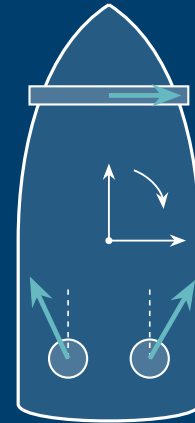


μDP

Portable Marine Control

Manuelle und autonome Steuerung von Offshore-Modellen auf ArduPilot, unabhängig vom Antrieb.
Eine Bedienung, ein Autopilot, jeder Rumpf.



Pierre-Louis Constant

Director / Lead Engineer, Oceanomatics Pty Ltd

Firmware-Basis: ArduPilot Rover 4.6

Überprüft mit jeder ArduPilot-Version

Zusammenfassung

μDP (micro dynamic positioning) ist das Steuerungssystem von Oceanomatics für Modelle von Offshore-Schiffen: DP-Funktionalität, von manuell bis vollständig autonom, auf Rümpfen von ein bis zwei Metern. Es entwickelt die Steuerung weiter, die 2019 für ein Modell-Versorgungsschiff entworfen wurde und bei der eine Modellbau-Fernsteuerung die Knüppel auf zwei Azimutantriebe und ein Bugstrahlruder mischte. Die Knüppelbelegung dieser Steuerung bleibt erhalten; das Mischen wird durch eine Schubverteilung ersetzt, die in die Motorbibliothek von ArduPilot eingebaut ist und aus einer Effektortabelle gespeist wird, so dass eine Methode Propeller mit Ruder, Azimutantriebe, Katamarane, Wasserstrahlantriebe und Voith-Schneider-Propeller abdeckt. Hinzu kommen RTK-Positionierung mit GNSS-Heading, eine Windvorsteuerung für Positions- und Bahnhalten, eine Strömungsschätzung, ein online identifiziertes und zwischen den Einsätzen gespeichertes Schiffsmodell, ein Konfigurationswerkzeug, das alle Eingaben des Stacks aus einem Rumpfumriss und einer Antriebsliste erzeugt, und eine Simulationskette, die das Erprobungsprotokoll vor den Wassererprobungen durchläuft.

Übersetzung des englischen Originals OCX-WP-2026-01, Rev. C. Bei Abweichungen gilt der englische Text. Die Quelltexte in den Anhängen sind unverändert und auf Englisch kommentiert.

Inhalt

1	Einleitung	3
2	Das Paradigma, und was sich ändert	3
2.1	Der Ansatz von 2019	3
2.2	Von den erwarteten Ausgängen zu den Gleichungen	4
2.3	Was sich ändert	4
3	Schiffs- und Effektormodelle	9
3.1	Koordinatensysteme und Notation	9
3.2	Effektortypen	9
3.3	Die Konfigurationsmatrix	13
3.4	Beispielkonfigurationen	13
4	Schubverteilung	15
4.1	Problemstellung	15
4.2	Nullraum und Bias-Modus	15
4.3	Sättigung und Vorrang	16
4.4	Schwenkbare Effektoren: Umschlagen, Sektor, Schwenkrate, Ausblenden	16
4.5	Fahrtabhängige Steuerung der Querstrahlruder	17
4.6	Drehpunkt	18
4.7	Erreichbare Kraftbereiche	18
5	Bedienschnittstelle	19
5.1	Eingänge, für jeden Rumpf identisch	19
5.2	Offener Kreis und ratengestützt manuell	19
5.3	Leerlauf, Bias, Scharfschalten	20
5.4	Sender- und Konsolenhardware	20
6	Integration in ArduPilot	22
6.1	Was Rover bietet und was ihm fehlt	22
6.2	Architektur	22
6.3	Woher die Forderung kommt	22
6.4	Warum nicht die Skript-Engine	23
6.5	Der Marine-Frame	23
6.6	Konfigurierbare Anordnungen: Effektoren, Halteeinrichtungen, Betriebsstellungen	24
6.7	Aktoren	25
6.8	Sensoren	26
6.9	Stromversorgung, Verkabelung, Wassereintritt	27
6.10	Konfigurationsübersicht	27
7	Autonomer Betrieb	29
7.1	Zuordnung der Modi	29
7.2	Positionshalten	29
7.3	Betrieb über die Bodenstation	29
7.4	Begleitrechner und Wahrnehmung	30
7.5	Wie weit ein Modell gehen kann	30

8	Störgrößen und ein selbstkorrigierendes Modell	31
8.1	Modell mit Störgrößen	31
8.2	Windvorsteuerung für Positions- und Bahnhalten	32
8.3	Strömung als Zustand des Stacks	33
8.4	Ein selbstkorrigierendes Rumpf- und Effektormodell	33
8.5	Persistenz	34
8.6	Wo es wohnt	34
9	Verifikation	34
9.1	Von den Paradigmen zu den Testfällen	34
9.2	Software in the Loop	36
9.3	Simulationsstufen	36
9.4	Gazebo: Rumpf, Effektoren und Hydrodynamik	37
9.5	ROS 2: Koordinatensysteme, Brücke und Testumgebung	37
9.6	Die Konfigurations-App	41
9.7	Identifikation am Prüfstand	42
9.8	Erprobung auf dem Wasser	42
10	Fahrplan	42
11	Fazit	43
A	Der Gleichungssatz von 2019, zur Dokumentation	45
B	Implementierungsskizze des Marine-Frames	46
C	SITL-Physikmodell	51
D	Parameterreferenz	53
E	Beispielhafte Hardware	54

1 Einleitung

2019 hat Oceanomatics die Steuerung eines Modell-Versorgungsschiffs entworfen: zwei Azimutantriebe achtern, ein Querstrahlruder vorn und eine Modellbau-Fernsteuerung als einziger Rechner an Bord. Das Mischen der Knüppel war im Sender implementiert und ist in [1] dokumentiert.

Seither hat sich die Anforderung auf andere Rumpf- und Antriebstypen erweitert, die ein Mischen im Sender nicht abdecken kann: Eine allgemeinere und flexiblere Steuerungssoftware wird gebraucht.

Auch die Werkzeuge haben sich geändert: Ein kleines Boot kann mit Firmware von der Stange auf RTK-GNSS die Position halten, und die Fernsteuerung ist eine von mehreren möglichen Bedienschnittstellen. Das neue Projekt heißt μDP, für *micro dynamic positioning*: DP-Funktionalität, von manuell bis vollständig autonom, auf Rümpfen von ein bis zwei Metern.

Dieses Papier

1. fasst den Ansatz von 2019 zusammen und benennt, was beibehalten und was ersetzt wird (Abschnitt 2);
2. verallgemeinert ihn: Jede Anordnung aus festen Strahlrudern, Azimutantrieben, Propeller-Ruder-Anlagen, Wasserstrahlantrieben oder Voith-Schneider-Propellern wird durch eine Tabelle von Effektoren beschrieben, aus der die Konfigurationsmatrix und die Schubverteilung folgen (Abschnitte 3 and 4);
3. wendet ihn auf ArduPilot an: manueller, ratengestützter und autonomer Betrieb, eine Implementierungsskizze für den Frame, Störgrößenvorsteuerung und ein selbstkorrigierendes Modell, ein Konfigurationswerkzeug, das die Eingaben des Stacks aus einem Rumpfumriss und einer Antriebsliste erzeugt, eine Simulationskette und ein Fahrplan (Abschnitte 5 to 10).

Zielgruppe sind Ingenieure und erfahrene Modellbauer, die mit Vektornotation und der Konfiguration eines Autopiloten vertraut sind, nicht unbedingt mit maritimer Schubverteilung. Herleitungen sind kurz gehalten; die Standardwerke sind [2, 3].

Die Methoden werden an Modellbooten erprobt, sind aber ebenso für unbemannte Überwasserfahrzeuge in voller Größe geschrieben. Schubverteilung, Störgrößbehandlung, Modellidentifikation und Konfigurationswerkzeug enthalten nichts, das vom Maßstab des Rumpfs abhängt; nur die Parameterwerte tun es.

Die Zusammenfassung in einem Satz

Die Knüppelbelegung von 2019 beibehalten (Längs, Quer, Gieren, Drehpunkt, Bias, Leerlauf). Das Mischen im Sender durch eine Schubverteilung ersetzen, die Position und Grenzen jedes Antriebs kennt. ArduPilot für Modi, Zustandsschätzung und Failsafes einsetzen. In der Simulation verifizieren, bevor auf dem Wasser erprobt wird.

2 Das Paradigma, und was sich ändert

2.1 Der Ansatz von 2019

Die Studie von 2019 [1] behandelte einen Rumpf mit einem programmierbaren Sender und seinem Simulator. Ihre Methode in sechs Schritten wird unverändert beibehalten; das Mischen, das sie hervorbrachte, nicht.

1. **Eingänge.** Festlegen, was der Bediener kommandiert: Längs- und Querbewegung, Gieren, einen Drehpunkt und einen Leerlauf/Bias-Schalter. Der Satz stammt von Offizieren mit Erfahrung auf diesem Schiffstyp und entspricht einem kommerziellen DP-Joystick.
2. **Ausgänge.** Jeden Aktor benennen und nummerieren.
3. **Paradigmen.** Jedes Manöver skizzieren (Abbildung 1): Wer schiebt wo, in welche Richtung, mit welchem Anteil.

4. **Gleichungen.** Die Blickrichtung umkehren: Für jeden Ausgang quer durch die Paradigmen lesen und ihn als Summe von Kurven der Eingänge schreiben (Abschnitt 2.2).
5. **Programmierung.** Die Gleichungen auf dem Zielsystem eingeben.
6. **Verifikation.** Jedes Paradigma im Simulator prüfen, bevor Hardware angefasst wird.

Drei Elemente bleiben, wie sie waren: der *Drehpunkt*-Eingang auf dem zweiten Knüppel, der das Drehzentrum wählt; der *Bias-Modus*, bei dem die Azimutantriebe für eine sofortige Schubantwort gegeneinander laufen, als Option beibehalten und bei Elektromodellen standardmäßig aus (Abschnitt 4.2); und die *Auswaschung des Querstrahlruders*, der Verlust seiner Wirksamkeit mit zunehmender Fahrt.

2.2 Von den erwarteten Ausgängen zu den Gleichungen

Schritt vier trägt die Methode und ist vom Zielsystem unabhängig. Die Paradigmen von Schritt drei sind von der Eingangsseite her formuliert (Längsknüppel voraus: beide Antriebe drehen nach achtern, beide Schübe steigen). Ein Mischer, ein Mikrocontroller oder ein Autopilot braucht die Umkehrung: einen Ausdruck pro Ausgang.

Die Umkehrung geschieht mit einer *Relationstabelle*: Ausgänge in den Zeilen, Eingänge in den Spalten, in jeder Zelle eine Kurve aus einer kleinen Bibliothek ($f_{|\cdot|}(x) = |x|$, $f_+(x) = x$, $f_-(x) = -x$, eine skalierte Gerade kx , die Rampe $\min(x + 1, 1)$). Eine Zeile liefert die Gleichung eines Ausganges; eine Spalte liefert das Paradigma eines Eingangs.

Tabelle 1 gibt die Tabelle von 2019 in der Notation dieses Papiers wieder, Kurven eingeschlossen. Jede Zelle wurde als Kurve im Editor des Senders eingegeben; die grafische Form macht ein falsches Vorzeichen oder eine falsche Steigung sofort sichtbar.

Ausgeschrieben hat jeder Ausgang dieselbe Gestalt,

$$O_j = \sigma \left(\sum_i k_{ji} f_{ji}(I_i) + a_j \right) + (1 - \sigma) c_j, \quad (1)$$

eine *Matrix von Kurven* f_{ji} mit Verstärkungen k_{ji} , einem Bias a_j , dem Leerlaufschalter σ und einer Parkkonstante c_j .

Die Kurven sind strukturell, die Verstärkungen nicht. Die Form einer Zelle kodiert das Paradigma: $|x|$ (der Schub steigt in beide Knüppelrichtungen), x und $-x$ (zwei Einheiten arbeiten gegeneinander), $\min(x + 1, 1)$ (der Drehpunkteingang nimmt nur Autorität weg). Diese Formen gelten für jeden Rumpf mit dieser Anordnung. Die Verstärkungen k_s , k_n , der Bias-Schub t_0 und der Parkwinkel C_0 sind Eigenschaften eines Rumpfs und seiner Aktoren.

Die Verstärkungen sind Halbkonstanten. Das Dokument von 2019 druckte $k_s = \frac{1}{2}$ und $k_n = \frac{1}{4}$; das waren Anfangswerte. Jede Verstärkung lag als globale Variable des Mixers auf einem Potentiometer des Senders und wurde in den Erprobungen nachgestellt. Sie werden deshalb in diesem Papier durchgehend als Symbole mit Anfangswert geschrieben.

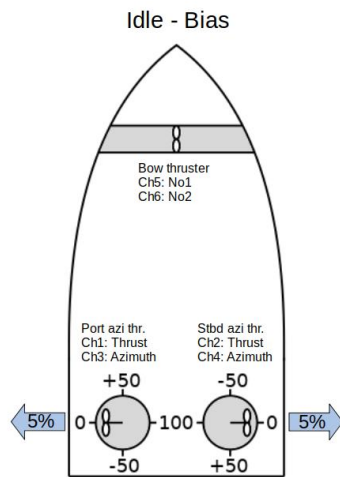
Die Gewichte je Bewegung w_{surge} , w_{sway} , w_{yaw} , 2019 auf eins gesetzt, werden gleich behandelt.

Die Moduslogik ist Arithmetik: Der Faktor σ und der Term $(1 - \sigma)c_j$ realisieren den Leerlaufschalter in einem Mischer ohne Verzweigung, und jedes Zielsystem kann den Ausdruck auswerten.

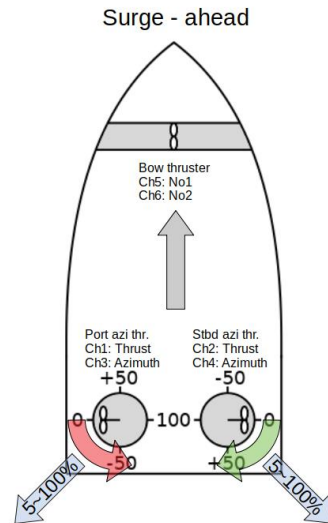
Weil die Tabelle je Ausgang geschrieben ist, ist sie vom Zielsystem unabhängig: Dieselbe Tabelle lässt sich in Mischerkurven, Mikrocontroller-Code oder Autopilot-Parameter übertragen.

2.3 Was sich ändert

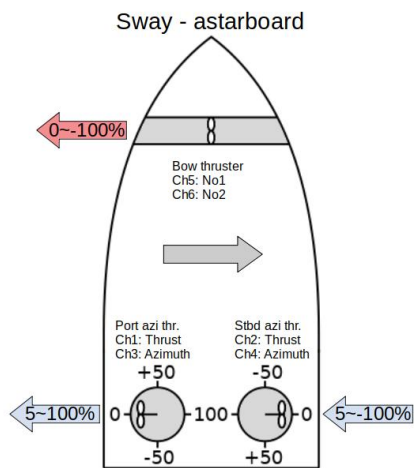
Die Relationstabelle kennt die Geometrie nicht. Ihre Verstärkungen werden durch Versuch gefunden, und kombinierte Forderungen addieren sich in den falschen Verhältnissen. Bei der Anfangsverstärkung $k_s = \frac{1}{2}$ lieferte Längsfahrt durch symmetrisches Schwenken der Azimutantriebe $2s \sin(s\pi/2)$ an Längskraft für einen Knüppelausschlag s (Abbildung 2).



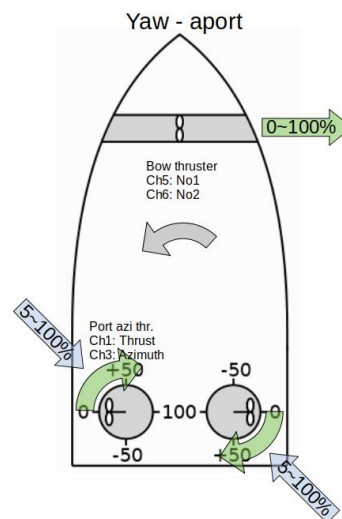
(a) Leerlauf und Bias.



(b) Fahrt voraus.



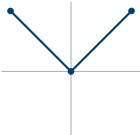
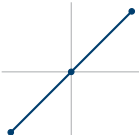
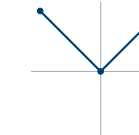
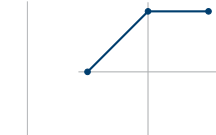
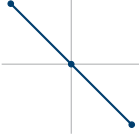
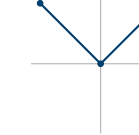
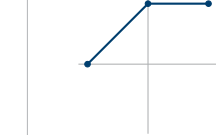
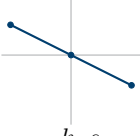
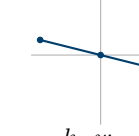
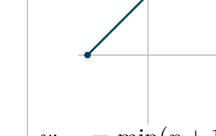
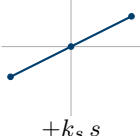
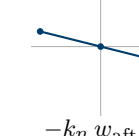
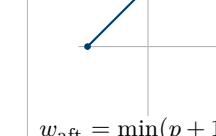
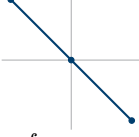
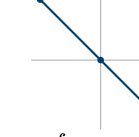
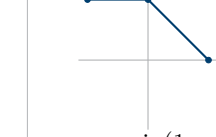
(c) Querbewegung nach Steuerbord.



(d) Gieren nach Backbord.

Abbildung 1: Die qualitativen Manövrierparadigmen von 2019. Grüne Pfeile sind Azimutdrehungen, blaue Pfeile Schubforderungen. Sie werden unverändert als Abnahmetests wiederverwendet (Abschnitt 9).

Tabelle 1: Die Relationstabelle von 2019, gezeichnet wie im Original: Jede Zelle ist eine Kurve des Eingangs am Kopf ihrer Spalte, und jeder Ausgang ist die Summe seiner Zeile, jede Kurve mit einer Halbkonstante skaliert. Eingänge: s längs, y quer, n Gieren, p Drehpunkt, $\sigma \in \{0, 1\}$ Leerlauf/Bias-Schalter, t_0 Bias-Schub, C_0 Parkwinkel. Die Verstärkungen k_s , k_n sind mit ihren Anfangswerten $\frac{1}{2}$, $\frac{1}{4}$ gezeichnet.

Ausgang	s (längs)	y (quer)	n (Gieren)	p (Drehpunkt)	Leerlauf
T_P Schub Bb.	 $f_{ · } = s $	 $f_+ = y$	 $w_{aft} f_{ · } = w_{aft} n $	 $w_{aft} = \min(p+1, 1)$	$+t_0$ alles $\times \sigma$
T_S Schub Stb.	 $f_{ · } = s $	 $f_- = -y$	 $w_{aft} f_{ · } = w_{aft} n $	 $w_{aft} = \min(p+1, 1)$	$+t_0$ alles $\times \sigma$
α_P Winkel Bb.	 $-k_s s$	ohne Einfluss	 $-k_n w_{aft} n$	 $w_{aft} = \min(p+1, 1)$	$\times \sigma$ $-C_0(1-\sigma)$
α_S Winkel Stb.	 $+k_s s$	ohne Einfluss	 $-k_n w_{aft} n$	 $w_{aft} = \min(p+1, 1)$	$\times \sigma$ $+C_0(1-\sigma)$
T_B Schub Bugstrahl	ohne Einfluss	 $f_- = -y$	 $w_{fwd} f_- = -w_{fwd} n$	 $w_{fwd} = \min(1-p, 1)$	alles $\times \sigma$

Eine Querbewegung erzeugte ein Giermoment, sofern der Hebelarm am Bug nicht genau doppelt so lang war wie der am Heck. Das Begrenzen je Kanal änderte die Richtung einer kombinierten Forderung ebenso wie ihren Betrag.

Das Schema war offene Steuerung; seine Hardware (OpenTX, Navio2, ein Arduino als i-Bus-Decoder) ist veraltet; die Sicherheit bestand aus einem Schalter. Der Sender war wegen einer einzigen Fähigkeit gewählt worden, arithmetisches Mischen, und die muss nicht mehr in der Fernsteuerung liegen (Abschnitt 5.4). Das Fazit von 2019 verlangte geschlossene Regelkreise, dynamische Positionierung und ein verteilbares Produkt; nichts davon kann das Mischen im Sender leisten.

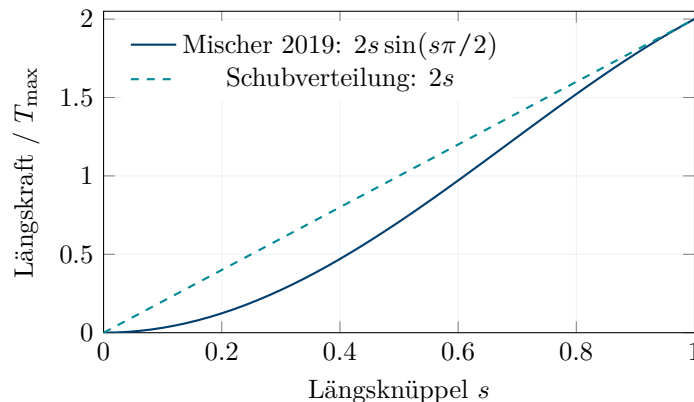


Abbildung 2: Eine messbare Folge des Prozentmischens: Längskraft, die der Längsmix von 2019 liefert, verglichen mit einer Schubverteilung, die beide Antriebe entlang der Forderung ausrichtet.

Die Änderung beschränkt sich auf Schritt vier. Eine *Effektortabelle* verzeichnet Position und Leistungsfähigkeit jedes Antriebs, die *Konfigurationsmatrix* $\mathbf{B}$ folgt aus der Tabelle, und eine *Schubverteilung auf Kraftebene* berechnet jeden Ausgang als seine Zeile von $\mathbf{B}_W^\dagger$, angewandt auf die Forderung τ_d .

Der strukturelle Gehalt der Kurven wandert in die Geometrie: Das $|x|$ der Schubzeilen und das $\pm k_s x$ der Winkelzeilen sind das, was $T = \|\mathbf{F}\|$ und $\alpha = \text{atan2}(F_y, F_x)$ liefern, sobald die Positionen der Antriebe bekannt sind.

Die Halbkonstanten zerfallen in zwei Gruppen (Tabelle 2). Die, die für Geometrie standen, entfallen, da die Matrix sie berechnet. Die, die eine Präferenz oder eine Aktoreigenschaft ausdrücken, bleiben als Parameter, editierbar von der Bodenstation aus. Die übrigen Schritte behalten ihre Rolle mit besseren Werkzeugen (Tabelle 3).

Die Steuerungssoftware muss heute Einrumpfschiffe mit Verstellpropellern, Rudern und Querstrahlrudern abdecken; Einrumpfschiffe mit Azimutantrieben; Katamarane mit verteilten Antrieben; und Fahrzeuge mit schwenkbaren Wasserstrahlantrieben oder Voith-Schneider-Propellern. Ohne Mischen in der Fernsteuerung bedienen eine Bedienschnittstelle, ein Autopilot-Stack und eine Verifikationskette alle davon.

Eine Relationstabelle lässt sich für jeden dieser Rümpfe weiterhin aufstellen und ist eine brauchbare Kontrolle einer Schubverteilung auf dem Papier; aber jeder Rumpf bräuchte seine eigenen Verstärkungen, während die Schubverteilung nur die Positionen braucht.

Die Folgen: Wer das Azimutmodell fahren kann, fährt das CPP/Ruder-Modell und den Katamaran ohne Umschulung, weil die Knüppel dieselben Kräfte kommandieren; ein neuer Rumpf ist eine neue Parameterdatei, kein neuer Code; und jeder ArduPilot-Modus, der Längs-, Quer- und Gierforderungen erzeugt, läuft auf jedem Rumpf.

Der Gleichungssatz von 2019 ist mit seinen Korrekturen in Abschnitt A abgeschrieben.

Tabelle 2: Wohin die Halbkonstanten von 2019 gehen.

Regler 2019	Anfangswert	2026
k_s , Azimutwinkel je Einheit Längsforderung	$\frac{1}{2}$	Von der Schubverteilung aus der Geometrie berechnet; keine Einstellung mehr.
k_n , Azimutwinkel je Einheit Gierforderung	$\frac{1}{4}$	Aus der Geometrie berechnet (a , b in Abbildung 3); keine Einstellung mehr.
w_{aft} , w_{fwd} , Drehpunktgewichte	$\min(p+1, 1)$, $\min(1-p, 1)$	Der Drehpunkteingang p wird auf eine Querkraftforderung abgebildet (Abschnitt 4.6); die Geometrie übernimmt die Aufteilung.
w_{surge} , w_{sway} , w_{yaw} , Bewegungsgewichte	1	MFR_KX, MFR_KY, MFR_KN: Kraft und Moment bei vollem Knüppelausschlag.
t_0 , Bias-Schub	0.05	MFR_BIAS, eine Nullraumkomponente (Abschnitt 4.2); bei einem Elektromodell standardmäßig null, auf Wunsch weiterhin ein Drehregler am Sender.
C_0 , Parkwinkel	$\frac{1}{2}$ (achtern)	MFx_ANG, Sektormitte und Ruhestellung jedes schwenkbaren Effektors.
Kurvensteigung und Expo	Mischerkurven	MFR_EXPO Schubformung, aus dem Pfahzug angepasst (Abschnitt 9.7).
Kanalbegrenzung	$\pm 100\%$	Sättigungsstrategie MFR_PRI0: gleichmäßige Skalierung der Forderung oder Giervorrang.

Tabelle 3: Die sechs Schritte, damals und heute.

Schritt	2019	2026
1. Eingänge	Knüppel und Schalter eines Senders.	Bedienerforderung τ_d in Längs, Quer, Gieren plus Drehpunkt und Modus, aus beliebiger Quelle: Sender, Joystick der Bodenstation, Regler des Autopiloten, Mission. Für jeden Rumpf identisch.
2. Ausgänge	Sechs Senderkanäle, nach den Antrieben benannt.	Effektortabelle: Typ, Position, Sektor, Schubgrenzen, Schwenkrate, Ausgangsplatz. Rumpfspezifisch, Daten statt Code.
3. Paradigmen	Prosa und Skizzen je Manöver.	Dieselben Skizzen, jetzt als Abnahmetests formalisiert: reine Querbewegung muss $N \approx 0$ liefern, und so weiter.
4. Gleichungen	Eine arithmetische Funktion je Kanal.	Konfigurationsmatrix $\mathbf{B}$ aus der Tabelle; Schubverteilung $\mathbf{u} = \mathbf{B}_W^\dagger \boldsymbol{\tau}$ mit Bias, Sättigung und Behandlung schwenkbarer Effektoren.
5. Programmierung	OpenTX-Mischer und -Kurven.	Ein Frame-Typ der Motorbibliothek von ArduPilot (Abschnitt 6.5), konfiguriert über Parameter, die von der Bodenstation aus editierbar sind, wie die alten Senderregler.
6. Verifikation	Sendersimulator, per Augenschein.	ArduPilot SITL mit einem 3-DOF-Physikmodell, automatisierte Tests aus Schritt 3, danach Identifikation am Prüfstand und Erprobung auf dem Wasser.

3 Schiffs- und Effektormodelle

3.1 Koordinatensysteme und Notation

Das körperfeste Koordinatensystem folgt der SNAME-Konvention: x voraus, y nach Steuerbord, z nach unten, Winkel positiv im Uhrzeigersinn von oben gesehen. Die Bewegung in der horizontalen Ebene wird durch den Geschwindigkeitsvektor $\boldsymbol{\nu} = [u, v, r]^\top$ (Längs-, Quergeschwindigkeit, Gierrate) beschrieben und über den verallgemeinerten Kraftvektor

$$\boldsymbol{\tau} = [X, Y, N]^\top, \quad (2)$$

gesteuert, die gesamte Längskraft, Querkraft und das Giermoment um den Schwerpunkt (CG). Ein Effektor i an der Position $\mathbf{r}_i = [l_{x,i}, l_{y,i}]^\top$ relativ zum Schwerpunkt, der eine horizontale Kraft $\mathbf{F}_i = [F_{x,i}, F_{y,i}]^\top$ erzeugt, trägt bei

$$\boldsymbol{\tau}_i = \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ l_{x,i}F_{y,i} - l_{y,i}F_{x,i} \end{bmatrix}. \quad (3)$$

Der Rumpf reagiert auf die Gesamtkraft nach dem 3-DOF-Manövriermodell [2],

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \quad (4)$$

mit $\mathbf{M}$ der Matrix aus Masse und hydrodynamischer Masse, $\mathbf{C}(\boldsymbol{\nu})$ den Coriolis- und Zentripetaltermen und $\mathbf{D}(\boldsymbol{\nu})$ der linearen plus quadratischen Dämpfung.

Die Schubverteilung verwendet dieses Modell nicht; die Simulation (Abschnitt 9), die Störgrößenschicht (Abschnitt 8) und der Identifikator (Abschnitt 8.4) tun es, alle in dieser Form.

Abbildung 3 zeigt die Geometrie des Schiffs von 2019. Konvention: Das Dokument von 2019 maß den Azimutwinkel als Strahlrichtung ab der Außenbordstellung; hier ist α_i die Richtung der Kraft auf den Rumpf, vom Bug aus gemessen, und $\alpha = 0$ (Strahl nach achtern, Schub voraus) ist die Parkstellung.

3.2 Effektortypen

Fünf Effektortypen decken die betrachteten Rümpfe ab (Tabelle 4). Jeder ist durch die Zahl der *erweiterten* Stellvariablen charakterisiert, die er beiträgt, und durch deren Beschränkungen.

Feste Antriebe tragen eine Spalte zur Konfigurationsmatrix bei: $\mathbf{b}_i = [\cos \gamma_i, \sin \gamma_i, l_{x,i} \sin \gamma_i - l_{y,i} \cos \gamma_i]^\top$. Ein Bugstrahlruder hat $\gamma = 90^\circ$; ein fester Heckpropeller hat $\gamma = 0$.

Azimutantriebe sind der Grund für die Darstellung als *erweiterter Schub* [4, 2]. Die physikalischen Variablen (T, α) gehen nichtlinear in $\boldsymbol{\tau}$ ein; die kartesischen Komponenten (F_x, F_y) gehen linear ein. Jeder Antrieb trägt deshalb zwei Spalten bei, $[1, 0, -l_y]^\top$ und $[0, 1, l_x]^\top$, und die physikalischen Kommandos werden hinterher zurückgewonnen.

Reale Antriebe haben eine maximale Schwenkrate (ein Modellbauservo über ein Getriebe liefert 150–400°/s) und manchmal Sektoren, in denen ihr Strahl den Rumpf, einen Skeg oder den Nachbarantrieb trifft und die Kraft verloren geht. Ist der Propeller umkehrbar, gibt es für jede Kraft zwei Lösungen, (T, α) und $(-T, \alpha + 180^\circ)$.

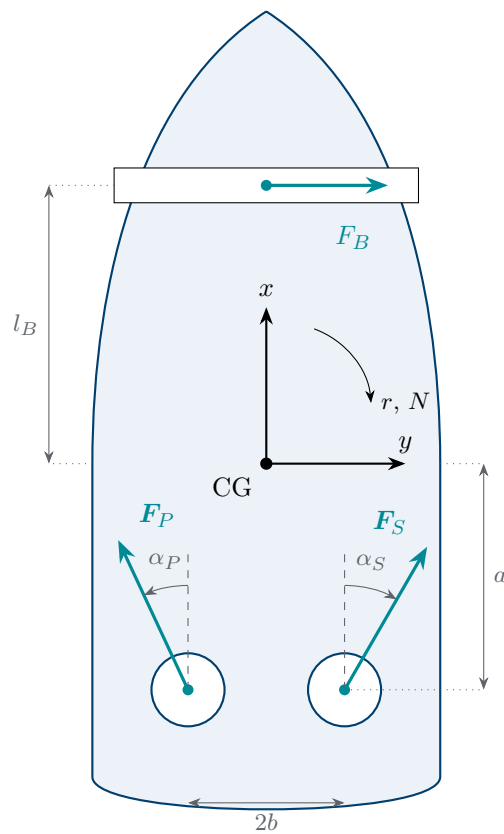


Abbildung 3: Geometrie und Vorzeichenkonventionen des Schiffs von 2019 (Beispiel A). Die Kraftwinkel α der Azimutantriebe werden vom Bug aus gemessen, positiv im Uhrzeigersinn; $\alpha = 0$ schiebt den Rumpf voraus.

Tabelle 4: Effektortypen, ihre erweiterten Variablen und Beschränkungen.

Typ	Variablen	Kraftmodell	Beschränkungen
Fester Antrieb (Querstrahlruder, Propeller mit fester Achse)	T	$\mathbf{F} = T[\cos \gamma, \sin \gamma]^\top$, γ fest	$-T_r \leq T \leq T_f$; Querstrahlruder verlieren mit der Fahrt an Wirkung; beim Katamaran ist ein Querstrahlruder, das zum anderen Rumpf hin ausströmt, unwirksam, also $T_r \approx 0$ mit γ in Richtung der Kraft, die es bei Ausströmen nach außen erzeugt, also zum anderen Rumpf hin.
Azimutantrieb	F_x, F_y	$T = \ \mathbf{F}\ $, $\alpha = \text{atan2}(F_y, F_x)$	$T \leq T_f$ voraus, $T \leq T_r$ umgekehrt; Schwenkrate $ \dot{\alpha} \leq \dot{\alpha}_{\max}$; Kraftrichtungssektor, in dem der Strahl einen Skeg oder den Nachbarantrieb träfe; ein $\pm 90^\circ$ -Servo genügt mit umkehrbaren Propellern.
Propeller + Ruder (frei oder in Kortdüse; Verstell- oder Festpropeller)	F_x, F_y	$F_x = T$, $F_y = -(k_T \max(T, 0) + k_R \min(T, 0) + k_u u) \delta$	$ \delta \leq \delta_{\max}$; die Querkraft folgt der Strömung, nicht dem Kommando: bei Rückwärtsschub stark reduziert (freies Ruder fast nichts, Kortdüse etwa ein Fünftel des Vorauswerts, Beckerruder mehr), umgekehrtes Vorzeichen bei Rückwärtsfahrt; wirksamer Sektor schmal.
Schwenkbarer Wasserstrahlantrieb	F_x, F_y	$\mathbf{F} = T_j g(q)[\cos \delta_j, \sin \delta_j]^\top$, Umkehrschaufel q , Düse δ_j	$ \delta_j \leq \delta_{j,\max} \approx 30^\circ$; Rückwärtsschub über die Schaufel, $g(-1) \approx -0.5$; die Pumpe läuft mit Schaufel in Neutralstellung leer.
Zykloidalpropeller (Voith Schneider, VSP)	F_x, F_y	$\mathbf{F}$ in jede Richtung	$\ \mathbf{F}\ \leq T_f$; sehr schnelles Vektorieren, keine Schwenkstrafe.

Propeller-Ruder-Anlagen sind der Normalfall bei AHTS- und PSV-Modellen, mit freiem Propeller oder in einer Kortdüse und mit verstellbarer oder fester Steigung; die Schubverteilung ist in allen Fällen dieselbe.

Das Ruder ist eine Tragfläche im Propellerstrahl, seine Seitenkraft ist also grob der Ausschlag mal der Summe aus einem Schubterm und einem Fahrtterm. Im Stand hat ein freies Ruder fast keine Autorität, solange der Propeller nicht voraus läuft; ein Ruder hinter einer Kortdüse behält rückwärts einen Teil seiner Autorität, und ein Beckerruder behält mehr.

In diesem Maßstab ist kein Verstellpropeller erhältlich. Ein CPP-Rumpf wird mit Festpropellern unter Drehzahlsteuerung nachgebildet; die Schubverteilung bleibt unverändert, da Steigung und Drehzahl beide Schubkommandos sind, und nur die Kennlinie Schub über Kommando (Abschnitt 9.7) unterscheidet sich.

Rückwärts verhält sich die Anlage entgegengesetzt zu einem Strahlruder. Die Querkraft eines Strahlruders folgt seinem Kommando; die eines Ruders folgt dem Wasser, das darüber strömt. Bei Propeller rückwärts trifft der Strahl das Ruder nicht mehr. Hinter einem freien Propeller kommt die Seitenkraft dann nur aus der Rückwärtsfahrt, wächst mit dem Quadrat dieser Fahrt und hat bei gleichem Ruderwinkel das entgegengesetzte Vorzeichen wie voraus; im Stand rückwärts gibt es keine.

Eine Kortdüse saugt ihren Zustrom nach vorn über das Ruder, so dass auch im Stand rückwärts etwas Autorität bleibt: als Arbeitswert etwa ein Fünftel des Vorauswerts, mit dem Vorzeichen der Rückwärtsfahrt. Ein Beckerruder, dessen Klappe den Auftrieb des Blatts grob verdoppelt, behält deutlich mehr. Der Term k_R trägt die Autorität rückwärts und der Term $k_u u|u|$ die Rückwärtsfahrt, und die Schubverteilung muss beide respektieren (Abschnitt 4.4). Beide Verstärkungen werden am Prüfstand gemessen (Abschnitt 9.7).

Nach [6] behandeln wir die Anlage als azimutähnlichen Effektor mit schmalem Sektor $|\alpha| \leq \delta_{\max}^{\text{eff}}$, im Stand in voller Breite nur für $F_x > 0$ und für $F_x < 0$ auf das geschrumpft, was k_R übrig lässt. Der Term $k_u u|u|$ erweitert diese Autorität, mit umgekehrtem Vorzeichen, sobald der Rumpf Fahrt hat, und Abschnitt 4.4 sagt, wie der Rumpf das nutzt.

Querbewegung auf so einem Rumpf ist das Traversieren mit zwei Wellen. Ein Propeller läuft voraus mit gelegtem Ruder, und der von diesem Ruder abgelenkte Strahl erzeugt die Seitenkraft am Heck. Der andere Propeller läuft rückwärts, um die Längskraft aufzuheben, und sein Ruder trägt wenig bei, nur das, was der Zustrom der Düse ihm gibt. Das Bugstrahlruder liefert die Seitenkraft vorn und gleicht das Gieren aus.

Eine lineare Schubverteilung kann das nicht von allein erzeugen, weil im Stand beide Ruder wenig oder keine Verstärkung haben, bis ein Propeller voraus läuft; Abschnitt 4.4 fügt dafür eine Regel hinzu.

Schwenkbare Wasserstrahlantriebe verhalten sich wie ein sektorbegrenzter Azimutantrieb mit asymmetrischem Betrag. Die Düse schwenkt etwa $\pm 30^\circ$, die Umkehrschaufel liefert Rückwärtsschub mit grob halbem Wirkungsgrad, und der Antrieb hält bei laufender Pumpe den Nettoschub null. Die letzte Eigenschaft ist das Gegenstück des Bias-Modus beim Wasserstrahl, und Abschnitt 4.2 gilt entsprechend.

Zykloidalpropeller haben den Voith-Schneider-Propeller (VSP) als Referenz. Sie vektorieren den Schub innerhalb eines Sekundenbruchteils in jede Richtung und sind der einfachste Fall: zwei unbeschränkte erweiterte Variablen je Einheit.

Abbildung 4 zeigt jeden Typ als Hardware im Modellmaßstab. Der Rumpf von 2019 fuhr mit einem Schottel-Azimutantrieb 40 mm von Bauer-Modelle.



(a) Azimut: Schottel-Ruderpropeller 40 mm.



(b) Azimut: Schottel 47BM-Z mit Motor.



(c) Fest: Querstromruder 33 mm mit Motor.



(d) Propeller + Ruder: Becker-ruder 70 mm.



(e) Schwenkbarer Wasserstrahl: Jet 19 BM mit Umkehrschaukel.



(f) Zykloldal: Voith-Schneider eVSP65BE.

Abbildung 4: Die fünf Effekortypen als Hardware im Modellmaßstab. Fotos © Bauer-Modelle, Eibenstock (<https://www.bauer-modelle.com>), mit freundlicher Genehmigung von Uwe Bauer wiedergegeben; Produktseiten in `figures/bauer/SOURCES.md`.

3.3 Die Konfigurationsmatrix

Stapelt man alle erweiterten Variablen in $\mathbf{u} \in \mathbb{R}^m$, ergibt sich die lineare Beziehung

$$\boldsymbol{\tau} = \mathbf{B}\mathbf{u}, \quad \mathbf{B} \in \mathbb{R}^{3 \times m}, \quad (5)$$

in der $\mathbf{B}$ Spalte für Spalte aus der Effekortabelle aufgebaut wird. Das Schiff ist *überaktuiert*, wann immer $m > 3$ und $\mathbf{B}$ vollen Zeilenrang hat, wie bei jeder Anordnung hier. Die Überaktuierung ist es, die Bias-Modi, Toleranz gegen einen ausgefallenen Antrieb und Energieoptimierung erlaubt.

3.4 Beispielkonfigurationen

Im ganzen Papier werden sieben Anordnungen verwendet. Die Geometrie ist die eines Modells von 1.6 m Länge, mit Platzhalterwerten, die durch Messungen zu ersetzen sind.

A: Schiff von 2019. Zwei Azimutantriebe bei $(-a, \mp b)$, ein Bugstrahlruder bei $(l_B, 0)$ (die zwei physischen Querstrahlruder des Rumpfs von 2019 können als eines angesteuert oder getrennt aufgeführt werden). Mit $\mathbf{u} = [F_{x,P}, F_{y,P}, F_{x,S}, F_{y,S}, F_B]^\top$,

$$\mathbf{B}_A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ b & -a & -b & -a & l_B \end{bmatrix}, \quad a = 0.65, b = 0.15, l_B = 0.60. \quad (6)$$

B: Einrumpf, zwei Verstellpropeller mit Rudern, Bugstrahlruder. Zwei Propeller-Ruder-Anlagen bei $(-a, \mp b)$ und ein Querstrahlruder. Gleiche Gestalt wie $\mathbf{B}_A$, aber die F_y -Spalten der Anlagen tragen die Sektor- und Vorzeichenbeschränkungen aus Tabelle 4. Reine Querbewegung im Stand kommt vom Querstrahlruder und von der Aufteilung aus Abschnitt 4.4, ein Propeller voraus mit Ruder, der andere rückwärts. Sie ist klein, und der Schubverteilung muss gesagt werden, dass Gieren Vorrang hat.

C: Katamaran, zwei Azimutantriebe, ein Querstrahlruder je Rumpf. Zwei Azimutantriebe bei $(-a, \mp c)$, einer unter jedem Rumpf, und ein Querstrahlruder je Rumpf bei $(l_B, \mp c)$. Der große Abstand gibt viel Gierautorität aus differentiell Schub.

Der Katamaran hat eine Regel für sich allein, und sie betrifft die Querstrahlruder. Ein Bugstrahlruder, das nach innen, zum anderen Rumpf hin, ausströmt, ist sehr unwirksam, weil seine Strömung gegen diesen Rumpf arbeitet. Jedes Querstrahlruder ist deshalb nur nützlich, wenn es nach außen ausströmt, was es in der Praxis zu einem Einrichtungsantrieb macht: Das Backbord-Querstrahlruder schiebt den Bug nach Steuerbord, das Steuerbord-Querstrahlruder nach Backbord.

Wir tragen jedes als festen Antrieb ein, dessen Kraftrichtung nach innen zeigt, die Richtung, in die es schiebt, wenn es nach außen ausströmt, mit einer Rückwärtsgrenze von null oder nahe null. Für eine gegebene Querrichtung arbeitet nur ein Querstrahlruder, was Abbildung 6 zeigt. Die Azimutantriebe sind nicht betroffen. Mit $\mathbf{u} = [F_{x,P}, F_{y,P}, F_{x,S}, F_{y,S}, T_{B,P}, T_{B,S}]^\top$, dem Backbord-Querstrahlruder nach Steuerbord schiebend, dem Steuerbord-Querstrahlruder nach Backbord und beide $T_B \geq 0$,

$$\mathbf{B}_C = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & -1 \\ c & -a & -c & -a & l_B & -l_B \end{bmatrix}, \quad a = 0.60, c = 0.35, l_B = 0.55. \quad (7)$$

D: zwei umkehrbare schwenkbare Wasserstrahlantriebe, Bugstrahlruder. Gleiche Gestalt wie $\mathbf{B}_A$; die F_y -Spalten der Strahlantriebe sind auf $|F_y| \leq F_x \tan \delta_{j,\max}$ mit $F_x \geq 0$ voraus begrenzt, und der Rückwärtsschub ist auf $|g(-1)|T_j$ begrenzt. Das Gegenstück zum Bias ist eine Pumpendrehzahl ungleich null bei neutraler Schaufel, die die Schubverteilung als minimales T_j behandelt und nicht als Kraft.

E: zwei Zykloidalpropeller achtern, Bugstrahlruder. Gleiche Gestalt wie $\mathbf{B}_A$, mit den beiden Antriebsspalten in der Richtung unbeschränkt und $\|\mathbf{F}\| \leq T_f$. Für die Schubverteilung ist es die einfachste Anordnung: Die Schritte Sektor, Umschlagen und Schwenkrate aus Abschnitt 4.4 entfallen, und das Ausblenden löst nie aus.

F: zwei CPP achtern, einziehbarer Azimutantrieb vorn, Bugstrahlruder. Anordnung B mit einem einziehbaren Azimutantrieb nahe dem Bug. Ausgefahren ist er ein Azimut-Spaltenpaar wie jedes andere, mit einem Kraftrichtungssektor, der seinen Strahl vom Rumpf fernhält; eingezogen fallen seine Spalten weg, und der Rumpf ist wieder Anordnung B. Der Wechsel der Betriebsstellung ist das neue Element, und Abschnitt 6.6 behandelt ihn.

G: Hubinsel-Ponton, vier einziehbare Azimutantriebe, vier Spudpfähle. Vier Azimutpaare an den Ecken ergeben eine Matrix mit acht Spalten und einem großen Nullraum, das Bias-Muster ist also eine Projektion und keine Paarformel. Die Spudpfähle sind keine Effektoren: Sind sie gesetzt, halten sie den Ponton, und die Schubverteilung endet. Diese Anordnung begründet die Halteeinrichtungen und Betriebsstellungen aus Abschnitt 6.6.

4 Schubverteilung

4.1 Problemstellung

Gegeben eine Forderung τ_d , finde Effektorkommandos, die sie innerhalb der Schubgrenzen, Sektorgrenzen und Schwenkraten erzeugen, mit möglichst wenig Leistung. In den erweiterten Variablen ist die unbeschränkte Fassung ein gewichtetes Kleinste-Quadrate-Problem,

$$\min_{\mathbf{u}} \frac{1}{2} \mathbf{u}^\top \mathbf{W} \mathbf{u} \quad \text{u. d. N.} \quad \mathbf{B} \mathbf{u} = \tau_d, \quad (8)$$

dessen Lösung die gewichtete Pseudoinverse ist,

$$\mathbf{u}^* = \mathbf{W}^{-1} \mathbf{B}^\top (\mathbf{B} \mathbf{W}^{-1} \mathbf{B}^\top)^{-1} \tau_d = \mathbf{B}_W^\dagger \tau_d. \quad (9)$$

Die 3×3 -Matrix $\mathbf{B} \mathbf{W}^{-1} \mathbf{B}^\top$ lässt sich geschlossen invertieren; der Aufwand in der Hauptschleife eines Flugreglers ist vernachlässigbar. Die Diagonalgewichtung $\mathbf{W}$ drückt Präferenzen aus: Ein hohes Gewicht auf den Spalten der Querstrahlruder lässt die Schubverteilung zuerst die Azimutantriebe nutzen. Ein Wirksamkeitsfaktor auf einer Spalte von $\mathbf{B}$, mit der Fahrt gesteuert, ergibt die Auswaschung aus Abschnitt 4.5.

Die Beschränkungen (Grenzen, Sektoren, Schwenkrate) werden um die Pseudoinverse herum behandelt, in den Schritten weiter unten. Von den in [3] verglichenen Methoden ist das die einfachste: exakt ohne Sättigung, billig und vorhersagbar. Ein quadratisches Programm mit expliziten Beschränkungen [5] ist geplant, sobald der Frame existiert (Abschnitt 6.5).

4.2 Nullraum und Bias-Modus

Weil $m > 3$, hat $\mathbf{B}$ einen Nullraum der Dimension $m - 3$, und jeder Vektor $\mathbf{u}_0$ darin kann zu $\mathbf{u}^*$ addiert werden, ohne die erzeugte Kraft zu ändern:

$$\mathbf{u} = \mathbf{B}_W^\dagger \tau_d + (\mathbf{I} - \mathbf{B}_W^\dagger \mathbf{B}) \mathbf{u}_0. \quad (10)$$

Für ein symmetrisches Azimutpaar an derselben Längsposition ist der Vektor, bei dem jeder Antrieb mit gleichem Betrag β zur Mittschiffslinie hin schiebt, Strahl nach außen wie beim realen Schiff,

$$\mathbf{u}_0 = \beta [0, +1, 0, -1, 0]^\top, \quad \mathbf{B}_A \mathbf{u}_0 = \begin{bmatrix} 0 \\ \beta - \beta \\ -a\beta + a\beta \end{bmatrix} = \mathbf{0}, \quad (11)$$

der Bias-Modus von 2019: Antriebe nach außen gerichtet, Schübe heben sich auf. 2019 wurde er aus der Brückenpraxis übernommen; hier ist er ein Nullraumvektor von $\mathbf{B}$. Der Betrag β ist ein Parameter.

Dieselbe Überlegung liefert den Bias für einen Katamaran, bei dem die zwei Querstrahlruder, jedes nach außen ausströmend, ebenfalls gegeneinander laufen können. Sie zeigt auch, dass ein CPP/Ruder-Rumpf keinen kraftfreien Bias hat, nur einen Leerlauf mit minimaler Steigung oder minimaler Drehzahl.

Der Bias stammt aus dem dieselmechanischen Antrieb großer Schiffe. Ein Motor kann seine Drehzahl nicht schnell ändern, und ein Verstellpropeller muss belastet bleiben, also liefern gegeneinander schiebende Antriebe eine schnelle Änderung des Nettoschubs aus Schub, der bereits vorhanden ist.

Auf einem kleinen Elektroboot bestehen diese Zwänge nicht. Das Motordrehmoment steht in Millisekunden zur Verfügung, der Propeller läuft in einem Sekundenbruchteil hoch, und der Rumpf reagiert weit langsamer als beide; die Zeit, die ein vorbelasteter Antrieb spart, ist am Rumpf nicht messbar. Der Bias kostet Batterie, Geräusch und Schubreserve.

Der Bias bleibt deshalb als Nullraumfähigkeit mit Standardwert null erhalten. Er wird für Vorführung und Ausbildung eingeschaltet, wo sichtbar nach außen laufende Antriebe erwartet werden, oder wo ein Antriebsstrang davon profitiert, belastet zu bleiben (Abschnitt 6.7).

4.3 Sättigung und Vorrang

Eine Regel kommt vor jeder Skalierung. Ein Einrichtungseffektor, ein Katamaran-Querstrahlruder mit $T_r = 0$ oder ein Propeller, der nicht umkehren kann, kann von der Pseudoinversen nach Schub in der Richtung gefragt werden, die er nicht hat. Seine Spalte wird dann entfernt und die Schubverteilung ohne sie erneut gelöst: ein Schritt eines Aktive-Menge-Verfahrens, höchstens einer je solchem Effektor.

Das Kommando stattdessen zu begrenzen, ließe ein unausgeglichenes Moment zurück und drehte die erzeugte Kraft von der Forderung weg, derselbe Fehler wie beim Begrenzen von 2019; und eine Skalierung gegen eine Nullgrenze machte die ganze Forderung zu null.

Überschreitet ein Effektor nach der Verteilung seine Grenze, wird die Forderung skaliert, nicht die Ausgänge. Die Grenze ist die der kommandierten Richtung, T_f voraus oder T_r rückwärts. Zwei Strategien sind vorgesehen.

- *Gleichmäßig*: $\tau_d \leftarrow \tau_d / \rho$ mit $\rho = \max_i \|\mathbf{F}_i\| / T_{\text{lim},i}$. Die erzeugte Kraft behält die geforderte Richtung und wird im Betrag reduziert; das behebt den Begrenzungsfehler von 2019.
- *Giervorrang*: Zuerst die Translationsforderung (X, Y) skalieren und N so lange wie möglich erhalten. Der Verlust des Kurses ist schwerwiegender als der Verlust von Fahrt, für Bediener wie Positionsregler; das ist die übliche Wahl in DP-Systemen [7].

4.4 Schwenkbare Effektoren: Umschlagen, Sektor, Schwenkrate, Ausblenden

Die Rückrechnung von (F_x, F_y) in (T, α) für einen Azimutantrieb, ein Ruder oder einen Wasserstrahl braucht fünf Entscheidungen, in dieser Reihenfolge.

1. *Kraftsektor*. Vor allem anderen: Liegt die Krafrichtung in einem Sektor, in dem der Strahl den Rumpf träfe, wird die Kraft auf die Sektorgrenze geklemmt und nur ihre nützliche Komponente behalten, $T \leftarrow T \cos(\alpha - \alpha_{\text{clamp}})$, oder mit dem Wirksamkeitsparameter dieses Sektors skaliert, etwa dort, wo der Strahl einen Skeg oder den Nachbarantrieb träfe. Geprüft wird die Krafrichtung, nicht der Antriebswinkel, denn ein umgekehrter Propeller schickt den Strahl genauso dorthin.
2. *Umschlagen*. Für einen Antrieb mit umkehrbarem Propeller wird von (T, α) und $(-T, \alpha + 180^\circ)$ die Lösung gewählt, die dem aktuellen Winkel am nächsten liegt. Das halbiert den Schwenkweg im ungünstigsten Fall. Ein Standardservo mit $180^\circ (\pm 90^\circ)$ und ein umkehrbarer Regler decken dann jede Krafrichtung ab. Beim Rumpf von 2019 waren die Antriebe nach Schottel-Art auf 360° Drehung umgebaut; das wird optional, um den Preis eines geringeren Wirkungsgrads beim Rückwärtsschub (Abschnitt 6.7).
3. *Mechanischer Sektor*. Liegt der Antriebswinkel nach dem Umschlagen außerhalb des Servobereichs, wird er auf die nächste Grenze geklemmt und nur die Kraftkomponente entlang der geklemmten Richtung behalten. Für ein Ruder reproduziert das die begrenzte Seitenkraft je Schubeinheit, die die Anlage liefern kann.
4. *Schwenkrate*. Der Winkelschritt je Zyklus wird auf $\dot{\alpha}_{\text{max}} \Delta t$ begrenzt. Servos haben endliche Geschwindigkeit, und Schrittmotoren oder Bus-Servos lassen sich mit bekannter Rate kommandieren.
5. *Ausblenden*. Solange der Istwinkel weit ($> 45^\circ$) vom kommandierten entfernt ist, wird der Schub um einen Faktor κ reduziert, damit der Antrieb während des Schwenks nicht in die falsche Richtung schiebt, wie es ein Azimutfahrer von Hand tut.

Ohne Winkelrückmeldung (Modellbauservos) wird angenommen, dass der Istwinkel dem ratenbegrenzten Kommando folgt; Bus-Servos mit Positionsrückmeldung machen diese Annahme überflüssig (Abschnitt 6.7).

Eine Propeller-Ruder-Anlage wird anders behandelt. Die Umkehr des Propellers dreht ihren Kraftvektor nicht: F_x wechselt das Vorzeichen, und F_y folgt der Strömung über dem Ruder, $F_y = -(k_T \max(T, 0) + k_R \min(T, 0) + k_u u|u|) \delta$. Der Frame berechnet deshalb die Querverstärkung $g_y = -(k_T \max(T, 0) + k_R \min(T, 0) + k_u u|u|)$ in jedem Zyklus aus dem kommandierten Schub und der Fahrt durchs Wasser und kommandiert $\delta = F_y / g_y$ innerhalb des Ruderbereichs.

Fällt $|g_y|$ unter eine Schwelle, wird die Querspalte der Anlage auf null skaliert, und die Querbewegung wandert zu den Querstrahlrudern. Bei Rückwärtsschub mit Rückwärtsfahrt wechselt g_y das Vorzeichen, und das Ruderkommando kehrt sich von selbst um.

Die Umkehr ist für den Bediener unsichtbar. Mit dem Mischen von 2019 hätte eine Querforderung rückwärts das Ruder falsch herum gelegt.

Für ein Propellerpaar bei geringer Fahrt gilt eine weitere Regel: Die Pseudoinverse kann keine Ruderautorität erzeugen, die noch nicht existiert. Übersteigt die Querforderung das, was die Querstrahlruder liefern können, und liegt die Fahrt durchs Wasser unter einer Schwelle, teilt der Frame die Propeller vor der Verteilung auf.

Er kommandiert einen Aufteilungsschub T_s voraus auf einer Seite und den Rückwärtsschub, der ihn aufhebt, auf der anderen, mit T_s proportional zur Querforderung bis zur Vorausgrenze.

Das Ruder auf der Vorausseite hat jetzt die Verstärkung $k_T T_s$, das Ruder auf der Rückwärtsseite nur das reduzierte $k_R T_s$ aus dem Zustrom der Düse, mit umgekehrtem Vorzeichen, und das Kräftepaar $2bT_s$, das die Aufteilung erzeugt, wird als bekannter Term von der Forderung abgezogen.

Der Rest der Verteilung ist dann linear wie zuvor. Das Vorausruder übernimmt die Seitenkraft am Heck, das Bugstrahlruder die Seitenkraft vorn und den Gierausgleich, und das Ruder des rückwärts laufenden Propellers übernimmt den kleinen Anteil, den sein k_R zulässt, mit dem Vorzeichen, das der Frame kennt. Die Seite, die voraus läuft, wird so gewählt, dass das Kräftepaar den Gierausgleich unterstützt.

4.5 Fahrtabhängige Steuerung der Querstrahlruder

Der Strahl eines Querstrahlruders wird mit zunehmender Fahrt nach achtern abgelenkt. 2019 blieb das dem Bediener überlassen, der das Bugstrahlruder nicht mehr benutzte, indem er beide Knüppel nach vorn schob. Hier wird die Spalte des Querstrahlruders in $\mathbf{B}$ mit einem Wirksamkeitsfaktor skaliert, der von der Längsfahrt durchs Wasser abhängt,

$$w_B(u_r) = \max(0, 1 - |u_r|/U_t), \quad u_r = u - [\mathbf{R}(\psi)^\top \mathbf{V}_c]_x, \quad (12)$$

mit U_t der Fahrt, bei der das Querstrahlruder nichts mehr nützt, im Modellmaßstab typischerweise 0.5–1 m/s, zu messen. Die Fahrt ist die durchs Wasser, denn der Strahl wird mit der Strömung über den Rumpf abgelenkt und nicht mit der Bewegung über Grund: Ein Rumpf, der in einer Strömung die Position hält, hat sein Querstrahlruder genauso ausgewaschen.

Die Fahrt über Grund aus dem Schätzer dient so lange, bis die Strömungsschätzung $\mathbf{V}_c$ aus Abschnitt 8.3 existiert. Mit $w_B \rightarrow 0$ weist die Pseudoinverse dem Querstrahlruder keinen Schub mehr zu und verlagert Quer- und Gierforderung auf die Heckeffektoren. Auf einem CPP/Ruder-Rumpf kann derselbe Mechanismus die Ruderautorität mit der Fahrt *vergrößern*, weil $k_u u_r |u_r|$ wächst.

4.6 Drehpunkt

Einen starren Körper um einen Punkt P an der Längsposition x_p auf der Mittschiffslinie zu drehen, heißt, dass der Schwerpunkt die Quergeschwindigkeit

$$v = -x_p r, \quad (13)$$

hat, wie Abbildung 5 zeigt: Eine Drehung nach Steuerbord um den Bug ($x_p > 0$, $r > 0$) bewegt den Schwerpunkt nach Backbord. Der Drehpunkteingang hat deshalb zwei Formen.

- *Geschlossener Kreis* (ratengestützt manuell, Positionshalten): das Paar $(v_d, r_d) = (-x_p r_d, r_d)$ kommandieren und die Quer- und Gierregler Y und N erzeugen lassen. Der Drehpunkt wird dann unabhängig von der Hydrodynamik eingehalten.
- *Offener Kreis* (direkt manuell): die giererzeugende Querkraft an dem Ende aufbringen, das dem Drehpunkt gegenüberliegt.

Mit dem Drehpunktknüppel $p \in [-1, 1]$ ($+1$ = Bug) ist das $Y_d = -p(2/L) N_d$, was die Wirkungslinie der Resultierenden auf $x_f = N_d/Y_d = -L/(2p)$ legt: am Heck für $p = +1$, wo die Schubverteilung die Heckeffektoren findet, am Bug für $p = -1$, wo sie das Querstrahlruder findet, und im Unendlichen für $p \rightarrow 0$, ein reines Kräftepaar. Die Gewichte $w_{\text{aft}}, w_{\text{fwd}}$ von 2019 näherten das an, ohne die Geometrie zu kennen.

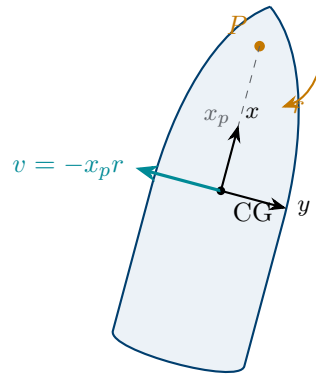


Abbildung 5: Kinematik des Drehpunkts. Eine Drehung r um P ist eine Drehung um den Schwerpunkt plus eine Quergeschwindigkeit $v = -x_p r$ am Schwerpunkt.

4.7 Erreichbare Kraftbereiche

Die Konfigurationsmatrix liefert auch die Menge der Kräfte, die der Rumpf erzeugen kann; sie zeigt vor dem Bau, ob eine Anordnung unter den vorgesehenen Bedingungen die Position halten kann. Abbildung 6 zeigt sie für die Anordnungen A und C in Längs- und Querrichtung bei Giermoment null.

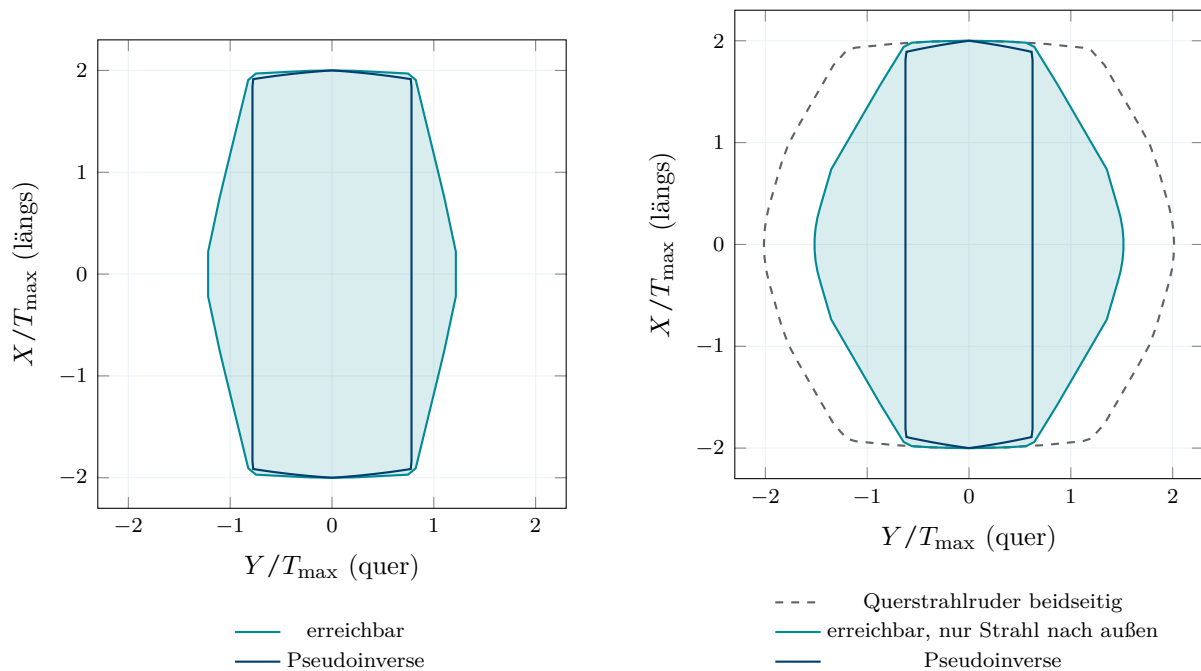
Zwei Kurven sind gezeichnet: die Menge, die die Pseudoinverse mit gleichmäßiger Skalierung erreicht, und die wirklich erreichbare Menge aus erschöpfender Abtastung der Effektorkommandos. Die wahre Menge ist konvex, ein Schnitt durch eine Summe von Scheiben und Strecken, ihr Rand ist also die konvexe Hülle der Abtastpunkte.

Der Unterschied ist der Preis der geschlossenen Schubverteilung. Er ist null entlang der Längsachse, wo beide Antriebe entlang der Forderung zeigen, und am größten, wo der Momentenausgleich gegenläufige Antriebsschübe verlangt: Bei Anordnung A erreicht die Pseudoinverse auf der Querachse $0.78 T_{\text{max}}$ gegenüber $1.22 T_{\text{max}}$ erreichbar, weil sie nie einen Antrieb voraus und den anderen rückwärts setzt, um das Moment des Querstrahlruders aufzuheben.

Ein beschränkter Löser holt den Unterschied zurück (Abschnitt 6.5). Bei beiden Rümpfen ist die Querkraft durch das Bugstrahlruder begrenzt; OSV in voller Größe tragen aus diesem Grund zwei oder drei davon.

Für den Katamaran zeigt die Abbildung auch die Wirkung der Querstrahlruderregel. Mit jedem Querstrahlruder nur beim Ausströmen nach außen wirksam, bedient ein Querstrahlruder je Querrichtung statt zwei, und die erreichbare Menge wird über die Breite um etwa ein Viertel schmaler, während Längs- und Gierrichtung unberührt bleiben.

Die Kurve der Pseudoinversen ist mit der Regel aus Abschnitt 4.3 gezeichnet: Das Querstrahlruder, das nach Schub gefragt wird, den es nicht hat, wird fallen gelassen und die Verteilung neu gelöst, was sie auf der Querachse bei $0.62 T_{\max}$ lässt, gegenüber $1.52 T_{\max}$ erreichbar.



(a) Anordnung A: zwei Azimutantriebe, ein Querstrahlruder ($0.4 T_{\max}$).

(b) Anordnung C: Katamaran, zwei Azimutantriebe, zwei Querstrahlruder (je $0.3 T_{\max}$), mit und ohne die Regel, dass ein zum anderen Rumpf hin ausströmendes Querstrahlruder unwirksam ist.

Abbildung 6: Erreichbare Längs-/Querkraft bei $N = 0$, normiert auf den Schub eines Azimutantriebs. Berechnet mit `scripts/envelope.py`.

5 Bedienschnittstelle

5.1 Eingänge, für jeden Rumpf identisch

Tabelle 5 legt die Bedienschnittstelle fest: die Belegung von 2019 mit dem Leerlaufschalter durch das Scharfschalten ersetzt. Sie ist für jede Anordnung aus Abschnitt 3.4 identisch.

5.2 Offener Kreis und ratengestützt manuell

Ein DP-Joystick bietet drei Stufen: manuell (die Knüppel kommandieren Kräfte), Auto-Heading (der Gierknüppel kommandiert eine Kursänderungsrate, der Kurs wird beim Loslassen gehalten) und Auto-Position (die Position wird gehalten, die Knüppel verschieben den Sollwert). ArduPilot Rover bietet dieselben drei.

Modus Manual. Die Knüppel gehen als τ_d durch: das Verhalten von 2019 mit korrekter Schubverteilung. Für Prüfstandsversuche und als Rückfallebene.

Modus Acro. Der Gierknüppel kommandiert eine *Gierrate*, geschlossen über das Gyroskop, der Längsknüppel eine *Fahrt*, geschlossen über die geschätzte Geschwindigkeit; bei Knüppel in Mit-

Tabelle 5: Bedienereingänge. Längsbewegung auf dem linken Knüppel wie in der Kanalliste von 2019; jeder Sender und jede Konsole aus Abschnitt 5.4 wird auf dieselben Achsen abgebildet.

Bedienelement	Forderung	Bedeutung
Linker Knüppel, vertikal	X_d (längs)	Kraft voraus / zurück, oder Fahrt in ratengestützten Modi.
Linker Knüppel, horizontal	Y_d (quer)	Kraft nach Steuerbord / Backbord, oder Quergeschwindigkeit.
Rechter Knüppel, horizontal	N_d (Gieren)	Moment nach Steuerbord / Backbord, oder Gierrate.
Rechter Knüppel, vertikal	p (Drehpunkt)	Drehzentrum: vorn = Bug, hinten = Heck, Mitte = Schwerpunkt. Federzentrierter Knüppel oder gerasteter Schieber.
Zweistufenschalter	Scharf / entschärft	Ersetzt den Leerlaufschalter von 2019. Entschärft: aller Schub null, schwenkbare Effektoren geparkt.
Dreistufenschalter	Modus	Manual (offener Kreis) / Acro (ratengestützt) / Loiter (Positionshalten) oder wie konfiguriert.
Drehregler	β (Bias)	Bias-Schub, wie im Dokument von 2019 beabsichtigt. Standardmäßig aus (Abschnitt 4.2); für Vorführungen; ohne Bedeutung auf CPP/Ruder-Rümpfen.
Tastenschalter	Hold	Sofortiges „Alles stopp“ mit Kraft null, das System bleibt scharf.

telstellung wird der Kurs gehalten. Die Querbewegung ist in der Serienfirmware offen; ein Quergeschwindigkeitsregelkreis ist eine kleine Ergänzung des Modus (Abschnitt 6.5).

Modus Loiter. Position und Kurs werden gehalten; die Knüppel verschieben den Sollwert. Das ist der DP-Modus Auto-Position (Abschnitt 7).

Der Drehpunkt verhält sich wie in Abschnitt 4.6: Gewichtung im offenen Kreis in Manual, kinematisches Kommando in Acro und Loiter.

5.3 Leerlauf, Bias, Scharfschalten

Der Leerlaufschalter von 2019 hatte zwei Funktionen, Sicherheit und Erscheinungsbild: Schub null, Antriebe nach achtern geparkt. Beides bildet sich auf das Scharfschalten von ArduPilot ab. Entschärft erzwingt der Frame Schub null und parkt jeden schwenkbaren Effektor in seiner Sektormitte, die für einen Azimutantrieb $\alpha = 0$ ist: Strahl nach achtern, bereit zum Schieben voraus.

Scharfgeschaltet wird der Bias β , falls gesetzt, als Nullraumkomponente addiert, so dass die Antriebe gegeneinander laufen; standardmäßig ist er null (Abschnitt 4.2). Das Scharfschalten wird verweigert, wenn die Vorprüfungen an GNSS, Schätzer, Batterie oder Funk scheitern; das System von 2019 hatte kein Gegenstück.

5.4 Sender- und Konsolenhardware

Der Sender von 2019 wurde gewählt, weil OpenTX Kanäle arithmetisch mischen konnte; kein anderer Modellbausender konnte das ohne externen Mikrocontroller. Diese Anforderung besteht nicht mehr. In μDP liegt das Mischen im Autopiloten, und das Bediengerät muss ungemischte Achsen und Schalter über eine Verbindung liefern, die schnell, zuverlässig und vom Begleitrechner unabhängig ist.

Die Auswahlkriterien sind Verbindungsqualität und Latenz, Kanalzahl, Telemetrie, Failsafe-Verhalten, Ergonomie der Konsole und Offenheit. Tabelle 6 bewertet die Optionen [14] danach.

OpenTX wird nicht mehr weiterentwickelt; EdgeTX, seine gepflegte Abspaltung, ist die aktuelle quelloffene Senderfirmware. Die neueren Optionen sind keine Senderfirmwares.

Tabelle 6: Optionen für das Bediengerät in μDP. Mischen ist kein Kriterium mehr.

Option	Charakter	Rolle in μDP
EdgeTX-Sender mit ExpressLRS (quelloffen, aktiv)	16 Kanäle, Latenz unter 10 ms, Reichweite im Kilometerbereich, CRSF-Telemetrie mit den Yaapu-Bildschirmen, MAVLink-Durchleitung in neueren ELRS-Versionen. Mischer und Kurven weiterhin vorhanden, falls je als Notbetrieb gewünscht.	Die sicherheitskritische manuelle Verbindung: bewährtes Failsafe, kein Rechner in der Kette. Standardwahl für die Phasen 1 bis 3.
Proprietäre Sender (FrSky Ethos, Jeti, Graupner HoTT, Futaba)	Ausgereift, geschlossen. Jeti und Graupner sind wegen ihrer Programmierbarkeit und Telemetrie die Favoriten der Schiffsmodellbauer; diese Programmierbarkeit ist jetzt überflüssig.	Akzeptabel, wenn bereits ein Sender vorhanden ist; der Autopilot nimmt S.Bus, F.Port, SRXL2 und i-Bus an. Für ein Produkt nicht empfohlen.
Joystick der Bodenstation über MAVLink (bevorzugt QGroundControl; Mission Planner, Cockpit)	Gamepad, oder industrielle Schiffsbedienelemente über eine USB-HID-Schnittstelle: Dreiachs-DP-Joysticks und Azimuthebel von Lilaas, an eine Leo-Bodnar-Platine BU0836X angeschlossen (12 Bit, acht Achsen, 32 Tasten), erscheinen der Bodenstation als gewöhnlicher Joystick, ohne Treiber. MAVLink MANUAL_CONTROL über WLAN, LTE, ELRS-MAVLink oder ein serielles Funkmodem. Quelloffen; Latenz und Failsafe hängen von der Telemetrie-Verbindung ab.	Die DP-Konsole für die Phasen 3 und 4: echte Brückenhardware, Sollwerte, Kursrad, Klicken-und-Fahren, Kamera, auf demselben Bildschirm. Braucht den RC-Sender als Rückfallebene.
Eigens gebaute Brückenkonsole (offene Hardware)	Mikrocontroller, der dieselben Lilaas-Bedienelemente, Kursrad, Drehpunktschieber und Schalter liest und CRSF an ein ELRS-Sendemodul spricht; gar keine Senderfirmware. CRSF ist dokumentiert, Bibliotheken gibt es.	Die Idee „Arduino Mega und Potentiometer“ von 2019, auf einer modernen Verbindung verwirklicht. Die Produktform der Bedienschnittstelle, wenn die Konsole ohne Rechner funktionieren muss; der Weg über BU0836X deckt sie ab, wenn ohnehin eine Bodenstation vorhanden ist.
Videozentrierte Verbindungen (OpenHD und ähnliche)	Digitales Video mit RC und MAVLink auf einer Verbindung gemultiplext; quelloffen.	Relevant, sobald Bordvideo zum Bediener muss; in diesem Stadium kein Ersatz für die RC-Verbindung.

Empfehlung: ein Sender der EdgeTX-Klasse auf ExpressLRS als manuelle Verbindung, weil sein Failsafe deterministisch und vom Begleitrechner unabhängig ist; darüber die DP-Konsole, zuerst als Lilaas-Bedienelemente auf einer BU0836X in die Bodenstation, später als eigens gebaute CRSF-Konsole, wo eine eigenständige Einheit gebraucht wird. Beide erzeugen dasselbe τ_d und dieselben Sollwerte.

Die FrSky Taranis X-Lite von 2019 läuft mit EdgeTX und nimmt ein ELRS-Modul im Lite-Schacht auf, bleibt also verwendbar.

6 Integration in ArduPilot

6.1 Was Rover bietet und was ihm fehlt

ArduPilot Rover in der Bootsklasse [8] stellt den Schätzstack bereit (EKF3 auf IMU, GNSS, Kompass und optional GNSS-Heading), die Moduslogik, die Kursraten- und Fahrtregler, einen Positionsregler, Wegpunktnavigation, Failsafes, Aufzeichnung und Telemetrie.

Seine Motorschicht unterstützt vier Frame-Familien: Lenkung und Gas (ein Ruderboot), Panzerlenkung (zwei feste Propeller), omnidirektional (drei oder vier feste Antriebe mit fester Mischmatrix) und einzelner vektorierter Schub (ein schwenkbarer Motor). Keine davon beschreibt einen Azimutantrieb, eine Propeller-Ruder-Anlage mit realistischem Ruderverhalten, einen schwenkbaren Wasserstrahl oder eine beliebige Geometrie. Die Verteilungsschicht aus Abschnitt 4 füllt diese Lücke.

6.2 Architektur

Abbildung 7 zeigt den Stack. Alles oberhalb der Verteilungsschicht ist Serien-ArduPilot; alles darunter sind die Effektortabelle und die Aktoren. Bediener, Bodenstation und Mission sind alle Quellen desselben τ_d .

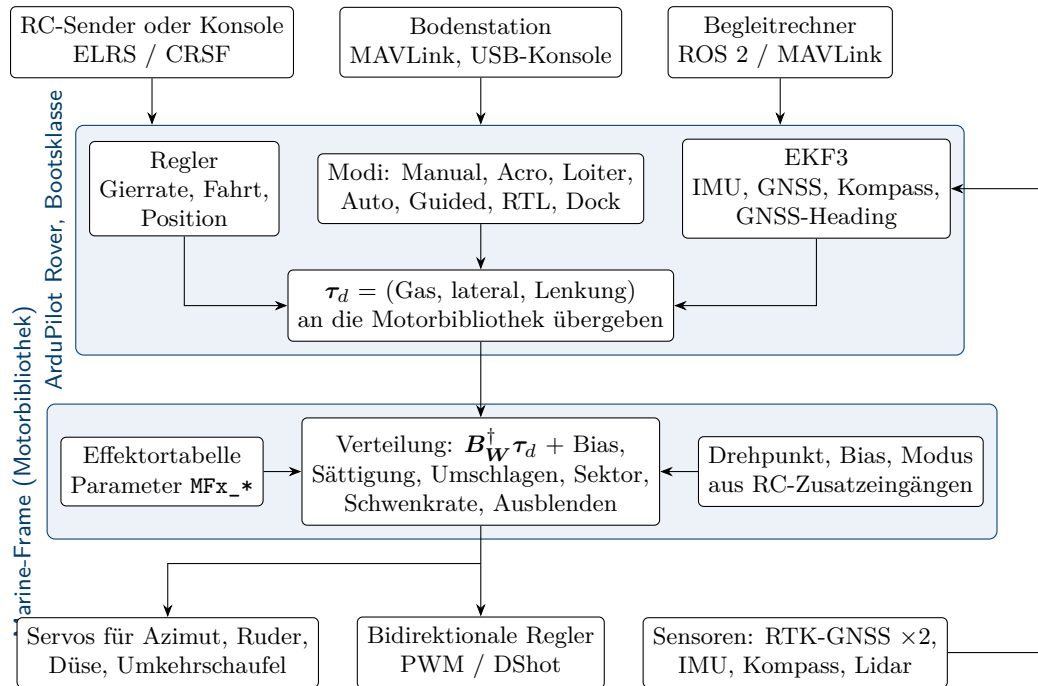


Abbildung 7: Steuerungsstack. Die Verteilungsschicht ist die einzige rumpfspezifische Komponente, und sie wird konfiguriert, nicht programmiert.

6.3 Woher die Forderung kommt

Jeder Rover-Modus endet damit, dass er der Motorbibliothek drei normierte Forderungen übergibt: Gas (X_d), Lenkung (N_d) und lateral (Y_d). In Manual sind das die Knüppel; in Acro sind es Reglerausgänge; in Loiter, Guided und Auto kommen sie von den Positions- und Lagereglern. Die Motorbibliothek verteilt sie dann an den verwendeten Frame.

Der Marine-Frame wird an dieser Stelle eingefügt: Er empfängt die drei Forderungen, verteilt sie auf seine Effektoren und schreibt die Ausgänge. Weil der Frame Querfähigkeit meldet,

kommandieren Positionsregler und Loiter Querbewegung ohne weitere Änderung am Fahrzeugcode. Nichts oberhalb der Motorbibliothek hängt vom Rumpf ab.

Firmware-Basis: ArduPilot Rover 4.6

Die Motorbibliothek, die Regler und die in diesem Abschnitt beschriebenen Integrationspunkte sind die der Versionsreihe ArduPilot Rover 4.6, für die diese Revision gilt. Das Dokument wird mit der Weiterentwicklung von ArduPilot regelmäßig überprüft, und die Integrationspunkte, die Nummerierung der Ausgangsfunktionen und die Parameternamen werden bei jeder Überprüfung und vor jedem Test am Rumpf auf einem neueren Zweig in SITL erneut bestätigt.

6.4 Warum nicht die Skript-Engine

ArduPilot enthält eine eingebettete Skript-Engine, und die Schubverteilung ließe sich dort unterbringen: kein Firmware-Build, eine Textdatei auf der SD-Karte, im Feld editierbar. Im Steuerpfad wird sie nicht verwendet, aus strukturellen Gründen.

Die Engine wurde für Erweiterungen entworfen: Telemetrie, Zusatzverhalten, Missionslogik, Sensortreiber. Sie läuft in einem eigenen Thread, in der Priorität unterhalb der Hauptregelschleife, in einem festen Heap und mit einem Befehlsbudget je Zyklus. Im einzigen Pfad zwischen Autopilot und Antrieben hat das vier Folgen.

- *Ein Skript fällt aus Sicht der Regelschleife lautlos aus.* Ein Laufzeitfehler, erschöpfter Speicher oder ein überschrittenes Befehlsbudget hält es an; eine Meldung geht an die Bodenstation, und nichts geht an den Failsafe-Zustandsautomaten. Das Fahrzeug bleibt scharf, die Modi rechnen weiter Forderungen, und nichts setzt sie um.
- *Hold kann das Schiff nicht anhalten.* Jede Sicherheitsreaktion von ArduPilot endet in einer Forderung von Kraft null. Bei einem toten Skript erreicht diese Forderung nie einen Ausgang; nur ein Ausgangs-Timeout je Kanal täte es, und das ist eine Eigenschaft davon, wie das Skript den Kanal geschrieben hat, nicht der Failsafe-Logik.
- *Sie steht außerhalb der Maschinerie, die den Rest vertrauenswürdig macht:* keine Vorprüfung der Effektortabelle vor dem Scharfschalten, keine Reaktion auf einen Fehler danach, nichts im Flugprotokoll, sofern das Skript es nicht selbst schreibt, mögliches Verhungern unter Aufzeichnungs- oder Schätzerlast, und eine fehlende oder beschädigte SD-Karte, die das Fahrzeug ganz ohne Schubverteilung scharfschaltbar lässt.
- *Sie lässt sich nicht wie Firmware begutachten:* kein Kompilierschnitt, keine statische Typisierung, keine kontinuierliche Integration gegen den Fahrzeugcode, keine Begutachtung im Upstream-Projekt.

Die Schubverteilung wird deshalb als Frame-Typ der Motorbibliothek implementiert. Die Skript-Engine bleibt für unkritische Funktionen wie Konsolenrückmeldungen verfügbar und ist nicht Teil des Regelungsentwurfs.

6.5 Der Marine-Frame

Der Frame ist ein neuer Typ von `AP_MotorsUGV` (Arbeitsname *marine*), der wie die vorhandenen Omni- und Vektor-Typen über `FRAME_TYPE` gewählt wird. Abschnitt B gibt eine Implementierungsskizze im Stil der Bibliothek.

Parameter. Eine Gruppe `MFR_*` mit Frame-Einstellungen (Verstärkungen von Knüppel zu Kraft, Bias, Ausblendfaktor, Auswaschfahrt der Querstrahlruder, Sättigungsstrategie, Schubformung, die Aufteilungsregel für Propellerpaare) und eine Gruppe `MFx_*` je Effektor (Typ, Position, Sektormitte und -breite, Schubgrenzen voraus und rückwärts, Schwenkrate, Ausgangsindex und die drei Ruderverstärkungen für Propeller-Ruder-Anlagen). Die Motorbibliothek registriert sie, sie werden also wie jeder andere Parameter gespeichert, aufgezeichnet, dokumentiert und sind von

jeder Bodenstation aus editierbar. Sie sind die Halbkonstanten aus Tabelle 2, und ein Drehregler am Sender kann weiterhin auf jede von ihnen gelegt werden. Die Namen sind Arbeitsnamen; eine Einreichung im Upstream-Projekt kann sie unter das eigene Präfix der Bibliothek stellen.

Ausgänge. Der Schub geht auf die vorhandenen Ausgangsfunktionen **MotorN**; Lenkwinkel gehen auf neue Ausgangsfunktionen **AzimuthN**, die der Servobibliothek hinzugefügt werden (für einen Rumpf mit zwei Antrieben dienen die vorhandenen Kippmotorfunktionen als Zwischenlösung). Servobereich, Trimmung und Umkehrparameter, die Schubratenbegrenzung der Motoren und die Schublinearisierung der Motorbibliothek wirken stromabwärts unverändert.

Schleife. In jedem Hauptschleifenzyklus mit der Schedulerrate des Fahrzeugs: die Spalten von **B** mit den fahrtabhängigen Wirksamkeitsfaktoren aufbauen, mit Einrichtungsregel und Sättigungsbehandlung verteilen, Bias addieren, falls aktiviert, jeden schwenkbaren Effektor über Umschlagen, Sektor, Schwenkrate und Ausblenden auflösen und schreiben. Entschärft: Schub null, schwenkbare Effektoren in ihrer Sektormitte geparkt.

Vorprüfungen. Mindestens ein Effektor; jeder aktivierte Effektor an einen Schubaussgang gebunden und, falls schwenkbar, an einen Winkelaussgang; Grenzen und Sektoren plausibel; Konfigurationsmatrix mit vollem Zeilenrang. Andernfalls verweigert das Fahrzeug das Scharfschalten, mit einer Meldung, die den betroffenen Effektor nennt.

Failsafes. Entschärfen, Hold und jedes Failsafe setzen die Ausgänge über denselben Codepfad auf null, den sie für jeden anderen Fahrzeugtyp verwenden; der Hardware-Watchdog deckt den Frame wie den Rest der Hauptschleife ab.

Aufzeichnung. Eine Frame-Protokollnachricht je Zyklus mit der Forderung, Schub und Winkel je Effektor und dem Sättigungsverhältnis, damit Erprobungen gegen den Kraftbereich aus Abschnitt 4.7 nachgespielt werden können.

Bedienereingänge. Drehpunkt und Bias-Schub werden über zwei zusätzliche RC-Eingangsfunktionen gelesen und in Modi mit offenem Kreis vor der Verteilung in die Forderung eingefaltet, in Modi mit geschlossenem Kreis in die Geschwindigkeitsforderung (Abschnitt 4.6).

Effektorausfall. Ein als ausgefallen markierter Effektor (von der Bodenstation oder über einen Zusatzschalter) verliert seine Spalten, und die Verteilung läuft mit den verbleibenden weiter.

Querregelkreis. Ein Quergeschwindigkeitsregelkreis für den Modus Acro, mit der körperfesten Geschwindigkeit aus dem Schätzer, ist eine kleine Ergänzung des Modus neben dem Frame.

Test. Die Schubverteilung ist eine gewöhnliche C++-Klasse ohne Hardwareabhängigkeit, die Abnahmetests aus Tabelle 10 sind also Host-Unit-Tests in der eigenen Testsuite der Firmware; dieselbe Binärdatei läuft dann in SITL und auf dem Flugregler. Eine beschränkte Schubverteilung als quadratisches Programm ersetzt die Pseudoinverse in Phase 4 hinter derselben Schnittstelle.

Upstream. Der Frame, seine Dokumentation und ein SITL-Modell werden im Upstream-Projekt eingereicht, damit die Funktion begutachtet, in der kontinuierlichen Integration getestet und über Firmware-Versionen hinweg mitgeführt wird.

Verworfen wurde auch, die Verteilung auf dem Begleitrechner zu rechnen und Servowerte hinunterzuschicken: Das setzt einen Allzweckrechner und eine Datenverbindung in den einzigen Pfad zu den Antrieben.

6.6 Konfigurierbare Anordnungen: Effektoren, Halteeinrichtungen, Betriebsstellungen

Der Kern der Schubverteilung ist von Zahl und Position der Effektoren unabhängig; jeder ist eine oder zwei Spalten von **B**. Einziehbare Azimutantriebe und Spudpfähle (Anordnungen F und G aus Abschnitt 3.4) fügen Anforderungen um den Kern herum hinzu, gelöst mit drei Tabellen im Frame und einem Verwalter.

Effektoren. Die Tabelle wächst auf zwölf Einträge, jeder mit bis zu drei Ausgängen (Schub, Lenkung, Hilfsfunktion wie eine Umkehrschaukel oder ein Ausfahrtrieb) und wahlweise mit Rückmeldungen: Lenkwinkel von einem seriellen Bus-Servo, ein Endschalter für ausgefahren oder eingezogen, Drehzahl aus der Reglertelemetrie.

Ab etwa acht Einheiten wandern die Ausgänge auf DroneCAN oder einen seriellen Servobus; ArduPilot erlaubt 32 Servokanäle. Der Bias wird zur allgemeinen Nullraumprojektion eines Musters „alle Antriebe nach außen“ und setzt kein symmetrisches Paar mehr voraus.

Ausfahrbare Effektoren. Eine einziehbar Einheit ist ein Zustandsautomat: eingezogen, ausgefahren, aktiv, einziehend, ausgefallen. Solange sie nicht aktiv ist, fallen ihre Spalten weg, genau wie die eines ausgefallenen Effektors, die Schubverteilung braucht also nichts Neues. Neu ist die Mechanik: der Ausfahrtrieb, die Rückmeldung oder ein zeitgesteuerter Übergang, und die Verriegelungen. Kein Schub, solange nicht ausgefahren; kein Einziehen unter Schub; kein Ausfahren oberhalb einer festgelegten Fahrt.

Halteeinrichtungen. Ein Spudpfahl, ein Anker oder eine Festmacherleine ist kein Effektor und wird nicht als einer modelliert. Er ist eine Halteeinrichtung mit eigenem Zustandsautomaten (gehoben, senkend, gesetzt, hebend), einem Aktor und einer Rückmeldung.

Einmal gesetzt, hält er das Fahrzeug: Der Schub geht auf null oder Leerlauf, der Positionsregler tritt zurück, und das Fahrzeug ist festgemacht. Beim Ablegen läuft die Folge rückwärts, wobei die Antriebe die Position halten, bevor die Pfähle heben. Eine Hubinsel, die den Rumpf aus dem Wasser hebt, ist der Extremfall, in dem nichts Hydrodynamisches mehr gilt.

Betriebsstellungen. Ausfahrbare Einheiten und Halteeinrichtungen geben dem Fahrzeug Betriebsstellungen: Fahrt (vorderer Azimutantrieb eingezogen, Querstrahlruder ausgeblendet), Manövrieren (alles ausgefahren), festgemacht (Halteeinrichtungen gesetzt, Schub aus), aufgejackt. Ein Verwalter der Betriebsstellungen schaltet die Übergänge mit den Verriegelungen und Fahrtschwellen, angestoßen von einem Zusatzschalter, der Bodenstation oder einem Missionsbefehl. Die Vorprüfung bewertet die Konfigurationsmatrix je Betriebsstellung und warnt, wo eine Betriebsstellung unteraktuiert ist, wie die Fahrt in Querrichtung, statt das Scharfschalten zu verweigern.

Wo es hingehört. Das teilt den Frame in drei: einen Verwalter für Effektoren und Halteeinrichtungen mit den Zustandsautomaten und den Ausgängen, die reine Schubverteilung und die dünne Verbindung zur Motorbibliothek. Die Teilung hält einen Hubinsel-Ponton mit zwölf Effektoren und einen Schlepper mit dreien auf einer Codebasis. Die Tabellen werden vom Konfigurationswerkzeug aus Abschnitt 9.6 erzeugt, nicht in einen Parametereditor eingegeben.

6.7 Aktoren

Motoren und Untersetzung. Die Schubqualität einer kleinen Azimut- oder Querstrahleinheit wird mehr vom Antriebsstrang bestimmt als vom Regler. Ein Modellpropeller von 40 bis 70 mm braucht bei vollem Schub etwa 1 000 bis 3 000 U/min; ein direkt angetriebener Modellbau-Außenläufer an 3S oder 4S dreht drei- bis zehnmal schneller und arbeitet bei Manövrierschüben im schlechtesten Teil seines Bereichs: Rastmoment, schlechte Auflösung nahe null, unsicherer Anlauf rückwärts.

Der empfohlene Antriebsstrang ist ein bürstenloser Motor mit niedrigem Kv-Wert (100 bis 400 U/min je Volt, Gimbal- oder Außenläufer mit großem Durchmesser) mit einer Getriebe- oder Zahnriemenuntersetzung von 2:1 bis 4:1. Die getriebe- und riemengetriebenen Azimuteinheiten aus dem Programm von Bauer-Modelle (Abbildung 4) sind so gebaut.

Der Motor läuft dann schnell genug für saubere sensorlose Kommutierung, während der Propeller langsam dreht; die Untersetzung vervielfacht das Drehmoment, die Umkehr ist also

scharf und kleiner Schub wiederholbar; der Motor mit niedrigem Kv ist leiser und zieht bei gleicher Wellenleistung weniger Strom; der Riemen entkoppelt ihn von Stoßlasten des Propellers.

Die Schubauflösung nahe null, wo das Positionshalten arbeitet, verbessert sich um das Untersetzungsverhältnis. Ein Regler mit Sensoren oder feldorientierter Regelung ist dann eine Verfeinerung, keine Voraussetzung.

Regler. Bidirektionale Regler, wie 2019 empfohlen, bleiben die Wahl: bürstenbehaftete bidirektionale Einheiten für Bürstenmotoren, sonst bürstenlose Einheiten im 3D-Modus. Der Rückwärtsschub eines Festpropellers liegt typischerweise bei 50–70 % des Vorausschubs und wird als `MFx_TR` eingetragen.

DShot mit bidirektionaler Telemetrie liefert eine Drehzahlrückmeldung und damit eine Schub-schätzung; sonst genügt PWM mit 50–400 Hz. Mit standardmäßig ausgeschaltetem Bias (Abschnitt 4.2) muss keine Einheit über längere Zeit wenige Prozent Schub halten; wird der Bias für Vorführungen benutzt, macht der Untersetzungsstrang ihn sauber.

Azimutaktoren. In aufsteigender Leistungsfähigkeit: ein 180°-Modellbauservo über ein Getriebe, gestützt auf die Umschlaglogik aus Abschnitt 4.4; ein Mehrumdrehungs-Bus-Servo mit Positionsrückmeldung, Endlosdrehung und bekannter Schwenkrate, von einem seriellen Anschluss des Autopiloten angesteuert; oder ein kleiner Schrittmotor oder Gimbalmotor mit Magnetencoder hinter einem Mikrocontroller mit einfachem seriellen Protokoll. Mit Rückmeldung nutzt die Ausblendlogik den gemessenen Winkel, und ein blockierter Antrieb wird erkannt.

Ruder, Düsen, Umkehrschaufeln. Gewöhnliche Servos mit `SERVOx_MIN/MAX/TRIM` so kalibriert, dass das ± 1 des Frames auf den mechanischen Bereich abgebildet wird. Die Umkehrschaufeln von Wasserstrahlantrieben haben eine stark nichtlineare Kennlinie Schub über Stellung, die gemessen und als kurze Tabelle abgelegt werden sollte.

6.8 Sensoren

Positionshalten im Modellmaßstab braucht Position auf den Dezimeter und Kurs auf das Grad genau bei Fahrt null. Die Position kommt von einem RTK-GNSS-Empfänger (Klasse u-blox F9P) mit fester Basisstation oder einem NTRIP-Korrekturdienst.

Für den Kurs ist ein gut kalibrierter Kompass die Grundausstattung. Hoch montiert, fern von Reglern und Motorleitungen, mit Kalibrierung und Motorstörkompensation des Autopiloten, liefert er den Kurs bei Fahrt null auf wenige Grad.

Das reicht für Transit, erste Erprobungen und manuellen Betrieb. Es reicht nicht für das Positionshalten, das den oben genannten Kurs auf das Grad genau braucht: Der Kompass allein dient in Phase 2, und ab Loiter ist der zweite Empfänger von unten erforderlich.

Besser ist eine Moving Baseline. Ein zweiter RTK-Empfänger mit Antenne auf der Mittschiffslinie erlaubt dem Autopiloten, den Kurs aus der Basislinie zwischen den beiden Antennen abzuleiten, unbeeinflusst von Motorströmen, Ballast oder dem Eigenfeld des Rumpfs, und mit dem Kompass fusioniert, statt ihn zu ersetzen. Ein Rumpf, der Position halten und anlegen muss, erhält den zweiten Empfänger; nur für Transit ist der Kompass annehmbar.

Bei RTK-Genauigkeit dient die Lageschätzung auch als Vertikalreferenz für die Antennen. Eine Antenne sitzt auf einem Mast über dem Schwerpunkt, und jedes Rollen und Stampfen schwenkt sie seitlich um die Masthöhe mal den Sinus des Winkels. Bei einem 1.6-m-Modell mit der Antenne 0.4 m hoch bewegen 10° Rollen sie um 7 cm, ein Mehrfaches des RTK-Fehlers.

Unkorrigiert erscheint das als Positionsbewegung, und der Positionshaltekreis reagiert mit den Antrieben.

Der Schätzer korrigiert das, solange die körperfeste Position jeder Antenne in den Antennenversatzparametern eingetragen ist (die Frames `gnss_a`, `gnss_b` aus Tabelle 12). Der Fix wird dann vor der Fusion mit Roll- und Stampfwinkel von der Antenne in den Schwerpunkt verschoben, und der Moving-Baseline-Vektor wird in die Horizontale projiziert, bevor er zum Kurs wird. Der IMU-Versatz wird aus demselben Grund eingetragen.

Diese Versätze werden am fertigen Rumpf gemessen, mit dem durch Auswiegen gefundenen Schwerpunkt (Abschnitt 9.7), und in der Rumpfbeschreibungsdatei eingetragen.

Für Anlegen und Hindernisvermeidung liefern ein Nahbereichs-Lidar oder eine Bordkamera relative Messungen an den Autopiloten oder den Begleitrechner.

6.9 Stromversorgung, Verkabelung, Wassereintritt

Die Servoschiene bekommt einen eigenen Regler, getrennt von der Versorgung des Flugreglers. Die Hauptbatterie ist für alle Effektoren bei vollem Schub plus 30 % auszulegen. Der Strom wird mit dem Batteriemonitor des Autopiloten überwacht, damit ein Failsafe handeln kann.

Empfänger, Autopilot und GNSS-Antennen kommen so hoch und so weit von den Reglern weg, wie der Rumpf es erlaubt. Wasserdichte Steckverbinder zwischen abnehmbarem Deck und Rumpf sind gerechtfertigt. Das sind die „geringfügigen zusätzlichen Anforderungen“, die das Fazit von 2019 vorausgesehen hat.

6.10 Konfigurationsübersicht

Tabelle 7 sammelt die ArduPilot-Parameter, die den Stack für Anordnung A festlegen. Eine vollständige Parameterdatei je Rumpf ist ein Ergebnis von Phase 2 (Abschnitt 10).

Tabelle 7: Wesentliche ArduPilot-Konfiguration für Anordnung A. Kursive Werte sind rumpfspezifisch.

Parameter	Wert	Zweck
FRAME_CLASS	Boat	Bootsverhalten (keine Crash-Erkennung bei Drift, Loiter-Semantik).
FRAME_TYPE	Marine	Der Marine-Frame aus Abschnitt 6.5; meldet dem Positionsregler die Querfähigkeit.
SERVO1..5_FUNCTION	Motor1, Azimuth1, Motor2, Azimuth2, Motor3	Schub Bb., Winkel Bb., Schub Stb., Winkel Stb., Bugstrahlruder.
SERVOx_MIN/MAX/TRIM	<i>kalibriert</i>	Mechanischer Bereich jedes Aktors; Trimmung = Schub null / Strahl nach achtern ($\alpha = 0$).
RCMAP_*	wie Tabelle 5	Throttle = längs, Roll = Gieren, Yaw = quer.
RCx_OPTION	Arm/Disarm, Mode, Pivot, Bias	Scharfschalter, Modusschalter, Drehpunkt- und Bias-Eingänge des Marine-Frames.
MFR_KX/KY/KN	<i>6, 3, 2</i>	Kraft und Moment bei vollem Knüppelausschlag (N, N, N m).
MFR_BIAS	0	Bias-Schubanteil; standardmäßig aus.
MF1_TYPE..OUT	2, -0.65, -0.15, 0, 180, <i>5, 3, 180, 1</i>	Azimet Bb. (Typ, l_x , l_y , Parkwinkel $\alpha = 0$ mit Kraft voraus, halbe Sektorbreite, T_f , T_r , Schwenkrate, Ausgangsindex).
MF2_*	2, -0.65, +0.15, 0, 180, <i>5, 3, 180, 2</i>	Azimet Stb.
MF3_*	1, 0.60, 0, 90, 0, <i>2, 2, 0, 3</i>	Bugstrahlruder, Kraft nach Steuerbord positiv.
GPS_TYPE, GPS_TYPE2	Moving-Base-Paar	Doppeltes RTK, GNSS-Heading.
EK3_SRC1_YAW	GNSS mit Kompass als Rückfall	Kurs aus der Antennenbasislinie, Kompass bleibt.
ATC_STR_RAT_*, ATC_SPEED_*	<i>abgestimmt</i>	Gierraten- und Fahrtregelkreise für Acro und darüber.
PSC_*, LOIT_*	<i>abgestimmt</i>	Positionsregler und Radius für das Positionshalten.
FS_*	Hold	Failsafes für RC, Bodenstation, Batterie, EKF auf „Alles stopp“.

7 Autonomer Betrieb

7.1 Zuordnung der Modi

Tabelle 8 ordnet die Rover-Modi dem zu, was der Bediener eines OSV-Modells erwartet. Die Phase 3 von 2019 verlangte dynamische Positionierung, Referenzsysteme, Navigationsskripte und Bahnplanung; alle vier sind Serienfunktionen, sobald die Verteilungsschicht existiert.

Tabelle 8: Rover-Modi von der Brücke aus gesehen.

Modus	Verhalten	Entsprechung auf der Brücke
Manual	Knüppel sind Kräfte.	Einzelhebel / manueller Joystick.
Acro	Gierrate und Fahrt geregelt, Kurs beim Loslassen gehalten.	Joystick mit Auto-Heading.
Steering	Kursratenregelung, für Transit abgestimmt.	Selbststeueranlage mit Folgesteuerung.
Hold	Kraft null, scharf.	Alles stopp.
Loiter	Position und Kurs innerhalb eines Radius gehalten; Knüppel verschieben den Sollwert.	Auto-Position (DP).
Auto	Wegpunktmission mit Fahrt- und Kursregelung.	Bahnführung (Track Pilot).
RTL / SmartRTL	Rückkehr zum Startpunkt, direkt oder entlang des zurückgelegten Wegs.	Rückkehr zur sicheren Position.
Guided	Positions-, Geschwindigkeits- oder Kursziele von Bodenstation oder Begleitrechner.	DP mit externer Referenz / Fernkonsole.
Dock	Anfahrt auf ein visuelles Ziel und Anhalten davor.	Automatisches Anlegen.
Follow	Versatz zu einem anderen Fahrzeug halten.	Eskorte / Begleitfahrt.
Circle	Kreisen um einen Punkt mit gesetztem Radius und Fahrt.	Vermessungsmuster.

7.2 Positionshalten

Loiter mit kleinem Radius ist ein einfacher DP-Modus. Der Positionsregler erzeugt Längs- und Quergeschwindigkeitsforderungen, der Lageregler macht daraus und aus dem Kursfehler τ_d , und die Schubverteilung verteilt sie.

Die Leistung hängt, in dieser Reihenfolge, von der Kursquelle bei Fahrt null (Abschnitt 6.8) ab, von der Abstimmung der Gierraten- und Fahrtregelkreise und von der Genauigkeit des Schubmodells in der Schubverteilung (Totband, Formung, Rückwärtswirkungsgrad). Die Loiter-Parameter bieten kursrelatives Verhalten oder Bug zum Ziel; für ein OSV-Modell wird der zuletzt kommandierte Kurs gehalten. Die Störgrößenschicht aus Abschnitt 8 liefert die verbleibende Verbesserung.

7.3 Betrieb über die Bodenstation

QGroundControl ist die bevorzugte Bodenstation. Es bietet Klicken-und-Fahren (Guided), Missionen (Auto), Geofences, Abstimmung im Betrieb, Video und eine Joystickschicht, die die BU0836X-Konsole aus Abschnitt 5.4 als gewöhnliches USB-Gerät annimmt. Mission Planner und MAVProxy bleiben für Konfiguration und Automatisierung nützlich.

Eine Konsole nach DP-Art ist eine dünne Schicht darüber: Kurs- und Positionssollwerte als Guided-Ziele gesendet, mit dem Joystickmodus der Bodenstation als manueller Rückfallebene.

Parameterdateien und Missionsdateien zusammen sind das verteilbare Produkt, das die Modelldatei des Senders 2019 hätte sein sollen.

QGroundControl ist eine Allzweck-Fahrzeugstation mit drei Einschränkungen für maritime Arbeit: Es verwaltet in der Praxis ein Fahrzeug zu einer Zeit, kennt kein Radarbild und verwendet Luftfahrt- oder Straßenkarten statt Seekarten.

Oceanomatics entwickelt eine maritime Bodenstation, die QGroundControl mit zunehmender Reife ablösen wird. Sie koordiniert mehrere Einheiten (mehrere μDP-Rümpfe, oder einen Rumpf und seine Begleiter) auf einer Anzeige mit gemeinsamen Sollwerten und Formationslogik. Sie setzt Radarechos von den Schiffen und von Landstationen zu einem Bild zusammen.

Sie stellt außerdem elektronische Seekarten nach S-57 und der S-100-Reihe (S-101) dar, so dass Positionshalten, Anlegen und Missionen auf der Karte geplant werden, die der Großbetrieb verwenden würde. Sie spricht dieselben MAVLink- und ROS 2-Schnittstellen wie Abschnitt 9.5; auf dem Schiff ändert sich nichts, wenn sich die Station ändert.

7.4 Begleitrechner und Wahrnehmung

Ein Raspberry Pi 5 oder ein Rechner der Jetson-Klasse hängt über MAVLink am Autopiloten oder, in neueren ArduPilot-Versionen, über DDS als ROS 2-Knoten. Bordbildverarbeitung, Lidar-Verarbeitung, Erkennung von Anlegezielen und Mehrschiffkoordination wohnen dort. Er sendet ein Anlegeziel, damit der Modus Dock an einer Kaimauer anlegen kann, speist Nahrungsdaten in die Hindernisvermeidung des Autopiloten, damit Auto und Guided um Hindernisse herum steuern, und führt übergeordnete Verhalten aus (Ankerhandhabungsabläufe, Eskortmuster), die Guided-Ziele ausgeben.

Die bevorzugte Hardwareform ist ein integriertes Bündel aus einem Flugregler der Pixhawk-Klasse und einem Raspberry-Pi-Rechenmodul in einem Gehäuse (das von Holybro ist das bekannteste; Oceanomatics steht in keiner Verbindung zu Holybro): ein Platinenstapel, ein Stromeingang, interne Verbindungen, und ein Begleitrechner mit Oceanomatics-Software auf gewöhnlichem Linux neben Serien-ArduPilot mit dem Marine-Frame.

Die Bedingung ist die Abdichtung. Diese Bündel sind für Luft- und Landfahrzeuge gebaut und nicht wasserdicht; auf einem Modell, das Gischt, Kondenswasser und gelegentliches Volllaufen abbekommt, brauchen sie ein abgedichtetes Gehäuse mit verschraubten Kabeleinführungen, einer Druckausgleichsmembran und Schutzlack, was gegen den Preis des Bündels abzuwägen ist.

Wo dieser Aufwand nicht gerechtfertigt ist, liefern ein separater Flugregler und ein getrennt abgedichteter Raspberry Pi, über ein kurzes seriell oder Ethernet-Kabel verbunden, dieselben Funktionen mit billiger zu ersetzenden Teilen.

7.5 Wie weit ein Modell gehen kann

Die DP-Klassennotation [7] verlangt ab Ausrüstungsklasse 2 redundante Positions- und Kursreferenzen, definiertes Verhalten bei Antriebsausfall und einen Capability-Plot; Klasse 1 toleriert einen Einzelfehler, der zum Positionsverlust führt. Im Modellmaßstab lassen sich alle vier nachbilden.

Zwei GNSS-Empfänger liefern Kursredundanz gegenüber dem Kompass und Positionskontinuität beim Ausfall eines Empfängers, allerdings nicht die unabhängige zweite Referenz, die ein klassifiziertes Schiff braucht, denn beide teilen sich Konstellation und Korrekturverbindung. Die Schubverteilung degradiert gutmütig, wenn ein Effektor als ausgefallen markiert ist (seine Spalten verlassen $\mathbf{B}$), und Abschnitt 4.7 ist der Capability-Plot.

Eine Gleichwertigkeit mit einem klassifizierten System wird nicht behauptet. Das Ergebnis ist ein getreues Lehrmodell eines solchen, und das war die Anforderung von 2019.

8 Störgrößen und ein selbstkorrigierendes Modell

Positionshalten und Bahnhalten sind durch Wind, Strömung und Modellfehler begrenzt. Die Regler von ArduPilot nehmen konstante Störungen mit ihren Integratoren auf, was für erste Erprobungen genügt, nicht aber für ein Produkt: Der Integralzustand ist keine physikalische Größe, wird nicht zwischen Modi geteilt, wird bei jedem Moduswechsel zurückgesetzt, hinkt jeder Bö und jeder Drehung hinterher und ist beim Ausschalten verloren.

Dieser Abschnitt ergänzt drei Elemente eines DP-Systems in Originalgröße: eine Windvorssteuerung aus einem Sensor, eine Strömungsschätzung, die als Zustand des Stacks geführt wird, und ein online identifiziertes und zwischen den Einsätzen gespeichertes Schiffsmodell. Abbildung 8 zeigt ihren Platz im Regelkreis.

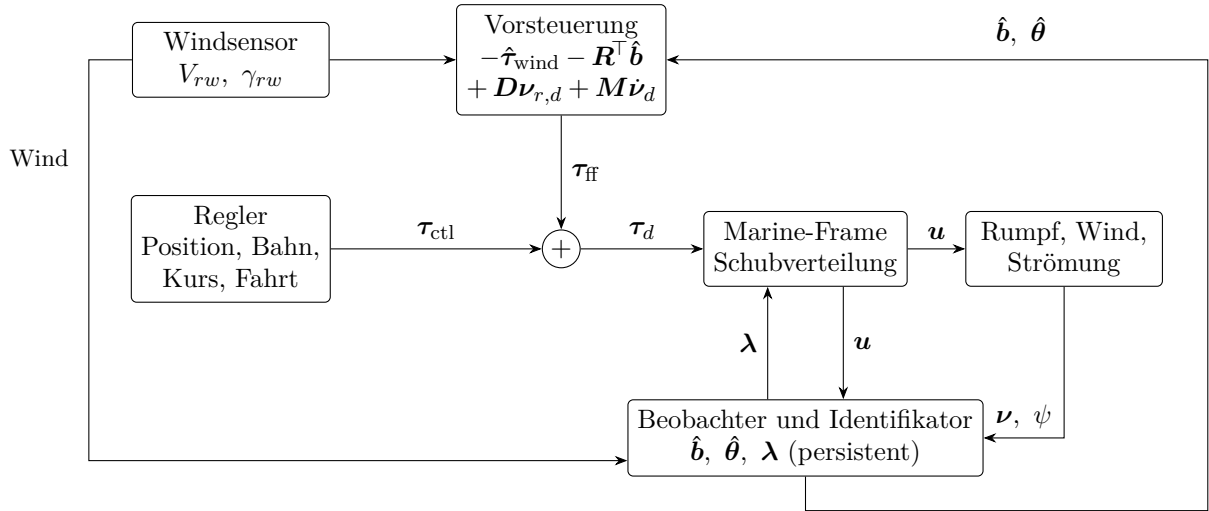


Abbildung 8: Störgrößenvorsteuerung und Modelladaption um den Marine-Frame. Die Regler sehen nur das Residuum; Beobachter und Identifikator lernen aus der verteilten Kraft $\mathbf{u}$ und der Bewegung aus dem Schätzer.

8.1 Modell mit Störgrößen

Das Manövriermodell aus Gl. (4) wird um die zwei Umweltterme ergänzt:

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \mathbf{R}(\psi)^T \mathbf{b}, \quad \boldsymbol{\nu}_r = \boldsymbol{\nu} - \mathbf{R}(\psi)^T \mathbf{V}_c, \quad (14)$$

wobei $\mathbf{V}_c$ die Strömungsgeschwindigkeit im Nord-Ost-System ist, so dass die Dämpfung auf die Geschwindigkeit relativ zum Wasser $\boldsymbol{\nu}_r$ wirkt; $\boldsymbol{\tau}_{wind}$ ist die Windlast; und $\mathbf{b}$ ist ein langsam veränderlicher Kraftbias im erdfesten System, der sammelt, was das Modell nicht kennt (Wellendrift zweiter Ordnung, Rumpfasymmetrie, ein bewachsener Propeller). Das ist das übliche DP-Streckenmodell [2].

Jeder Term wird von einem anderen Mechanismus behandelt, und alle gelangen auf dieselbe Weise in den Frame, als Vorsteuerung, die vor der Verteilung zur Forderung der Regler addiert wird:

$$\tau_d = \tau_{ctl} + \tau_{ff}, \quad \tau_{ff} = -\hat{\tau}_{wind} - \mathbf{R}(\psi)^T \hat{\mathbf{b}} + \mathbf{D}(\nu_{r,d})\nu_{r,d} + \mathbf{M}\dot{\nu}_d, \quad \nu_{r,d} = \nu_d - \mathbf{R}(\psi)^T \hat{\mathbf{V}}_c. \quad (15)$$

Hier ist $\nu_{r,d}$ die gewünschte Geschwindigkeit durchs Wasser, und $\hat{\mathbf{b}}$ ist der Teil des Bias, den die Strömungsschätzung nicht erklärt (Abschnitt 8.3), so dass die Strömung einmal zählt, in der Dämpfung, und nicht noch einmal im Bias.

Die Regler regeln dann nur das Residuum: Positionshalten und Bahnhalten gelingen ohne hohe Verstärkungen, und die Integratoren bleiben nahe null, so dass Moduswechsel ohne Übergangsvorgänge bleiben.

8.2 Windvorsteuerung für Positions- und Bahnhalten

Der Windsensor erfüllt zwei Funktionen.

Beim *Positionshalten* ist der Wind im Modellmaßstab die größte und am schnellsten wechselnde Störung. Eine Bö kommt innerhalb einer Sekunde; ein Integrator antwortet in Zehnersekunden, und bis dahin hat sich der Rumpf bewegt. Mit der aus dem Sensor berechneten und vor der Verteilung aufgehobenen Windlast sieht der Positionskreis nur den Fehler der Vorsteuerung, und die Auslenkung bei einer Bö ist durch die Reaktionszeit der Antriebe begrenzt.

Beim *Bahnhalten* (Transits in Auto und Guided) erzeugt ein Wind von der Seite Querversatz und ein Giermoment. Ohne Vorsteuerung verlässt das Schiff die Bahn, bis der Bahnfehlerintegrator eine Korrektur aufgebaut hat, und wiederholt das bei jedem Kurswechsel und jeder Winddrehung. Mit Vorsteuerung begegnet die Schubverteilung der Last unmittelbar; der Mechanismus hängt von der Fahrt ab.

Bei langsamer Fahrt wird der Kurs gehalten. Der Bug bleibt in Bahnrichtung, und die Querstrahlruder oder Azimutantriebe liefern die Querkraft, die die Bahn hält; ein Ruderrumpf kann das nicht und nutzt die Aufteilung aus Abschnitt 4.4.

Bei Marschfahrt hält das Schiff einen Kurs, der um wenige Grad vom Sollkurs abweicht, um Wind und Strömung auszugleichen, und der Widerstand des Rumpfs selbst liefert die Querkraft. Das ist ein kleiner Vorhaltewinkel, kein Krebsgang; mit Vorsteuerung wird er aus der bekannten Last gesetzt, statt vom Integrator auf jedem Schenkel gefunden.

Gegen- und Rückenwind gehen auf dieselbe Weise in den Fahrtregelkreis ein, so dass die Fahrt über den Schenkel gehalten wird.

Ein Windsensor auf dem Modell liefert scheinbare Windgeschwindigkeit und -richtung (V_{rw}, γ_{rw}) relativ zum Bug. Die Windlast ist

$$\tau_{\text{wind}} = \frac{1}{2} \rho_a V_{rw}^2 \begin{bmatrix} C_X(\gamma_{rw}) A_F \\ C_Y(\gamma_{rw}) A_L \\ C_N(\gamma_{rw}) A_L L \end{bmatrix}, \quad (16)$$

mit A_F , A_L der frontalen und seitlichen Projektionsfläche über der Wasserlinie und L der Länge. Die Koeffizientenfunktionen werden mit veröffentlichten Aufbautenkoeffizienten initialisiert. Als kurze Fourierreihe in γ_{rw} geschrieben gehen sie linear in Gl. (16) ein, so dass der Rumpfidentifikator aus Abschnitt 8.4 sie auf dem Wasser mit identifiziert.

Im Modellmaßstab muss der Sensor klein sein: ein Ultraschallanemometer von wenigen zehn Gramm, oder ein Windfahnenpotentiometer mit Schalen- oder Propelleranemometer an einem Impulsausgang. Die Platzierung ist wichtiger als das Gerät.

Der Sensor muss unabgeschattet sitzen: über den Aufbauten und frei von Mast, Schornstein und Antennen. Ein Sensor im Nachlauf eines Deckshauses misst einen Wind, dem der Rumpf nicht ausgesetzt ist, und die Vorsteuerung wirkt dann in die falsche Richtung. Der Masttopp oder eine eigene kurze Stange ist die richtige Position; der Identifikator bezieht die Windkoeffizienten auf diese Höhe.

Die Umrechnung von scheinbarem in wahren Wind, gebraucht für Aufzeichnung, Bodenstation und Transitplanung, verwendet die Geschwindigkeit aus dem Schätzer.

Die Windfahnenbibliothek von ArduPilot, aus seiner Segelbootunterstützung, liest analoge Fahnen, Impuls- und NMEA-Anemometer und berechnet den wahren Wind aus dem scheinbaren; das Marine-Modell verwendet ihre Ausgabe statt eines zweiten Treibers.

Die Vorsteuerung wird mit einem Vertrauensgewicht angewendet, das von null auf eins ansteigt, während die Koeffizienten konvergieren, so dass ein unkalibrierter Sensor den Rumpf nicht in die falsche Richtung treiben kann. Bis der Identifikator aus Abschnitt 8.4 existiert (Phase 3), ist das Gewicht ein fester konservativer Wert aus Prüfstands- und veröffentlichten Koeffizienten.

8.3 Strömung als Zustand des Stacks

Strömung lässt sich nicht wie Wind messen. Eine Logge oder Doppler-Logge taugt auf einem Rumpf dieser Größe nicht: Sie sitzt weniger als einen Meter von den Antrieben entfernt, und deren Strahl erreicht sie, sobald das Schiff manövriert, also dann, wenn die Strömung zählt. Eine Logge, die Antriebsstrahl als Strömung liest, würde der Vorsteuerung einen falschen Wert liefern. Die Strömung wird deshalb geschätzt, nicht gemessen.

Die geschätzte Größe ist der Bias $\mathbf{b}$ aus Gl. (14). Ein DP-Beobachter nach Fossen propagiert das Modell mit der verteilten Kraft, vergleicht es mit Position, Geschwindigkeit und Kurs aus dem Schätzer und korrigiert einen langsam veränderlichen erdfesten Kraftbias mit einer Zeitkonstante von Minuten.

Mit identifizierter Dämpfung, oder bis dahin der vom Prüfstand, wird der Bias über den linearen Teil von $\mathbf{D}$ in eine äquivalente Strömung umgerechnet,

$$\hat{\mathbf{V}}_c \approx \mathbf{R}(\psi) \mathbf{D}^{-1} \mathbf{R}(\psi)^\top \hat{\mathbf{b}}, \quad (17)$$

was die Bodenstation als Versetzung nach Richtung und Stärke anzeigt. Nach der Umrechnung geht nur der verbleibende Bias in die Vorsteuerung aus Gl. (15).

Eine Logge, falls eingebaut, dient nur als Gegenprobe auf dem Transit bei gleichmäßiger Fahrt und gleichmäßig laufenden Antrieben, wo $\mathbf{V}_c = \mathbf{v}_{\text{ground}} - \mathbf{R}(\psi) \boldsymbol{\nu}_r$ verlässlich ist; beim Positionshalten ist sie keine Quelle.

Die Strömung ist auf der ArduPilot-Seite ein Zustand: ein Nord-Ost-Geschwindigkeitsvektor mit Kovarianz, im Besitz der Schätzschicht und unabhängig vom Modus. Der Frame liest ihn für die Vorsteuerung, die Modi für die Planung (ein Transit gegen eine bekannte Versetzung wird mit der richtigen Fahrt und dem richtigen Kurs geplant), und Aufzeichnung und Telemetrie geben ihn aus.

Die Serienfirmware hält dieselbe Information im Integrator eines Reglers. Als Zustand kann sie angezeigt, geprüft, gespeichert und zwischen Einsätzen verglichen werden.

8.4 Ein selbstkorrigierendes Rumpf- und Effektormodell

Die Parameter aus Gl. (14) und Gl. (16) werden einmal am Prüfstand gemessen (Abschnitt 9.7) und driften danach: Die Batteriespannung sackt ab, Propeller bewachsen, Ballast verrutscht, und die Windkoeffizienten werden gar nicht gemessen. Statt sie nach jeder Erprobung von Hand nachzustellen, identifiziert das Fahrzeug sie im Betrieb aus den eigenen Daten.

Das Modell ist linear in seinen Parametern, sobald der Regressor gewählt ist. Die verteilte Kraft $\boldsymbol{\tau}$ ist bekannt, weil der Frame sie berechnet hat, und die Bewegung $\boldsymbol{\nu}$, $\dot{\boldsymbol{\nu}}$ kommt aus dem Schätzer. Mit den Unbekannten in $\boldsymbol{\theta}$ gesammelt (diagonale Masse und hydrodynamische Masse, lineare und quadratische Dämpfung, Fourierkoeffizienten des Winds),

$$\boldsymbol{\tau} = \boldsymbol{\Phi}(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}}, V_{rw}, \gamma_{rw}) \boldsymbol{\theta} + \mathbf{R}(\psi)^\top \mathbf{b}, \quad (18)$$

im Schleifentakt gelöst von einem rekursiven Kleinste-Quadrate-Schätzer mit Vergessensfaktor [9, 10].

Die Effektorwirksamkeit, das Verhältnis von geliefertem zu kommandiertem Schub, wird daneben als Faktor λ_i je Effektor identifiziert. Mit festgehaltenen Rumpfpparametern wird die

Kraft, die der Rumpf tatsächlich erhalten hat, $\Phi\hat{\theta} + R^T\hat{b}$, auf die kommandierten Effektorkräfte regressiert, $\Phi\hat{\theta} + R^T\hat{b} = B \text{diag}(u) \lambda$, wieder linear.

Die verteilte Kraft kann hier nicht verwendet werden, denn sie ist per Konstruktion gleich Bu . Ein λ_i unter seiner Schranke zeigt einen bewachsenen Propeller oder einen ausfallenden Regler an; es wird dem Bediener gemeldet und sofort in der Verteilung angewendet, damit die anderen Effektoren die Last übernehmen.

Vier Regeln beschränken den Schätzer. *Anregung*: Er aktualisiert nur, wenn der Regressor informativ ist. Erprobungen in Manual und Acro liefern das; ruhiges Positionshalten nicht, also wird die Aktualisierung über ein Anregungsmaß freigegeben, und der Vergessensfaktor bleibt nahe eins. *Schranken*: Jeder Parameter ist auf ein Intervall um seinen Prüfstandswert beschränkt, und ein Schritt darüber hinaus wird verworfen und protokolliert, nie angewendet.

Vertrauen: Die Spur der Kovarianz wird als Gütemaß gemeldet, und die modellbasierte Vorsteuerung aus Gl. (15) wird eingeblendet, während sie fällt. *Konsistenz*: Die identifizierten Parameter sind die der Simulationsmodelle aus Abschnitt 9, so dass Physikmodell und Gazebo-Welt sich beim Fahren dem realen Rumpf annähern und der Kraftbereich aus Abschnitt 4.7 aus identifizierten statt nominellen Schüben neu gezeichnet wird.

8.5 Persistenz

Zu speichern ist das identifizierte Rumpfmodell. Die Identifikation braucht einen Anregungslauf und eine Strecke Fahrt und muss nicht bei jedem Neustart des Reglers wiederholt werden.

Die identifizierten Parameter, die Wirksamkeitsfaktoren und die letzte Strömungsschätzung werden in einer Parametergruppe (Arbeitsname MDL_*) abgelegt, die beim Entschärfen in den Speicher geschrieben wird, mit Gütemaß und Zeitstempel, nach dem Mechanismus, den ArduPilot schon für gelernte Kompassoffsets verwendet.

Beim nächsten Start ist das Modell sofort verfügbar, die Vorsteuerung ab der ersten Minute aktiv, und der Schätzer verfeinert die vorhandenen Werte.

Ein persistentes Modell wird mit der Prüfung aus DP-Abnahmefahrt und FMEA validiert: Alle Positionsreferenzen werden für einige Minuten abgeschaltet, das System hält allein auf dem Modell die Position (koppelnd aus verteilten Kräften, Wind und Strömungsschätzung), und der Versatz wird gemessen, wenn die Referenzen zurückkommen.

Ein kleiner Versatz bestätigt das Modell; die Richtung eines großen benennt den falschen Term. Die Prüfung wird in der Simulation als Szenario und auf dem Wasser während der Erprobung gefahren.

Drei Sicherungen gelten: Die Bodenstation kann das Modell einfrieren (für Vorführungen oder nach einem Rumpfwechsel), auf die Prüfstandswerte zurücksetzen oder exportieren; jeder gespeicherte Satz steht auch im Flugprotokoll, so dass sich eine Verschlechterung auf die Sitzung zurückführen lässt, die sie erzeugt hat; und die Rumpfbeschreibungsdatei aus Abschnitt 9.5 wird aus dem exportierten Satz aktualisiert.

8.6 Wo es wohnt

Tabelle 9 ordnet jedes Element im Stack ein. Die Schnittstelle der Schubverteilung bleibt unverändert: Der Frame erhält eine Forderung und einen Satz Wirksamkeitsfaktoren.

9 Verifikation

9.1 Von den Paradigmen zu den Testfällen

Die Manövrierparadigmen von 2019 werden zu automatisierten Abnahmetests der Schubverteilung, gegen B ohne Physik ausgeführt (Tabelle 10). Sie erkennen Vorzeichen- und Geometriefehler in Sekunden und laufen bei jeder Änderung einer Parameterdatei erneut.

Tabelle 9: Komponenten der Störgrößen- und Modellschicht.

Element	Wo	Eingänge und Ausgänge
Windsensor	Vorhandene Windfahnenbibliothek; neue Sensorplatzierung im Transformationsbaum.	Scheinbare Windgeschwindigkeit und -richtung; wahrer Wind für Aufzeichnung, Telemetrie und Transitplanung.
Windvorsteuerung	Marine-Modellbibliothek, von den Modi vor dem Frame aufgerufen.	Koeffizienten vom Identifikator; $\hat{\tau}_{\text{wind}}$ mit Vertrauensgewicht, in die Positions-, Bahn- und Fahrtregelkreise.
DP-Beobachter	Marine-Modellbibliothek, im Schleifentakt.	Verteilte Kraft, Schätzerzustand; erdfester Bias $\hat{\mathbf{b}}$ und Kovarianz.
Strömungszustand	Schätzschicht; neue Nachricht in Protokoll und Telemetrie.	Bias und identifizierte Dämpfung (eine Logge nur als Gegenprobe auf dem Transit, da der Antriebsstrahl sie beim Manövrieren verfälscht); $\hat{\mathbf{V}}_c$ mit Kovarianz, als Versetzung angezeigt.
Identifikator	Marine-Modellbibliothek; rekursive kleinste Quadrate, anregungsgesteuert.	Bewegung, Kraft, Wind; Rumpfparameter $\hat{\boldsymbol{\theta}}$, Effektorwirksamkeit $\boldsymbol{\lambda}$, Gütemaß.
Effektorwirksamkeit	Marine-Frame.	$\boldsymbol{\lambda}$ in der Verteilung angewendet; niedrige Werte gemeldet.
Persistenz	Parametergruppe MDL_*, beim Entschärfen gespeichert; YAML-Export des Rumpfs.	Gelernter Zustand über Sitzungen hinweg; Einfrieren, Zurücksetzen, Export von der Bodenstation.

Tabelle 10: Abnahmetests der Schubverteilung, alle Anordnungen.

Forderung	Erwartet
Entschärft	Aller Schub null; schwenkbare Effektoren in der Sektormitte.
Scharf, Forderung null	$\mathbf{B}\mathbf{u} = \mathbf{0}$ innerhalb der Toleranz; nur Bias vorhanden, keiner, sofern nicht aktiviert.
Reine Längsforderung $\pm$	$Y \approx 0$, $N \approx 0$; Azimutantriebe auf 0° oder umgeschlagen; Querstrahlruder aus.
Reine Querforderung $\pm$	$X \approx 0$, $N \approx 0$; Querstrahlruder und Heckeffektoren im Verhältnis der Hebelarme.
Reines Gieren $\pm$, Drehpunkt Mitte	$X \approx 0$, $Y \approx 0$; nur Kräftepaar.
Gieren, Drehpunkt Bug / Heck	Querkraft achtern / vorn aufgebracht, $Y_d = -p(2/L)N_d$.
Kombiniert bei vollem Knüppel	Erzeugte Kraft parallel zur Forderung (gleichmäßig) oder N erhalten (Vorrang).
Fahrt durchs Wasser über U_t	Spalte des Querstrahlruders auf null skaliert; Quer und Gieren nur vom Heck.
Sprung in der Azimutforderung	Winkel schwenkt mit der konfigurierten Rate; Schub ausgeblendet, solange weit entfernt.
Effektor als ausgefallen markiert	Die verbleibenden Effektoren erzeugen die Forderung oder die nächste skalierte Fassung.
Einrichtungseffektor falsch herum gefragt	Spalte entfernt und neu gelöst; erzeugte Kraft parallel zur Forderung, $N \approx 0$; kein begrenzter Ausgang.

9.2 Software in the Loop

ArduPilot SITL führt die echte Firmware auf einer Workstation aus und nimmt ein externes Physikmodell über eine JSON-UDP-Schnittstelle an: Der Autopilot sendet die 16 Servoausgänge, das Modell liefert Position, Lage, Geschwindigkeit und IMU-Messwerte zurück. Der Frame-Code ist in SITL und auf dem Flugregler dieselbe Binärdatei.

Ein 3-DOF-Manövriermodell mit Masse und hydrodynamischer Masse, linearer und quadratischer Dämpfung, Wind- und Strömungseingängen und derselben Effekortabelle wie die Schubverteilung (Abschnitt C) genügt, um jeden Modus mit einer echten Bodenstation und einem echten oder simulierten Sender zu üben, Failsafes und Positionshalten eingeschlossen. Es ersetzt den Sendersimulator von 2019 durch einen bewegten Rumpf.

Die Gazebo-Brücke implementiert dieselbe JSON-Schnittstelle, so dass das Modell ohne Änderung an der Autopilotseite durch eine vollständige Gazebo-Welt ersetzt werden kann (Abschnitt 9.3).

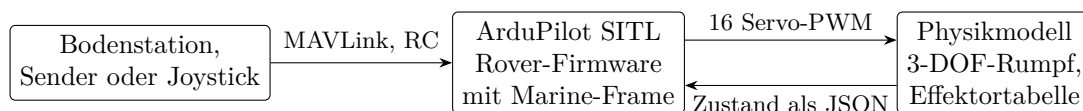


Abbildung 9: Software-in-the-Loop-Kette. Dieselbe Firmware und Parameterdatei gehen später auf den Flugregler.

9.3 Simulationsstufen

Drei Simulationsstufen werden verwendet, in aufsteigender Reihenfolge von Genauigkeit und Aufwand (Tabelle 11). Alle teilen sich eine Schnittstelle: SITL kommuniziert mit jedem Physik-Backend über dasselbe JSON-Protokoll, so dass das Python-Modell aus Abschnitt C und eine Gazebo-Welt hinter demselben Autopiloten, Frame und derselben Parameterdatei austauschbar sind.

Stufe 0 läuft ohne Anzeige in Sekunden und wird in der kontinuierlichen Integration verwendet. Stufe 2 ergänzt Wellen, Strömung, Kameras und Lidar und übt Wahrnehmung, Anlegen und die ROS 2-Seite.

Tabelle 11: Simulationsstufen.

Stufe	Was läuft	Wozu
0. Nur Verteilung	Die Verteilungsklasse des Frames als Host-Unit-Test gegen B , kein Autopilot, keine Physik.	Abnahmetests aus Tabelle 10; Sekunden; jede Parameteränderung und jeder Firmware-Build.
1. SITL + 3-DOF-Modell	ArduPilot SITL mit dem Marine-Frame, Python-JSON-Modell (Abschnitt C), Bodenstation oder geskriptetes MAVLink.	Modi, Reglerabstimmung, Failsafes, Positionshalten in ruhigem Wasser; CI ohne Anzeige.
2. SITL + Gazebo + ROS 2	ArduPilot SITL mit dem Marine-Frame, Gazebo Sim mit Hydrodynamik-, Auftriebs-, Propeller- und Gelenk-Plugins, <code>ardupilot_gazebo</code> -Brücke, <code>AP_DDS</code> zu ROS 2, <code>ros_gz</code> -Brücken für Sensoren.	Wellen, Strömung und Wind; Kamera und Lidar für Bildverarbeitung, Anlegen und Ausweichen; Software des Begleitrechners; Vorführungen.

9.4 Gazebo: Rumpf, Effektoren und Hydrodynamik

Die Gazebo-Welt wird aus derselben Effektortabelle gebaut wie die Schubverteilung. Der Rumpf ist ein Link mit Masse, Trägheit und einem Kollisionsvolumen für den Auftrieb; jeder Effektor ist ein Link, plus ein Gelenk für die schwenkbaren Typen, an (l_x, l_y) vom Schwerpunkt platziert. Vier Gazebo-Sim-Systeme liefern die Physik.

- *Hydrodynamik* bringt die hydrodynamischen Massen- und Dämpfungsterme nach Fossen auf den Rumpf-Link. Ihre Koeffizienten sind die Einträge von $\mathbf{M}_A$ und $\mathbf{D}$ des 3-DOF-Modells ($\mathbf{xDotU}$, $\mathbf{yDotV}$, $\mathbf{nDotR}$ für die hydrodynamische Masse; $\mathbf{xU}$, $\mathbf{xUabsU}$, $\mathbf{yV}$, $\mathbf{yVabsV}$, $\mathbf{nR}$, $\mathbf{nRabsR}$ für lineare und quadratische Dämpfung), so dass Python-Modell und Gazebo-Welt einmal identifiziert werden und konsistent bleiben. Dasselbe System nimmt einen Strömungsvektor an, womit Positionshalten unter Störung getestet wird.
- *Auftrieb* (abgestuft) liefert Tauchen, Rollen und Stampfen als Rückstellung aus dem Rumpfvolumen, was das 3-DOF-Modell ignoriert und was für die Eintauchtiefe der Querstrahlruder und die Kamerabewegung zählt.
- *Propeller* bildet jeden Propeller mit dem Propellergesetz $T = \rho K_T n |n| D^4$ aus Schubbeiwert, Durchmesser und Fluidichte nach, mit optionaler Asymmetrie rückwärts; eine Instanz je Azimutantrieb, Querstrahlruder oder Wellenpropeller. Ein Wasserstrahl wird durch einen Propeller mit begrenztem Düsegelenk und einem Schaufelgelenk dargestellt, das den Schub zwischen $+1$ und $g(-1)$ skaliert.
- *Gelenkpositionsregler* treibt die Gelenke von Azimut, Ruder, Düse und Schaufel mit Schwenkrate und Bereich der echten Aktoren. Der Ruderauftrieb, wenn er nicht in das Sektormodell der Schubverteilung eingefaltet ist, kommt mit dem Auftriebs-Widerstands-System auf den Ruder-Link.

Das Plugin `ardupilot_gazebo` schließt die Schleife. Es implementiert die SITL-JSON-Schnittstelle auf der Gazebo-Seite und bildet jeden der 16 Servoausgänge über einen `<control>`-Block (PWM-Bereich, Multiplikator, Offset, Kommandotyp) auf ein Gelenk- oder Propellerkommando ab. Die Zuordnung ist die letzte Spalte der Effektortabelle, so dass Gazebo-Modell und Frame-Parameter ohne Handarbeit darin übereinstimmen, welcher Ausgang was treibt.

Wellenfelder für Seegangstests stammen aus der Gazebo-Arbeit des Wettbewerbs Virtual RobotX [11] und den daraus abgeleiteten Wellen-Plugins.

9.5 ROS 2: Koordinatensysteme, Brücke und Testumgebung

Koordinatensysteme. ROS folgt REP 103 und REP 105 [12, 13]: Weltsysteme sind Ost-Nord-Oben, Körpersysteme sind Vorn-Links-Oben, Winkel sind rechtshändig um z nach oben. Maritime Regelung, der interne Zustand von ArduPilot und die Schubverteilung aus Abschnitt 4 verwenden Nord-Ost-Unten und Vorn-Rechts-Unten (SNAME). Beides wird gebraucht; die Umrechnung darf nur an einer Stelle erfolgen. Tabelle 12 legt den Baum mit einer einzigen Umrechnung fest, der festen Drehung zwischen `base_link` (FLU) und `base_link_frd` (FRD), einer Rolldrehung um 180° :

$$\mathbf{p}_{FRD} = [x, -y, -z]_{FLU}^T, \quad \mathbf{p}_{NED} = [y, x, -z]_{ENU}^T, \quad \psi_{NED} = \frac{\pi}{2} - \psi_{ENU}. \quad (19)$$

Jeder Effektor bekommt ein Koordinatensystem an seiner Position auf dem Rumpf mit z entlang der Schwenkachse. Der Azimutwinkel α ist dann der Gelenkwinkel, und `tf` zeigt die Antriebe beim Drehen.

Tabelle 12: Transformationsbaum für μDP. Statische Transformationen werden aus der Effektortabelle und der Sensorplatzierungsdatei erzeugt.

Frame	Elternteil	Konvention	Ursprung und Verwendung
map	—	ENU	EKF-Ursprung von ArduPilot (Home); gefilterte Pose aus AP_DDS; Gazebo-Welt mit sphärischen Koordinaten auf denselben Ursprung gesetzt.
odom	map	ENU	Kontinuierliches lokales System aus dem EKF; identisch mit map, sofern keine externe Odometriequelle fusioniert wird.
base_link	odom	FLU	Rumpfschwerpunkt, Körpersystem nach REP 105; was ROS-Knoten und Gazebo sehen.
base_link_frd	base_link	FRD	Rumpfschwerpunkt, Körpersystem nach SNAME; das System von τ , der Effektortabelle und der Schubverteilung.
gnss_a, gnss_b	base_link	FLU	Antennenphasenzentren; ihre Basislinie ist das Moving-Baseline-Heading, und ihr Versatz ist, was ArduPilot in seinen GPS-Positionsparametern braucht.
imu_link	base_link	FLU	IMU des Flugreglers; muss zum Parameter für die Platinaausrichtung des Autopiloten passen.
eff_i	base_link	FLU, $z =$ Schwenkachse	Effektor i bei $(l_x, -l_y, 0)$; das Kind <code>eff_i_rot</code> trägt den Gelenkwinkel.
camera_link, camera_optical	base_link	FLU / optisch	Bordkamera; optisches System nach REP 103 (z voraus, x rechts, y unten).
lidar_link	base_link	FLU	Näherungssensor für Ausweichen und Anlegen.
wind_link	base_link	FLU	Windsensor; sein Gierversatz ist, was die Windfahnenbibliothek braucht, um den scheinbaren Wind auf den Bug zu beziehen.

Brücke. ArduPilot unterstützt ROS 2 nativ über AP_DDS auf einem micro-XRCE-Agenten an einem seriellen Anschluss oder über UDP, in SITL wie auf dem Flugregler. Es veröffentlicht die gefilterte Pose und Geschwindigkeit, GNSS, IMU, Batterie, Zeit und die statischen Transformationen, die es kennt, und es abonniert Geschwindigkeitskommandos und externe Posen, mit Diensten für Scharfschalten, Moduswechsel und Vorprüfungen.

In Stufe 2 erreichen die Gazebo-Sensoren (Kamera, Lidar, wahre Pose) ROS 2 über die `ros_gz`-Brücken, und die Simulationsuhr wird veröffentlicht, so dass jeder Knoten auf simulierter Zeit läuft.

Die ROS 2-Seite besteht aus drei Knoten, von denen keiner die Schubverteilung berührt:

- ein *Frame-Publisher*, der Effektortabelle und Sensorplatzierungsdatei liest und die statischen Transformationen aus Tabelle 12 veröffentlicht, plus die vom Autopiloten oder von Gazebo zurückgelesenen Effektorgelenkwinkel;
- eine *DP-Konsole*, die Sollwerte von Bediener oder Wahrnehmung (diese Position und diesen Kurs halten, jenen Kai anlaufen, jenem Schiff folgen) über AP_DDS in Geschwindigkeits- oder Positionsziele des Modus Guided übersetzt, das ROS 2-Gegenstück zur Bodenstationskonsole aus Abschnitt 7.3;
- die *Wahrnehmungsknoten* (Kamerabildverarbeitung, Lidar-Verarbeitung), die Anlegeziele und Hindernisse für den Modus Dock und die Hindernisvermeidung des Autopiloten erzeugen.

Eine einzige Quelle der Wahrheit. Die Effektortabelle existiert an drei Stellen: in der ArduPilot-Parameterdatei (`MFx_*`), im Gazebo-SDF (Links, Gelenke, Propeller- und Steuerblöcke) und im Transformationsbaum. Alle drei werden aus einer Rumpfbeschreibung exportiert, so dass eine Änderung an Geometrie oder Ausgangsplatz sie nicht inkonsistent machen kann. Die Rumpfbeschreibung definiert einen Rumpf in μDP und entsteht im Konfigurationswerkzeug aus Abschnitt 9.6.

Testumgebung. Die Stufen 1 und 2 werden über ROS 2-Launch-Dateien mit `launch_testing` gefahren, so dass ein Szenario ein Testfall mit Bestehenskriterium ist, und über `rosbag2`-Aufzeichnungen und die eigenen Protokolle des Autopiloten zum Nachspielen.

Die Szenarien aus Tabelle 13 sind das Erprobungsprotokoll fürs Wasser, im Voraus gefahren. Jedes wird für jede Rumpfanordnung aus Abschnitt 3.4 gefahren; die Positionshaltefälle gegen den Capability-Plot aus Abschnitt 4.7, so dass die Position bei der Strömungsgeschwindigkeit verloren geht, die der Kraftbereich vorhersagt, und nicht früher.

Tabelle 13: Simulationsszenarien mit Bestehenskriterien (Werte sind Platzhalter für einen 1.6-m-Rumpf).

Szenario	Aufbau	Bestehenskriterium
Sprungantworten	Modus Manual, Einheitssprung auf jeder Achse, ruhiges Wasser.	Kraftrichtung innerhalb 5° der Forderung; kein durch Querbewegung erzeugtes Gieren über 0.02 rad/s .
Kurshalten	Modus Acro, Kurssprung 90° , dann Strömung 0.3 m/s von der Seite.	Einschwingen innerhalb 10 s , Überschwingen $< 10^\circ$, bleibender Fehler $< 2^\circ$.
Positionshalten	Loiter, Strömung von 0 auf 0.6 m/s aus vier Richtungen hochgefahren, Wellen optional.	Positionsfehler $< 0.3 \text{ m}$ bis zur Grenze des Kraftbereichs, der Kraftbereich mit dem auf der Fahrt durchs Wasser ausgeblendeten Querstrahlruder neu berechnet; gutmütiges Abdriften darüber hinaus.
Drehpunkt	Acro, Gieren bei jeder Drehpunkteinstellung.	Gemessenes Drehzentrum innerhalb 0.15 m des kommandierten Punkts.
Effektorausfall	Positionshalten, ein Effektor mitten im Lauf als ausgefallen markiert.	Kein Kursverlust; Positionsfehler durch den verkleinerten Kraftbereich begrenzt.
Failsafes	RC-Verlust, Verlust der Bodenstation, niedrige Batterie, EKF-Varianz, jeweils einzeln.	Konfigurierte Aktion innerhalb 1 s ; Schub null in Hold; Antriebe geparkt, wenn entschärft.
Mission	Auto, Viereck aus vier Wegpunkten mit Fahrtänderungen.	Bahnfehler $< 0.5 \text{ m}$; Spalte des Querstrahlruders null über U_t durchs Wasser.
Anlegen	Modus Dock, visuelles Ziel an einem Kai in Gazebo.	Berührungsgeschwindigkeit $< 0.1 \text{ m/s}$, Kurs innerhalb 5° des Kais.
Wind, Positionshalten	Loiter, Windsprung von 0 auf 6 m/s von der Seite, Vorsteuerung aus, dann an.	Maximale Positionsauslenkung mit Vorsteuerung halbiert; kein Schieben in die falsche Richtung mit unkalibrierten Koeffizienten.
Wind, Bahnhalten	Auto, Viereck aus vier Schenkeln bei stetigem Wind von 5 m/s , Vorsteuerung aus, dann an.	Bahnfehler nach jedem Schenkelwechsel mit Vorsteuerung in unter 5 s eingeschwungen; Fahrt über den Schenkel innerhalb 10% des Sollwerts.
Strömungsschätzung	Loiter, Strömungsrampe ohne Sensor.	$\hat{V}_c$ innerhalb 0.05 m/s und 15° der simulierten Strömung nach 3 min ; auf der Bodenstation gezeichnet.
Identifikation	Anregungslauf in Manual und Acro, dann Positionshalten.	Identifizierte Masse, Dämpfung und Wirksamkeit innerhalb 10% der Simulatorwerte; bewachsener Propeller ($\lambda = 0.5$) innerhalb 60 s gemeldet.
Persistenz	Neustart von SITL nach der Identifikation.	Gelernte Parameter und Strömungsschätzung wiederhergestellt; Vorsteuerung ab der ersten Minute aktiv.
Modellprüfung (Versatz)	Positionshalten, alle Positionsreferenzen für 3 min abgeschaltet, dann wieder zugeschaltet.	Versatz unter 1 m in ruhigem Wasser und unter 3 m bei 0.3 m/s Strömung; das Gütemaß des Identifikators sagt vorher, welcher.
Wechsel der Betriebsstellung	Anordnung F: Fahrt zu Manövrieren und zurück bei mehreren Geschwindigkeiten.	Vorderer Azimutantrieb fährt nur unterhalb der eingestellten Fahrt aus; kein Schub darauf, bevor er ausgefahren ist; die Querfähigkeit ändert sich mit der Betriebsstellung, wie der Vorprüfbericht es vorhersagt.
Festmachfolge	Anordnung G: Positionshalten, Spudpfähle gesetzt, Schub aus, Pfähle gehoben, Positionshalten wieder aufgenommen.	Position durch beide Übergänge innerhalb 0.3 m gehalten; kein Schub bei gesetzten Pfählen; Regler nehmen ohne Einschwingvorgang wieder auf.

9.6 Die Konfigurations-App

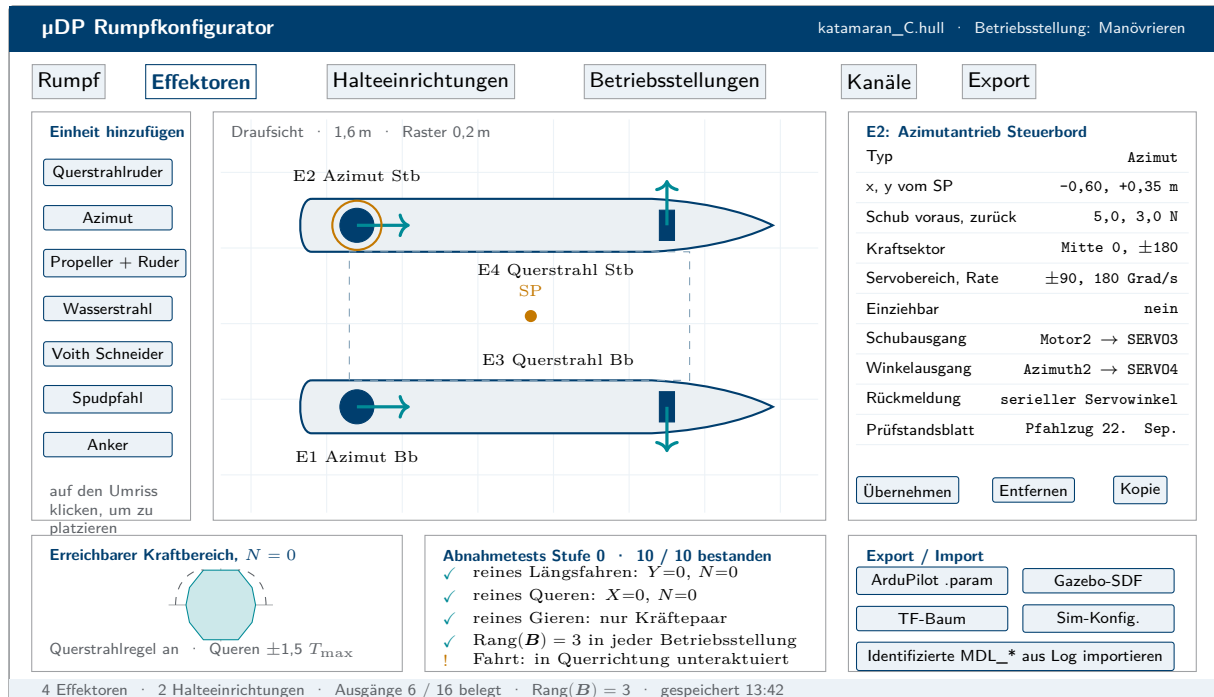


Abbildung 10: Konzept der Konfigurations-App: Ein Rumpf wird gezeichnet oder importiert, Antriebe werden auf dem Umriss platziert und mit Kanälen verdrahtet, und Kraftbereich, Abnahmetests und Exporte folgen aus derselben Beschreibung.

Ein Rumpf sollte nicht durch das Eintippen von hundert Parametern in einen Bodenstationseditor definiert werden. Das Konfigurationswerkzeug nimmt Abmessungen und Umriss des Schiffs auf, dann Antriebe und Kanäle einzeln. Abbildung 10 skizziert es: Draufsicht in der Mitte, Eigenschaften der gewählten Einheit rechts, Live-Kraftbereich und Testergebnisse darunter, Exporte am unteren Rand.

Die Rumpfbeschreibung, die es speichert, ist die einzige Quelle, aus der Parameterdatei, Simulationsmodelle und Transformationsbaum erzeugt werden.

Das Werkzeug öffnet mit einer Draufsicht. Der Benutzer gibt Länge und Breite ein, zeichnet oder importiert den Umriss (DXF oder SVG des Decksplans) und setzt den durch Auswiegen des Rumpfs gefundenen Schwerpunkt (Abschnitt 9.7).

Antriebe werden durch Klicken auf den Umriss hinzugefügt. Der Typ wird gewählt, und das Werkzeug fragt die Parameter dieses Typs ab: Sektor, Schub voraus und rückwärts aus dem Pfahlzugversuch, Schwenkrate, Ruderverstärkungen, Einziehbarkeit.

Jede Einheit wird dann verdrahtet: ein Ausgangskanal für den Schub, einer für die Lenkung, einer für die Hilfsfunktion, falls vorhanden, und ein Rückmeldeeingang, falls verfügbar. Das Werkzeug kennt Kanalzahl und Busoptionen des Autopiloten und weist kollidierende Zuordnungen zurück. Halteeinrichtungen und Betriebsstellungen werden auf dieselbe Weise hinzugefügt, mit Aktoren, Rückmeldungen und Verriegelungen.

Während Einheiten hinzukommen, zeichnet das Werkzeug den erreichbaren Kraftbereich aus Abschnitt 4.7 und lässt die Abnahmetests der Stufe 0 aus Tabelle 10 auf der aktuellen Tabelle laufen, so dass Vorzeichenfehler und unzureichende Anordnungen vor dem Export sichtbar sind. Das Queren eines Ruderrumpfs, die Querstrahlregel eines Katamarans und die Fähigkeit je Betriebsstellung werden angezeigt, während der Benutzer arbeitet.

Der Export erzeugt die ArduPilot-Parameterdatei, das Gazebo-Modell, den Transformationsbaum und die Konfiguration des Python-Physikmodells aus einer Beschreibung. Der Import liest die identifizierten Parameter und Wirksamkeitsfaktoren aus Abschnitt 8.4 aus den Aufzeichnungen des Fahrzeugs in die Rumpfdati zurück und hält sie so mit dem gefahrenen Rumpf aktuell. Die Rumpfdati liegt mit dem Projekt unter Versionskontrolle.

Die erste Fassung ist ein eigenständiges Desktop-Werkzeug für Phase 1, denn Tabelle und Exporte gebraucht werden, bevor Hardware gebaut wird. Später wird es eine Seite der Oceanomatics-Bodenstation (Abschnitt 7.3), so dass ein Rumpf in einer Anwendung konfiguriert, simuliert und gefahren wird.

9.7 Identifikation am Prüfstand

Drei Messungen am Prüfstand, etwa ein Nachmittag Arbeit, liefern der Schubverteilung ihre Zahlen.

1. *Pfahlzug* jedes Effektors voraus und rückwärts gegen eine Küchenwaage, bei fünf Reglerkommandos, um T_f , T_r und den Formungsexponenten anzupassen und das Totband des Reglers zu finden.
Für eine Propeller-Ruder-Anlage liefert derselbe Aufbau mit hart gelegtem Ruder, voraus und rückwärts, k_T und k_R .
2. *Schwenkrate* und Hysterese jedes schwenkbaren Effektors mit einer Handykamera oder der eigenen Rückmeldung des Bus-Servos.
3. *Positionen* l_x, l_y jedes Effektors relativ zum Schwerpunkt, der Schwerpunkt durch Auswiegen des fertigen Rumpfs gefunden.

9.8 Erprobung auf dem Wasser

Die Erprobung folgt der Abstufung der Modi: Manual, Acro, Loiter mit großem Radius, Loiter mit engem Radius, Auto auf einer kurzen Mission, dann die zusammengesetzten Forderungen, die 2019 ungeprüft blieben. Der Autopilot zeichnet jede Erprobung auf; das Protokoll lässt sich in der Bodenstation nachspielen und liefert die Verläufe von Schub, Winkel, Position und Kurs, mit denen W , die Schubformung und die Reglerverstärkungen verfeinert werden.

10 Fahrplan

Phase 0, dieses Dokument. Methodik, Modelle, Implementierungsskizze, Abnahmetests. Mit dieser Revision abgeschlossen.

Phase 1, Software. Den Marine-Frame in der Motorbibliothek auf einer Abspaltung von Rover implementieren (Abschnitt 6.5); Host-Unit-Tests für die Abnahmetests der Anordnungen A bis E; SITL mit dem Physikmodell; den Quergeschwindigkeitsregelkreis ergänzen; die erste Fassung der Konfigurations-App mit ihren Exporten bauen (Abschnitt 9.6). Ergebnis: Firmware-Zweig, Tests, Physikmodell, die App, eine Rumpfbeschreibung je Anordnung.

Phase 2, Hardware. Flugregler, doppeltes RTK-GNSS, ELRS-Empfänger, Regler, Servos; Identifikation am Prüfstand; Parameterdatei für den physischen Rumpf; Erprobung in Manual und Acro auf dem Wasser. Ergebnis: ein Rumpf, der fährt, wie der Entwurf von 2019 es vorsah.

Phase 3, Autonomie. Erprobung in Loiter und Auto, RTK-Basis, BU0836X/Lilaas-Konsole an QGroundControl, Windsensor und Strömungszustand mit Vorsteuerung (Abschnitt 8), auf Prüfstands- und veröffentlichten Koeffizienten mit festem Vertrauensgewicht, bis der Identifikator aus Phase 4 übernimmt, Failsafe-Übungen, Capability-Plot aus gemessenen Schüben. Ergebnis: Positionshalten und Missionen.

Phase 4, Ausbau. Simulation der Stufe 2 mit Wellen und Strömung; Begleitrechner mit Bordbildverarbeitung, Anlegen, Ausweichen und den ROS 2-Knoten aus Abschnitt 9.5; Online-Identifikation,

Effektorwirksamkeit und Persistenz (Abschnitt 8.4); beschränkte Schubverteilung als quadratisches Programm im Frame, Einreichung im Upstream-Projekt; ausfahrbare Effektoren, Halteeinrichtungen und Betriebsstellungen (Abschnitt 6.6) mit den Anordnungen F und G; die Konfigurations-App in die Bodenstation integriert; zweiter und dritter Rumpftyp auf demselben Stack; die maritime Bodenstation aus Abschnitt 7.3, zuerst für einen Rumpf, dann mit den Mehrschiff-, Radar- und Kartenschichten. Ergebnis: μDP als antriebsunabhängiges Produkt.

11 Fazit

Die Studie von 2019 nahm ihre Anforderungen von erfahrenen Schiffsführern und erfüllte sie mit den verfügbaren Werkzeugen. Ihre Bedienersemantik, ihre Konzepte von Bias und Drehpunkt und ihre Praxis, vor dem Bauen zu simulieren, bleiben erhalten.

Die zentrale Änderung ist der Ersatz des Prozentmischens durch Schubverteilung auf Kraftebene über eine Konfigurationsmatrix, die aus einer Effektortabelle gebaut wird. Das behebt die Mängel des Schemas von 2019, lässt einen Satz Knüppel jeden der betrachteten Rumpftypen fahren und gibt den geschlossenen Regelkreisen des Autopiloten ein korrektes Stellgliedmodell.

ArduPilot liefert Modi, Zustandsschätzung und Failsafes. Die Schubverteilung ist in der Motorbibliothek implementiert, nicht in einem Skript, damit die Failsafes die Antriebe erreichen. Windvorsteuerung, ein Strömungszustand und ein identifiziertes, persistentes Modell kommen darum herum, mit einer maritimen Bodenstation darüber. Das ist μDP. Phase 1 beginnt bei der Frame-Skizze, dem Physikmodell und den Abnahmetests in den Anhängen.

Danksagung

Wir danken Professor Thomas Bräunl von der University of Western Australia für seine anhaltende Unterstützung unserer Arbeit und für seine Betreuung der Promotion, die neben ihr läuft.

Wir danken Uwe Bauer von Bauer-Modelle, Eibenstock, für die Fotos in Abbildung 4 und dafür, dass er Ausrüstung in einer Qualität baut, die ein Modell wie ein Schiff fahren lässt.

References

- [1] P.-L. Constant, *Design of a control system for an OSV model*, Oceanomatics Pty Ltd, interner Bericht, April 2019.
- [2] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, Wiley, 2011. Kapitel 12, Control allocation.
- [3] T. A. Johansen und T. I. Fossen, „Control allocation: a survey“, *Automatica*, Bd. 49, Nr. 5, S. 1087–1103, 2013.
- [4] O. J. Sørдалen, „Optimal thrust allocation for marine vessels“, *Control Engineering Practice*, Bd. 5, Nr. 9, S. 1223–1231, 1997.
- [5] T. A. Johansen, T. I. Fossen und S. P. Berge, „Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming“, *IEEE Trans. Control Systems Technology*, Bd. 12, Nr. 1, S. 211–216, 2004.
- [6] K.-P. Lindegaard und T. I. Fossen, „Fuel-efficient rudder and propeller control allocation for marine craft: experiments with a model ship“, *IEEE Trans. Control Systems Technology*, Bd. 11, Nr. 6, S. 850–862, 2003.
- [7] IMCA, *Guidelines for the Design and Operation of Dynamically Positioned Vessels*, IMCA M 103, Rev. 2.1, 2019.
- [8] ArduPilot Development Team, *Rover documentation; motor library; SITL JSON interface; AP_DDS ROS 2 interface; ardupilot_gazebo plugin*, <https://ardupilot.org>, https://github.com/ArduPilot/ardupilot_gazebo, abgerufen September 2026.

- [9] L. Ljung, *System Identification: Theory for the User*, 2. Aufl., Prentice Hall, 1999.
- [10] K. J. Åström und B. Wittenmark, *Adaptive Control*, 2. Aufl., Dover, 2008.
- [11] B. Bingham et al., „Toward maritime robotic simulation in Gazebo“, *Proc. MTS/IEEE OCEANS*, Seattle, 2019. Virtual RobotX, <https://github.com/osrf/vrx>.
- [12] T. Foote und M. Purvis, *REP 103: Standard units of measure and coordinate conventions*, ROS Enhancement Proposal, 2010 (Rev. 2014).
- [13] W. Meeussen, *REP 105: Coordinate frames for mobile platforms*, ROS Enhancement Proposal, 2010.
- [14] Projekte EdgeTX, ExpressLRS, Cockpit (Blue Robotics) und OpenHD, <https://edgetx.org>, <https://www.expresslrs.org>, <https://github.com/bluerobotics/cockpit>, <https://openhdfpv.org>, abgerufen September 2026.

A Der Gleichungssatz von 2019, zur Dokumentation

Mit den Eingängen s (längs), y (quer), n (Gieren), p (Drehpunkt) in $[-1, 1]$, dem Bias-Schub t_0 , dem Leerlauf-Boolean σ , der Achternkonstante C_0 und den an Reglern gesetzten Halbkonstanten k_s, k_n (Anfangswerte $\frac{1}{2}, \frac{1}{4}$, siehe Abschnitt 2.2) lauteten die Sendermischer von 2019

$$\begin{aligned} T_P &= \left(|s| + y + w_{\text{aft}} |n| + t_0 \right) \sigma, & T_S &= \left(|s| - y + w_{\text{aft}} |n| + t_0 \right) \sigma, \\ \alpha_P &= \left(-k_s s - k_n w_{\text{aft}} n \right) \sigma - C_0(1 - \sigma), & \alpha_S &= \left(+k_s s - k_n w_{\text{aft}} n \right) \sigma + C_0(1 - \sigma), \\ T_{B1} = T_{B2} &= \left(-y - w_{\text{fwd}} n \right) \sigma, & w_{\text{aft}} &= \min(p + 1, 1), \quad w_{\text{fwd}} = \min(1 - p, 1), \end{aligned}$$

jeder Ausgang auf $[-1, 1]$ begrenzt, die Azimutwinkel in Einheiten, in denen $\pm 1 \pm 180^\circ$ ab der Außenbordstellung ist, und $C_0 = \frac{1}{2}$ richtet die Antriebe nach achtern. Das ist die kompakte Form aus dem Originaldokument mit den gedruckten Verstärkungen durch Symbole ersetzt.

Die ausgeschriebene Form und die Prosa des Originals enthalten Übertragungsfehler: vertauschte Eingangskanalnummern für Längsbewegung und Drehpunkt, „linker Knüppel“ für den Gierknüppel, „50 % = 0.25“, ein falsches Vorzeichen am Leerlaufterm des Steuerbord-Azimutwinkels, ein fehlendes $+1$ im vorderen Drehpunktgewicht und eine uneinheitliche Nummerierung der Gewichtsfunktionen. Die Simulator-Bildschirmfotos bestätigen, dass die kompakte Form oben das ist, was wir programmiert haben. Auch die qualitative Drehpunkttafel des Originals weicht davon ab: Sie gibt jedem Ende in der Drehpunktmitte halbes Gewicht, während die kompakte Form dort $w_{\text{aft}} = w_{\text{fwd}} = 1$ hat, ein reines Kräftepaar. Die kompakte Form ist das, was die Bildschirmfotos zeigen, und sie ist der Gleichungssatz von 2019 in diesem Papier.

Zwei Eigenschaften des Satzes. Querbewegung nach Steuerbord ($y > 0$) ergibt $T_P = t_0 + y$, $T_S = t_0 - y$ mit beiden Antrieben nach außen gerichtet, beide schieben also das Heck nach Steuerbord, während die Querstrahlruder den Bug nach Steuerbord schieben; das erzeugte Giermoment ist $-2ay + l_B y$ in den Einheiten von Abbildung 3, null nur für $l_B = 2a$. Und die Drehpunktgewichte wirken nur auf das Gieren; ihre Wirkung ist die Form mit offenem Kreis aus Abschnitt 4.6.

B Implementierungsskizze des Marine-Frames

Das folgende Listing ist `code/marine_frame_outline.cpp` aus dem beiliegenden Ordner: eine Skizze des Frames aus Abschnitt 6.5 im Stil der Motorbibliothek, mit ausgeschriebener Verteilungsschleife, Auflösung der schwenkbaren Effektoren, Ausgangsstufe und Vorprüfung, und mit den am Ende vermerkten Integrationspunkten. Es ist ein Entwurfsentwurf, kein kompilierter Code; Namen und Signaturen sind in Phase 1 gegen den Zielzweig zu bestätigen. Die Kommentare im Quelltext sind, wie im Original, auf Englisch.

```

1 // marine_frame_outline.cpp -- outline of the microDP marine frame for
2 // ArduPilot Rover (AP_MotorsUGV). This is a design sketch in ArduPilot style,
3 // not compiled code: names, signatures and integration points are to be
4 // confirmed against the target branch. The allocator itself has no HAL
5 // dependency so that it can be built as a host unit test.
6
7 #pragma once
8 #include <AP_Param/AP_Param.h>
9 #include <AP_Math/AP_Math.h>
10 #include <SRV_Channel/SRV_Channel.h>
11
12 #define MARINE_MAX_EFF 12
13
14 // One effector: parameters (persisted) and state (runtime).
15 class AP_MarineEffector {
16 public:
17     static const struct AP_Param::GroupInfo var_info[]; // MFX_*
18     AP_Int8 type; // 0 off, 1 fixed, 2 azimuth, 3 prop+rudder, 4 steerable jet,
19                 // 5 cycloidal (as 2 with rng = 180, rate large, tr = 0)
20     AP_Float lx, ly; // m from CG, x forward, y starboard (FRD)
21     AP_Float ang; // deg, force direction (0 ahead, 90 starboard): fixed effector's
22                 // direction, or a steerable one's sector centre and park angle
23     AP_Float rng; // deg: sector half-width (180 = unrestricted)
24     AP_Float tf, tr; // N: max thrust ahead / astern (tr = 0: unidirectional)
25     AP_Float rate; // deg/s: steering slew limit
26     AP_Float kt, kr, ku; // type 3 only: rudder side force per unit thrust ahead / astern /
27                        // per u/u, at unit angle. kr ~ 0.2 kt for a Kort nozzle, more with a
28                        // Becker flap rudder, ~0 for an open rudder; measured at the bench
29     AP_Float fsec; // deg: half-width about ang of the force-direction sector that is fully effective
30     AP_Float feff; // effectiveness of a force outside that sector (wash into a skeg: 0)
31     AP_Int8 out; // n: thrust on SRV function MotorN, angle on AzimuthN
32
33     AP_Int8 depl; // 0 fixed, 1 retractable (deploy actuator on MarineAuxN)
34     AP_Float dspd; // m/s: no deployment above this speed
35     AP_Int8 fb; // feedback: 0 none, 1 serial-servo angle, 2 limit switch, 3 ESC rpm
36
37     enum class State : uint8_t { STOWED, DEPLOYING, ACTIVE, RETRACTING, FAILED };
38     State state; // columns are in B only while ACTIVE
39     float angle_rad; // current steering angle relative to ang
40     float thrust_n; // current thrust command
41     bool failed; // dropped from the allocation (GCS or aux switch)
42 };
43
44 // A holding device is not an effector: when SET it holds the vessel and the
45 // allocation stands down. Spud leg, anchor, mooring line. Both tables and the
46 // postures are authored in the configuration app and exported, not typed in.
47 class AP_MarineHold {
48 public:
49     static const struct AP_Param::GroupInfo var_info[]; // HDx_*
50     AP_Int8 type; // 1 spud leg, 2 anchor, 3 mooring line
51     AP_Int8 out; // actuator output (MarineAuxN)
52     AP_Int8 fb; // feedback: limit switch or timed
53     AP_Float t_set, t_raise; // s: timed transitions when no feedback
54     enum class State : uint8_t { RAISED, LOWERING, SET, RAISING, FAILED };
55     State state;
56 };
57
58 // Postures sequence deployable effectors and holding devices with interlocks.
59 // TRANSIT: forward azimuth stowed, tunnels faded; MANOEUVRE: all deployed;
60 // MOORED: holding devices set, thrust zero, position controller stood down;
61 // JACKED: hull clear of the water, everything off. Driven by an aux switch,

```

```

62 // the ground station or a mission command; pre-arm reports the capability of
63 // each posture (rank and envelope of B with the columns that posture keeps).
64 enum class MarinePosture : uint8_t { TRANSIT, MANOEUVRE, MOORED, JACKED };
65
66 // The frame. Owned by AP_MotorsUGV and selected with FRAME_TYPE = Marine.
67 class AP_MotorsUGV_Marine {
68 public:
69     static const struct AP_Param::GroupInfo var_info[];    // MFR_*
70
71     // Build the extended-thrust columns from the effector table.
72     bool setup();
73
74     // Refuse to arm on a degenerate table. Called from AP_MotorsUGV::pre_arm_check.
75     bool pre_arm_check(char *failmsg, uint8_t failmsg_len) const;
76
77     // Called every main-loop cycle from AP_MotorsUGV::output with the mode's
78     // demands already normalised to [-1, 1]. speed is the signed body-axis
79     // surge speed THROUGH THE WATER (negative when making sternway): the EKF
80     // surge velocity minus the body-axis component of the current estimate
81     // when one exists, the EKF velocity alone until then.
82     void output(bool armed, float steering, float throttle, float lateral,
83                float speed, float dt);
84
85     // Lets the position controller and Loiter command sway.
86     bool have_lateral_control() const { return true; }
87
88     void set_effector_failed(uint8_t idx, bool failed);
89
90 private:
91     // MFR_* parameters (the surviving semi-constants of the 2019 study)
92     AP_Float _kx, _ky, _kn;    // full-stick force X, Y (N) and moment N (N m)
93     AP_Float _bias;           // null-space bias, fraction of tf; default 0
94     AP_Float _blank;          // thrust factor while a pod is > 45 deg off target
95     AP_Float _btspd;          // water-relative speed (m/s) at which lateral fixed thrusters fade out
96     AP_Int8 _prio;            // 1 = yaw priority on saturation, 0 = uniform
97     AP_Float _expo;           // thrust-to-command shaping, 0 linear .. 1 sqrt
98     AP_Float _split;          // prop+rudder pairs: split thrust per N of sway demand
99     AP_Float _split_spd;      // ... applied below this water-relative surge speed (m/s)
100
101     AP_MarineEffector _eff[MARINE_MAX_EFF];
102     AP_MarineHold _hold[4];
103     MarinePosture _posture;
104     void update_postures(float speed);    // state machines, interlocks, deploy/hold outputs
105
106     struct Column { float bx, by, bn, w; uint8_t eff; uint8_t kind; }; // kind: T, X, Y
107     Column _cols[2 * MARINE_MAX_EFF];
108     uint8_t _ncols;
109
110     void build_columns(float speed);    // column effectiveness scheduled on
111     // water-relative speed
112     bool allocate(const Vector3f &tau, float *u) const;    // weighted pseudo-inverse (3x3 inverse)
113     // One-directional rule, then saturation policy (MFR_PRIO): a column whose
114     // effector cannot thrust the way the pseudo-inverse asks (tr = 0 and u < 0)
115     // is dropped and the allocation re-solved, one step per such effector;
116     // then the demand is scaled against the limit of the direction commanded.
117     bool allocate_saturating(const Vector3f &tau, float *u) const;
118     float saturation(const float *u) const;    // max |F_i| / T_lim,i, limit by direction
119     void add_bias(float *u) const;    // null-space bias: wash outboard, force
120     // inboard
121     void apply_split(Vector3f &tau, float speed);    // twin-screw walking: see below
122     void resolve_steerable(AP_MarineEffector &e, float fx, float fy, float dt) const;
123     void resolve_rudder(AP_MarineEffector &e, float fx, float fy, float speed, float dt) const;
124     void write_outputs(bool armed);
125     void log(const Vector3f &tau, float sat) const;    // MARF log message
126 };
127
128 // ----- output -----
129 void AP_MotorsUGV_Marine::output(bool armed, float steering, float throttle,
130                                  float lateral, float speed, float dt)
131 {
132     // demand in body FRD: X from throttle, Y from lateral, N from steering

```

```

132 Vector3f tau(throttle * _kx, lateral * _ky, steering * _kn);
133 if (!armed) {
134     tau.zero();
135 }
136 apply_split(tau, speed);    // propeller pairs at low speed: create rudder authority first
137 build_columns(speed);      // column weights use the split thrust for the rudder gains
138
139 float u[2 * MARINE_MAX_EFF];
140 if (!allocate_saturating(tau, u)) {
141     // singular table: cannot happen past pre-arm, but never trust it
142     for (auto &e : _eff) { e.thrust_n = 0; }
143     write_outputs(false);
144     return;
145 }
146 if (armed) {
147     add_bias(u);
148 }
149
150 for (uint8_t i = 0; i < _ncols; i++) {
151     AP_MarineEffector &e = _eff[_cols[i].eff];
152     if (_cols[i].kind == 'T') {
153         e.thrust_n = constrain_float(u[i], -e.tr, e.tf);    // no-op after the one-directional rule
154     } else if (_cols[i].kind == 'X') {
155         // X and Y columns of one effector are adjacent
156         if (e.type == 3) {
157             resolve_rudder(e, u[i], u[i + 1], speed, dt);
158         } else {
159             resolve_steerable(e, u[i], u[i + 1], dt);
160         }
161         i++;
162     }
163 }
164 write_outputs(armed);
165 log(tau, saturation(u));
166 }
167
168 // Flip for reversible units, sector clamp, slew limit, blanking during swing.
169 // A cycloidal unit (type 5) comes through here with rng = 180, a large rate
170 // and tr = 0, so that the flip, sector, slew and blanking steps are no-ops.
171 void AP_MotorsUGV_Marine::resolve_steerable(AP_MarineEffector &e, float fx, float fy, float dt) const
172 {
173     const float centre = radians(e.ang);
174     const float rng     = radians(e.rng);
175     float T = norm(fx, fy);
176     float a = wrap_PI(atan2f(fy, fx) - centre);
177
178     // force-direction sector, checked BEFORE the flip: a force whose wash
179     // would strike a skeg or the neighbouring pod is lost whichever way the
180     // propeller turns. (A catamaran tunnel is handled differently: it is a
181     // fixed thruster with tr = 0 and ang set to the force direction it has
182     // when washing outboard, i.e. inboard, toward the other hull.)
183     const float fsec = radians(e.fsec);
184     if (fabsf(a) > fsec) {
185         if (e.feef <= 0.0f) {
186             const float clipped = (a > 0) ? fsec : -fsec;
187             T *= MAX(0.0f, cosf(a - clipped));
188             a = clipped;
189         } else {
190             T *= e.feef;
191         }
192     }
193
194     if (e.tr > 0) {    // reversible: least travel
195         const float alt = wrap_PI(a + M_PI);
196         if (fabsf(wrap_PI(alt - e.angle_rad)) < fabsf(wrap_PI(a - e.angle_rad))) {
197             a = alt; T = -T;
198         }
199     }
200     if (fabsf(a) > rng) {    // sector: keep useful component
201         const float clipped = (a > 0) ? rng : -rng;
202         T *= MAX(0.0f, cosf(a - clipped));
203         a = clipped;

```

```

204     }
205     const float step = radians(e.rate) * dt;           // slew
206     e.angle_rad += constrain_float(wrap_PI(a - e.angle_rad), -step, step);
207     if (fabsf(wrap_PI(a - e.angle_rad)) > radians(45)) { // blanking
208         T *= _blank;
209     }
210     e.thrust_n = constrain_float(T, -e.tr, e.tf);
211 }
212
213 // Twin-screw walking. A linear allocator cannot make sway from a pair of
214 // propellers and rudders at rest: both rudder gains are small or zero until a
215 // propeller pushes ahead. When |Y_d| exceeds what the fixed lateral thrusters can give
216 // and |speed| < _split_spd, command a split thrust T_s ahead on one propeller
217 // and the astern thrust that cancels it on the other (T_s = _split * |Y_d|,
218 // capped at tf). The ahead rudder then has gain kt*T_s, the astern one only
219 // the reduced kr*T_s of the nozzle inflow (reversed sign), and the couple
220 // 2*b*T_s of the split is subtracted
221 // from tau as a known term before the linear allocation. The side that goes
222 // ahead is chosen so that this couple helps the yaw balance rather than
223 // fighting it. The linear step then puts the stern side force on the ahead
224 // rudder and the forward side force and yaw balance on the bow thruster.
225 void AP_MotorsUGV_Marine::apply_split(Vector3f &tau, float speed)
226 {
227     // ... find the propeller pair (two type-3 effectors), decide the side,
228     // set their pre-commanded thrusts, and subtract the split couple.
229 }
230
231 // Propeller + rudder: the force vector does not rotate when the propeller
232 // reverses. F_x is the propeller thrust; F_y follows the flow over the rudder,
233 // F_y = -(kt*max(T,0) + kr*min(T,0) + ku*u|u|) * delta. The lateral gain is
234 // recomputed every cycle, so astern with sternway the rudder command inverts by
235 // itself. Astern at rest an open rudder has next to no authority; a rudder
236 // behind a Kort nozzle keeps roughly a fifth of the ahead value (kr), with the
237 // sign of sternway, and a Becker flap rudder keeps more.
238 // build_columns() gives the pair's Y column the weight |gy| / gy_ref, which
239 // drops to zero below a threshold and moves sway to the tunnels.
240 void AP_MotorsUGV_Marine::resolve_rudder(AP_MarineEffector &e, float fx, float fy, float speed, float dt)
241 {
242     const
243     const float T = constrain_float(fx, -e.tr, e.tf);
244     const float gy = -(e.kt * MAX(T, 0.0f) + e.kr * MIN(T, 0.0f) + e.ku * speed * fabsf(speed));
245     float delta = 0.0f;
246     if (fabsf(gy) > 1e-3f) {
247         delta = constrain_float(fy / gy, -radians(e.rng), radians(e.rng));
248     }
249     const float step = radians(e.rate) * dt;           // rudder slew
250     e.angle_rad += constrain_float(delta - e.angle_rad, -step, step);
251     e.thrust_n = T;
252 }
253
254 // Thrust to MotorN, angle to AzimuthN. Disarmed: zero thrust, park at centre.
255 // SERVOx_MIN/MAX/TRIM/REVERSED and MOT_SLEWRATE apply downstream as usual.
256 void AP_MotorsUGV_Marine::write_outputs(bool armed)
257 {
258     for (uint8_t i = 0; i < MARINE_MAX_EFF; i++) {
259         AP_MarineEffector &e = _eff[i];
260         if (e.type == 0 || e.out < 1) continue;
261         if (!armed) { e.thrust_n = 0; e.angle_rad = 0; }
262         const float lim = (e.thrust_n < 0) ? e.tr : e.tf;
263         float x = (lim > 0) ? MIN(fabsf(e.thrust_n) / lim, 1.0f) : 0.0f;
264         x = (1 - _expo) * x + _expo * sqrtf(x);           // shaping
265         if (e.thrust_n < 0) x = -x;
266         SRV_Channels::set_output_scaled(SRV_Channel::k_motor1 + e.out -
267             1), x * 100.0f);
268         if (e.type >= 2) {
269             const float ratio = constrain_float(e.angle_rad / MAX(radians(e.rng), 0.01f), -1.0f, 1.0f);
270             SRV_Channels::set_output_scaled(SRV_Channel::k_azimuth1 +
271                 e.out - 1), ratio * 4500.0f);
272         }
273     }
274 }
275 }
276 }

```

```

273 // Pre-arm: at least one effector, every enabled effector bound to an output,
274 // limits and sectors sane, configuration matrix of full row rank.
275 bool AP_MotorsUGV_Marine::pre_arm_check(char *failmsg, uint8_t failmsg_len) const
276 {
277     uint8_t n = 0;
278     for (uint8_t i = 0; i < MARINE_MAX_EFF; i++) {
279         const AP_MarineEffector &e = _eff[i];
280         if (e.type == 0) continue;
281         n++;
282         if (!SRV_Channels::function_assigned(SRV_Channel::k_servo1 +
283             e.out - 1)) {
284             hal.util->snprintf(failmsg, failmsg_len, "Marine: MF%u thrust output not assigned", i + 1);
285             return false;
286         }
287         if (e.type >= 2 &&
288             !SRV_Channels::function_assigned(SRV_Channel::k_azimuth1 + e.out -
289             1)) {
290             hal.util->snprintf(failmsg, failmsg_len, "Marine: MF%u angle output not assigned", i + 1);
291             return false;
292         }
293         if (e.tf <= 0 || e.rng <= 0 || e.rate <= 0) {
294             hal.util->snprintf(failmsg, failmsg_len, "Marine: MF%u limits invalid", i + 1);
295             return false;
296         }
297     }
298     if (n == 0) { hal.util->snprintf(failmsg, failmsg_len, "Marine: no effectors"); return false; }
299     float u[2 * MARINE_MAX_EFF];
300     if (!allocate(Vector3f(1, 1, 1), u)) { // B W^-1 B' singular
301         hal.util->snprintf(failmsg, failmsg_len, "Marine: configuration matrix singular");
302         return false;
303     }
304     return true;
305 }
306
307 // Integration point in AP_MotorsUGV::output():
308 //
309 // if (_frame_type == FRAME_TYPE_MARINE) {
310 //     _marine.output(armed, _steering / 4500.0f, _throttle / 100.0f,
311 //         _lateral / 100.0f, water_speed_x, dt);
312 // }
313 //
314 // where water_speed_x is the body-axis surge velocity from the EKF minus the
315 // body-axis component of the current estimate (see the paper, Section 8.3),
316 // or the EKF velocity alone until that estimate exists;
317 //
318 // with FRAME_TYPE_MARINE added to the frame-type enum, have_lateral_control()
319 // forwarded to the marine object, pre_arm_check() chained, and the new output
320 // functions k_azimuth1..k_azimuth6 added to SRV_Channel. Pivot point and thrust
321 // bias are read through two new auxiliary RC input functions and folded into
322 // tau before allocation (open loop) or into the mode's velocity demand
323 // (closed loop), as described in the paper.

```

C SITL-Physikmodell

code/vessel_sim_json.py: ein minimales 3-DOF-Modell eines Überwasserfahrzeugs für die SITL-JSON-Schnittstelle von ArduPilot. Die Werte für Masse, hydrodynamische Masse und Dämpfung sind Platzhalter für einen Rumpf von 1.6 m und durch Schätzungen vom tatsächlichen Modell zu ersetzen (ein Auslauf- und ein Driftversuch liefern die Dämpfung; die hydrodynamische Masse kann aus der Streifenmethode kommen oder einfach als 10 % in Längs- und 80 % in Querrichtung der Verdrängung angesetzt werden).

```

1  #!/usr/bin/env python3
2  """Minimal 3-DOF surface-vessel physics for ArduPilot SITL (JSON backend), microDP.
3
4  Run SITL with:  sim_vehicle.py -v Rover --model JSON -f rover --console --map
5  then start this script. SITL sends servo PWM on UDP 9002; we reply with the
6  vehicle state. The marine frame runs unchanged inside SITL, so the whole
7  chain (sticks -> modes -> allocator -> thrusters -> hull) is exercised.
8
9  Model (Fossen, 3-DOF manoeuvring):   $M \dot{u} + C(u) u + D(u) u = \tau$ 
10 State  $u = [u, v, r]$  (surge, sway, yaw rate),  $\eta = [x, y, \psi]$  in NED.
11 Thrusters are described with the same effector table as the frame's MFx_*
12 parameters (layout A of the paper). Angles are force directions:  $\text{ang} = 0$  is
13 wash aft, push ahead, the park position.
14 """
15 import json
16 import math
17 import socket
18 import struct
19 import time
20
21 import numpy as np
22
23 # ----- vessel -----
24 M = np.diag([18.0, 29.0, 3.2])          # mass + added mass [kg, kg, kg m^2]:
25                                       # 16 kg displacement, +10 % surge, +80 % sway
26 D_LIN = np.diag([1.5, 6.0, 1.2])       # linear damping
27 D_QUAD = np.diag([4.0, 25.0, 3.0])     # quadratic damping
28
29 # effector table: type, lx, ly, ang_deg, rng_deg, tf, tr, rate_deg_s,
30 # pwm slot for thrust (0-based servo channel), pwm slot for angle (or None)
31 EFFECTORS = [
32     dict(type=2, lx=-0.65, ly=-0.15, ang=0, rng=180, tf=5.0, tr=3.0,
33           rate=180, ch_t=0, ch_a=1),
34     dict(type=2, lx=-0.65, ly=+0.15, ang=0, rng=180, tf=5.0, tr=3.0,
35           rate=180, ch_t=2, ch_a=3),
36     dict(type=1, lx=+0.60, ly=0.00, ang=90, rng=0, tf=2.0, tr=2.0,
37           rate=0, ch_t=4, ch_a=None),
38 ]
39 for e in EFFECTORS:
40     e["angle"] = 0.0 # current steering angle relative to ang [rad]
41
42
43 # SERVOx_MIN/MAX/TRIM assumed 1000/2000/1500 for thrust and angle alike.
44 def pwm_to_thrust(pwm, tf, tr, expo=0.5):
45     x = max(-1.0, min(1.0, (pwm - 1500) / 500.0))
46     # invert the (1-expo)*x + expo*sqrt(x) shaping used by the allocator
47     mag = abs(x)
48     if expo > 0:
49         # solve (1-e) m + e sqrt(m) = mag for m by fixed point
50         m = mag
51         for _ in range(8):
52             if expo < 1:
53                 m = max(0.0, (mag - expo * math.sqrt(m)) / (1 - expo))
54             else:
55                 m = mag * mag
56     else:
57         m = mag
58     return (tf if x >= 0 else tr) * m * (1 if x >= 0 else -1)
59
60

```

```

61 def pwm_to_angle(pwm, rng_deg):
62     x = max(-1.0, min(1.0, (pwm - 1500) / 500.0))
63     return x * math.radians(max(rng_deg, 1))
64
65
66 def thruster_forces(pwm, dt):
67     tau = np.zeros(3)
68     for e in EFFECTORS:
69         T = pwm_to_thrust(pwm[e["ch_t"]], e["tf"], e["tr"])
70         centre = math.radians(e["ang"])
71         if e["type"] == 1:
72             a = centre
73         else:
74             target = pwm_to_angle(pwm[e["ch_a"]], e["rng"])
75             step = math.radians(e["rate"]) * dt
76             e["angle"] += max(-step, min(step, target - e["angle"]))
77             a = centre + e["angle"]
78             fx, fy = T * math.cos(a), T * math.sin(a)
79             tau += [fx, fy, e["lx"] * fy - e["ly"] * fx]
80     return tau
81
82
83 def step(eta, nu, tau, dt):
84     u, v, r = nu
85     m_u, m_v, _ = np.diag(M)
86     C = np.array([[0, 0, -m_v * v], [0, 0, m_u * u], [m_v * v, -m_u * u, 0]])
87     D = D_LIN + D_QUAD * np.abs(nu)
88     nu_dot = np.linalg.solve(M, tau - C @ nu - D @ nu)
89     nu = nu + nu_dot * dt
90     psi = eta[2]
91     c, s = math.cos(psi), math.sin(psi)
92     R = np.array([[c, -s, 0], [s, c, 0], [0, 0, 1]])
93     eta = eta + R @ nu * dt
94     return eta, nu, nu_dot
95
96
97 def main():
98     sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
99     sock.bind(("", 9002))
100     eta, nu = np.zeros(3), np.zeros(3)
101     t = 0.0
102     while True:
103         data, addr = sock.recvfrom(1024)
104         magic, frame_rate, frame_count = struct.unpack("<HHI", data[:8])
105         pwm = struct.unpack("<16H", data[8:40])
106         if magic != 18458:
107             continue
108         dt = 1.0 / max(frame_rate, 1)
109         tau = thruster_forces(pwm, dt)
110         eta, nu, nu_dot = step(eta, nu, tau, dt)
111         t += dt
112         psi = eta[2]
113         vel_ned = [nu[0] * math.cos(psi) - nu[1] * math.sin(psi),
114                   nu[0] * math.sin(psi) + nu[1] * math.cos(psi), 0.0]
115         state = {
116             "timestamp": t,
117             "imu": {
118                 "gyro": [0.0, 0.0, float(nu[2])],
119                 "accel_body": [float(nu_dot[0] - nu[2] * nu[1]),
120                               float(nu_dot[1] + nu[2] * nu[0]), -9.80665],
121             },
122             "position": [float(eta[0]), float(eta[1]), 0.0],
123             "attitude": [0.0, 0.0, float(psi)],
124             "velocity": vel_ned,
125         }
126         sock.sendto(("\\n" + json.dumps(state) + "\\n").encode(), addr)
127
128
129 if __name__ == "__main__":
130     main()

```

D Parameterreferenz

Tabelle 14: Parameter des Marine-Frames (Arbeitsnamen).

Name	Standard	Bedeutung
MFR_KX, MFR_KY, MFR_KN	6, 3, 2	Längskraft, Querkraft (N) und Giermoment (N m) bei vollem Knüppelausschlag.
MFR_BIAS	0	Nullraum-Bias-Schub als Anteil von T_f , nur Azimutpaare; standardmäßig aus (Abschnitt 4.2).
MFR_BLANK	0.3	Schubfaktor, solange ein schwenkbarer Effektor mehr als 45° von seinem Ziel entfernt ist.
MFR_BTSPD	0.6	Fahrt durchs Wasser (m/s), bei der seitliche feste Antriebe vollständig ausgeblendet sind; 0 deaktiviert.
MFR_PRI0	1	Sättigungsstrategie: 1 Giervorrang, 0 gleichmäßige Skalierung.
MFR_EXPO	0.5	Formung von Schub zu Kommando, 0 linear, 1 Quadratwurzel.
MFR_SPLT, MFR_SPSPD	0, 0.3	Nur Propeller-Ruder-Anlagen: Aufteilungsschub je Einheit Quererforderung (Anteil von T_f je N) und die Fahrt durchs Wasser (m/s), unterhalb derer die Aufteilungsregel gilt.
MFx_TYPE	0	0 aus, 1 fest, 2 Azimut, 3 Propeller + Ruder, 4 schwenkbarer Wasserstrahl, 5 zyklodal (aufgelöst wie ein Azimut mit unbeschränktem Sektor und ohne Schwenkbegrenzung).
MFx_LX, MFx_LY	0	Position relativ zum Schwerpunkt (m), x voraus, y nach Steuerbord.
MFx_ANG	0	Fest: Kraftrichtung (Grad, 0 voraus, 90 Steuerbord). Schwenkbar: Sektormitte und Parkwinkel, in derselben Kraftkonvention (0 = Strahl nach achtern, Schub voraus).
MFx_RNG	180	Halbe Breite des erlaubten Sektors (Grad).
MFx_TF, MFx_TR	1, 1	Maximaler Schub voraus und rückwärts (N); TR=0 kennzeichnet eine Einrichtungseinheit, etwa ein Katamaran-Querstrahlruder, eingetragen mit der Kraftrichtung, die es beim Ausströmen nach außen hat, zum anderen Rumpf hin.
MFx_RATE	180	Maximale Schwenkrate (Grad/s).
MFx_FSEC, MFx_FEFF	180, 1	Kraftrichtungssektor: halbe Breite (Grad) um ANG, in der die Kraft voll wirksam ist, und Wirksamkeit außerhalb (0–1), für einen Antrieb, dessen Strahl einen Skeg oder einen Nachbarantrieb träge.
MFx_KT, MFx_KR, MFx_KU	0, 0, 0	Nur Propeller + Ruder: Ruderseitenkraft je Schubeinheit voraus, je Schubeinheit rückwärts und je Einheit $u u $, bei Einheitsausschlag; k_R liegt bei einer Kortdüse um 0.2 k_T , bei einem Beckerruder höher, bei einem freien Ruder nahe null, am Prüfstand gemessen. Vorzeichenkonvention: Ruder nach Steuerbord gibt voraus Kraft nach Backbord; ein positives k_R kehrt das rückwärts um, wie die Rückwärtsfahrt.
MFx_OUT	0	Ausgangsindex n : Schub auf Motor n , Lenkwinkel auf Azimuth n , Hilfsfunktion (Umkehrschaufel, Ausfahren) auf MarineAux n .
MFx_DEPL, MFx_DSPD	0, 0	Ausfahrbare Einheit (0 fest, 1 einziehbar) und die Fahrt (m/s), oberhalb derer sie nicht ausfahren darf.
MFx_FB	0	Rückmeldequelle: 0 keine, 1 Winkel vom seriellen Servo, 2 Endschalter, 3 Drehzahl aus der Reglertelemetrie.
HDx_*	—	Halteeinrichtungen (Spudpfahl, Anker, Festmacher): Typ, Aktorausgang, Rückmeldeeingang, Setz- und Hebezeiten.

E Beispielhafte Hardware

Tabelle 15: Beispielhafte Hardware für Phase 2, Anordnung A. Ohne Preise; die Verfügbarkeit ändert sich schnell.

Posten	Anmerkungen
Flugregler	ArduPilot-Platine der STM32H7-Klasse mit mindestens 8 PWM-Ausgängen, 3 UARTs, CAN optional (Klasse Pixhawk 6C/6X, Cube Orange+, Matek H743).
GNSS	Zwei RTK-Empfänger (Klasse u-blox F9P) mit Helix- oder Patchantennen ≥ 0.3 m auseinander auf der Mittschiffslinie; RTK-Basis oder NTRIP.
Funk	Sender der EdgeTX-Klasse mit internem ELRS 2.4 GHz; ELRS-Empfänger mit CRSF zum Autopiloten. DP-Konsole nach Abschnitt 5.4 ab Phase 3.
Motoren	Bürstenlos mit niedrigem Kv (100–400 U/min je V) mit Getriebe oder Zahnriemen 2:1 bis 4:1 je Einheit (Abschnitt 6.7); Bürstenmotoren mit Getriebe annehmbar.
Regler	Bidirektionale Regler (bürstenlos im 3D-Modus, oder bürstenbehaftet), für die Motoren mit Reserve ausgelegt; einer je Antrieb.
Azimutantrieb	180°-Servo mit Getriebe und Umschlaglogik, oder Mehrumdrehungs-Bus-Servo mit Rückmeldung.
Windsensor (Phase 3)	Kleines Ultraschallanemometer, oder Windfahnenpotentiometer plus Schalen-/Propelleranemometer mit Impulsausgang, im Masttopp oder auf eigener Stange, frei von Abschattung durch Aufbauten, Mast oder Antennen; von der Windfahnenbibliothek gelesen.
Fahrtmesser durchs Wasser (optional)	Schaufelrad- oder elektromagnetische Logge über NMEA 0183, oder die Wasserspur einer kleinen Doppler-Logge; nur als Gegenprobe der Strömungsschätzung auf gleichmäßigem Transit, da der Antriebsstrahl sie beim Manövrieren erreicht.
DP-Konsole (Phase 3)	Dreiachs-Joystick und Azimuthebel von Lilaas, Kursrad, Schalter, an einer USB-Schnittstelle Leo Bodnar BU0836X in QGroundControl; später die maritime Bodenstation von Oceanomatics (Abschnitt 7.3).
Begleitrechner (Phase 4)	Raspberry Pi 5 oder Jetson Orin Nano, Kamera für die Bordbildverarbeitung, optional 2-D-Lidar; oder ein integriertes Bündel aus Pixhawk und Raspberry Pi (Klasse Holybro, keine Verbindung zum Hersteller), wenn es sich zu vertretbaren Kosten abdichten lässt (Abschnitt 7.4).
Stromversorgung	3S–4S-LiPo, eigenes Servo-BEC, Stromsensor zum Autopiloten, Hauptsicherung.