fazit

Im Rahmen dieser Arbeit wurde eine Systemarchitektur für autonomes Greifen mit einem Unterwasserroboter entwickelt. Die entwickelte Methode baut auf einer bereits vorhandenen visuellen Lokalisierung auf und beinhaltet leistungsfähige Algorithmen zur Pfadplanung und -regelung. Die Bestandteile der Architektur wurden mittels der BlueROV2-Unterwasserdrohne in Simulation und Experiment untersucht. Die folgenden Abschnitte fassen die Ergebnisse der Arbeit zusammen und geben einen Ausblick auf zukünftige Weiterentwicklungsmöglichkeiten des entwickelten Systems.

6.1 Zusammenfassung

Nach einer Einführung in die Grundlagen der Bewegungsplanung und -regelung in Kap. 2 wurde in Kap. 3 das im Zusammenhang mit dieser Arbeit untersuchte System, das BlueROV2, vorgestellt. Am Beispiel des BlueROV2 wurde das mathematische Modell für Unterwasserfahrzeuge nach [Fossen11] erklärt. Anschließend wurde die Hardware und Software des verwendeten Systems ausführlich beschrieben.

In Kap. 4 wurde ein System zur Führung, Navigation und Regelung bei Greifprozessen mit dem BlueROV2 entwickelt. Zunächst wurde eine Formulierung des vorliegenden Problems mit Vereinfachungen vorgenommen. Anschließend wurde eine vorhandene visuelle Lokalisierungsmethode, welche auf der Erkennung visueller Landmarken und Filterung durch ein erweitertes Kalman Filter basiert, nach [DueckerEtAl20] beschrieben. Mit dem Ziel den Roboter in eine bestimmte Konfiguration zu steuern, wurde danach ein Pfadplanungsalgorithmus basierend auf Bézierkurven hergeleitet. Eine Pfadverfolgungsmethode wurde angelehnt an *Pure*

Pursuit entwickelt und die Bewegungsregelung des Fahrzeugs durch PID-Regler realisiert.

In Kap. 5 wurde die entwickelte Architektur implementiert und untersucht. Hierzu wurde zunächst die Simulation mit Gazebo und das Hardware-Experiment beschrieben. Die Architektur wurde nachfolgend mithilfe von ROS implementiert. Um die Funktion der einzelnen Bestandteile des Systems zu gewährleisten, wurden diese vorerst isoliert voneinander untersucht. Die Lokalisierung wurde in einem Hardware-Experiment auf Genauigkeit und Robustheit getestet. Die Algorithmen zur Pfadplanung und Bewegungsregelung wurden entkoppelt von der Lokalisierung in einer Simulation untersucht. Abschließend wurde das System ganzheitlich getestet und die Funktion und Zuverlässigkeit in einem Experiment gezeigt.

6.2 Ausblick

Die Entwicklung des Systems zur Bewegungsplanung und -regelung hat mehrere Anknüpfungspunkte für Weiterentwicklungen aufgezeigt. Eine Möglichkeit stellt die umfangreiche experimentelle Untersuchung der Pfadplanung und -verfolgung dar. Sowohl die verwendeten Berechnungsparameter des Kontrollpolygons der Bézierkurven als auch die vorausschauende Distanz der Pfadverfolgung wurden nur heuristisch ermittelt und nicht weiter untersucht. Hier bietet sich eine strukturierte Analyse der Größen und ihrer Einflüsse auf Pfadgenerations- und Pfadverfolgungsverhalten an. Gleiches gilt für das Regelkreis-Design. Hierbei können neben der Untersuchung der PID-Reglerparameter auch andere Regelungsalgorithmen wie Gleitregimeregelung (engl. Sliding Mode Control) oder linear-quadratische Regelung getestet werden.

Eine andere interessante Weiterentwicklung der Bewegungsplanung stellt die Verwendung anderer Pfadplanungsmethoden dar. Im Zusammenhang hiermit ist auch die Berücksichtigung von Hindernissen in der Arbeitsumgebung des Roboters denkbar. Zusätzlich kann die Zeit-Skalierung von Pfaden bzw. Trajektoriengeneration betrachtet werden.

Das Greifen stellt wie in Kap. 2 erwähnt nur den ersten Schritt einer Manipulation der Umgebung dar. Aufbauend auf dem Greifprozess können Algorithmen für das Umorientieren, Verschieben oder Transportieren eines gegriffenen Objekts entwickelt werden. Eine Herausforderung, die sich hierbei ergibt, ist die veränderte Dynamik des Roboters. Es kann untersucht werden, welche Kräfte und Momente für verschiedene Manipulationsaufgaben notwendig sind und wo die Grenzen des Systems hierbei liegen.

Literaturverzeichnis

[Antonelli14] Antonelli, G.: Underwater Robots. Cham, Schweiz: Springer International Publishing, 3 Edn., 2014.

[Bauschmann18] Bauschmann, N.: Visuelle On-Board-Lokalisierung mit einem Partikelfilter als Grundlage für die Regelung von Unterwasserrobotern. Bachelorarbeit BSC-087, Institut für Mechanik und Meerestechnik, Technische Universität Hamburg, 2018.

[BicchiKumar00] Bicchi, A.; Kumar, V.: Robotic grasping and contact: a review. In IEEE International Conference on Robotics and Automation (ICRA), S. 348–353. San Francisco, CA, USA, 2000.

[BillardKragic19] Billard, A.; Kragic, D.: Trends and challenges in robot manipulation. Science, Bd. 364, Nr. 6446, S. 1162–1170, 2019.

[BlueRobotics21] BlueRobotics: Internetseite des Unternehmens. `https://bluerobotics.com`, 2021. Aufgerufen am 12. November 2021.

[ChalonEtAl11] Chalon, M.; Wedler, A.; Baumann, A.; Bertleff, W.; Beyer, A.; Butterfaß, J.; Grebenstein, M.; Gruber, R.; Hacker, F.; Kraemer, E.; Landzettel, K.; Maier, M.; Sedlmayr, H.J.; Seitz, N.; Wappler, F.; Willberg, B.; Wimboeck, T.; Hirzinger, G.; Didot, F.: Dexhand: A Space qualified multifingered robotic hand. In IEEE International Conference on Robotics and Automation (ICRA), S. 2204–2210. Shanghai, China, 2011.

[ChoiCurryElkaim08] Choi, J.w.; Curry, R.; Elkaim, G.: Path Planning Based on Bézier Curve for Autonomous Ground Vehicles. In Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science, S. 158–166. London, U.K., 2008.

[ChungFuKröger16] Chung, W.K.; Fu, L.C.; Kröger, T.: Motion Control. In Springer Handbook of Robotics, S. 163–194. Berlin Heidelberg: Springer, 2016.

[Coulter92] Coulter, R.C.: Implementation of the Pure Pursuit Path Tracking Algorithm. Techn. bericht., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[DueckerEtAl20] Duecker, D.A.; Bauschmann, N.; Hansen, T.; Kreuzer, E.; Seifried, R.: Towards Micro Robot Hydrobatics: Vision-based Guidance, Navigation, and Control for Agile Underwater Vehicles in Confined Environments. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas, USA, 2020.

[Farin02] Farin, G.: Curves and Surfaces for CAGD: A Practical Guide. San Francisco, CA, USA: Morgan Kaufmann Publishers, 5. Edn., 2002.

[Fossen94] Fossen, T.I.: Guidance and Control of Ocean Vehicles. Chichester, West Sussex, U.K: Wiley, 1994.

[Fossen11] Fossen, T.I.: Handbook of marine craft hydrodynamics and motion control. Chichester, West Sussex, U.K: Wiley, 2011.

[Hastedt19] Hastedt, P.: Development and Experimental Validation of a Software-in-the-Loop Simulation Environment for Micro Underwater Robots. Projektarbeit PRO-034, Institut für Mechanik und Meerestechnik, Technische Universität Hamburg, 2019.

[Haykin01] Haykin, S.: Kalman Filtering and Neural Networks. New York, USA: Wiley, 2001.

[Huang08] Huang, H.M.: Autonomy Levels for Unmanned Systems (ALFUS) Framework. National Institute of Standards and Technology, 2008.

[Kalman60] Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering, Bd. 82, Nr. 1, S. 35–45, 1960.

[KavrakiLaValle16] Kavraki, L.E.; LaValle, S.M.: Motion Planning. In Springer Handbook of Robotics, S. 139–161. Berlin Heidelberg: Springer, 2016.

[KoenigHoward04] Koenig, N.; Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Bd. 3, S. 2149–2154. Sendai, Japan, 2004.

[KuffnerXiao16] Kuffner, J.; Xiao, J.: Motion for Manipulation Tasks. In Springer Handbook of Robotics, S. 897–930. Berlin Heidelberg: Springer, 2016.

[Latombe91] Latombe, J.C.: Robot Motion Planning. New York, USA: Springer, 1991.

[LiYang20] Li, J.; Yang, C.: AUV Path Planning Based on Improved RRT and Bezier Curve Optimization. In IEEE International Conference on Mechatronics and Automation (ICMA). Beijing, China, 2020.

[Malyuta17] Malyuta, D.: Guidance, Navigation, Control and Mission Logic for Quadrotor Full-cycle Autonomy. Masterarbeit, Jet Propulsion Laboratory, Pasadena, CA, USA, 2017.

[MeierHoneggerPollefeys15] Meier, L.; Honegger, D.; Pollefeys, M.: PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In IEEE International Conference on Robotics and Automation (ICRA). Seattle, WA, USA, 2015.

[Mogk20] Mogk, J.B.: Modeling and Experimental System Identification of an Underwater Vehicle. Bachelorarbeit, Forschungsgruppe smartPORT, Technische Universität Hamburg, 2020.

[MurrayLiSastry94] Murray, R.M.; Li, Z.; Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. Boca Raton, Florida, USA: CRC Press, 1994.

[Olson11] Olson, E.: AprilTag: A robust and flexible visual fiducial system. In IEEE International Conference on Robotics and Automation (ICRA), S. 3400–3407. Shanghai, China, 2011.

[PadenEtAl16] Paden, B.; Čáp, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E.: A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. IEEE Transactions on Intelligent Vehicles, Bd. 1, Nr. 1, S. 33–55, 2016.

[PiazziGuarinoRomano07] Piazzi, A.; Guarino, C.; Romano, M.: η^3-Splines for the Smooth Path Generation of Wheeled Mobile Robots. IEEE Transactions on Robotics, Bd. 23, Nr. 5, S. 1089–1095, 2007.

[QuigleyEtAl09] Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A.: ROS: an open-source Robot Operating System. In IEEE International Conference on Robotics and Automation (ICRA). Kobe, Japan, 2009.

[RE2Robotics21] RE2Robotics: Internetseite des Unternehmens. `https://www.resquared.com/`, 2021. Aufgerufen am 29. November 2021.

[RibasEtAl15] Ribas, D.; Ridao, P.; Turetta, A.; Melchiorri, C.; Palli, G.; Fernández, J.J.; Sanz, P.J.: I-AUV Mechatronics Integration for the TRIDENT FP7 Project. IEEE/ASME Transactions on Mechatronics, Bd. 20, Nr. 5, S. 2583–2592, 2015.

[Sandøy16] Sandøy, S.S.: System Identification and State Estimation for ROV uDrone. Masterarbeit, Department of Marine Technology, Norwegian University of Science and Technology, 2016.

[SchjølbergEtAl16] Schjølberg, I.; Gjersvik, T.B.; Transeth, A.A.; Utne, I.B.: Next Generation Subsea Inspection, Maintenance and Repair Operations. IFAC-PapersOnLine, Bd. 49, Nr. 23, S. 434–439, 2016.

[SNAME50] SNAME: Nomenclature for Treating the Motion of a Submerged Body Through a Fluid: Report of the American Towing Tank Conference. Technical and research bulletin. Society of Naval Architects and Marine Engineers, 1950.

[WangOlson16] Wang, J.; Olson, E.: AprilTag 2: Efficient and robust fiducial detection. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), S. 4193–4198. Daejeon, Korea, 2016.

[ÅströmHägglund95] Åström, K.J.; Hägglund, T.: PID Controllers: Theory, Design, and Tuning. Research Triangle Park, NC, USA: Instrument Society of America, 2. Edn., 1995.

Anhang

A.1 Inhalt Archiv

Im Archiv ist ein Verzeichnis **BSC_132_Grothusen/** abgelegt. Dieses enthält in der obersten Dateistruktur die Einträge

- **BSC_132_Grothusen.pdf**: das pdf-File zur Arbeit BSC-132.
- **Daten/**: ein Verzeichnis mit den für diese Arbeit relevanten Daten, Hilfsprogrammen, Skripts und Simulationsumgebungen.
- **Latex/**: ein Verzeichnis mit den *.tex-Dateien des in Latex verfassten Berichts zur Arbeit BSC-132 sowie alle dazugehörigen Grafiken (falls vorhanden auch als *.svg Dateien).
- **Vortrag/**: ein Verzeichnis mit den für den Vortrag relevanten Daten wie die Präsentation, Bilder und Videos.

Erklärung

Ich, Jannik Jorge Grothusen (Student der Allgemeinen Ingenieurwissenschaften an der Technischen Universität Hamburg, Matrikelnummer 52576), versichere, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.

Unterschrift	Datum