Master Thesis

Control of a boat in a vector field and compensation of currents with limited sensor data

Calvin Lacher, B.Sc.

This work has been prepared at the Institute for Engineering Mathematics of the Faculty of Aerospace Engineering of the Bundeswehr University Munich and at Lab-Sticc of ENSTA Bretagne.

Author:	Calvin Lacher
Matriculation Number:	1151184
Course of studies:	Aerospace Engineering
Supervisors:	Prof. Dr. Matthias Gerdts, Prof. Dr. Luc Jaulin
Date:	August 24, 2019

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt und keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum

Name und Unterschrift

Abstract

In this work a controller based on a feedback linearization is used to improve the control of a boat along the direction of vectors within a vector field. A Kalman filter is used to compensate for statistical disturbances. This approach is implemented and improved on an already existing system in the *Boatbot* project. Simulated and experimental results show the capability of the control strategy as opposed to a controller that was used previously. Additionally, hydrographic data from the experiments is collected and evaluated.

Keywords: Feedback linearization, vector field, state estimation

Contents

1 Introduction **1**

1.1 Motivation and Goal . . . 1
1.2 Boatbot Project . . . 2
1.2.1 Objective . . . 2
1.2.2 Previous Projects . . . 2
1.2.3 Area of Deployment . . . 4
1.3 Guidance, Navigation and Control (GNC) . . . 6
1.4 Control Theory . . . 7
1.4.1 Control of nonlinear systems . . . 8
1.4.2 Feedback linearization method . . . 8
1.5 Kalman-Filter . . . 10
1.6 Robot operating system (ROS) . . . 14
1.7 Structural Overview . . . 15

2 Control Problem **16**

2.1 Control of a boat in a vector field . . . 16
2.1.1 Restrictions . . . 16
2.1.2 Vector field control . . . 16
2.1.3 Path following . . . 19
2.2 Mathematical model of the boat . . . 20
2.2.1 Dubin's car . . . 20
2.2.2 Sea currents and speed over water . . . 21
2.2.3 Second order model . . . 23
2.3 PD control problems . . . 23
2.4 Feedback linearization . . . 26
2.4.1 Field following controller . . . 26
2.4.2 Offset due to currents . . . 27
2.4.3 Currents compensating controller . . . 28
2.5 Currents and speed over water estimation . . . 30
2.5.1 Mathematical model . . . 30
2.5.2 Discretization . . . 31

3 Implementation **33**
3.1 Robot . . . 33
3.1.1 Hardware . . . 34
3.1.2 Software . . . 40
3.2 Magnet sensor setup . . . 44
3.2.1 Hardware . . . 44
3.2.2 Software . . . 45

4 Validation **46**
4.1 Path following . . . 46
4.1.1 Simulation . . . 47
4.1.2 Deployment . . . 52
4.2 Currents estimation . . . 59
4.2.1 Controller performance . . . 59
4.2.2 Estimation results . . . 60
4.3 Depth estimation . . . 64
4.3.1 Sonar . . . 64
4.3.2 Camera . . . 64
4.4 Magnetic data collection . . . 64
4.4.1 Magnetometer problems . . . 65
4.4.2 Visualization and findings . . . 66

5 Conclusion **68**

Appendix **69**

List of Figures

1.1 *Boatbot* deployed on a magnetic data survey on July 18th, 2019. 1
1.2 The **VAIMOS** autonomous sailboat during *Submeeting 2019* 3
1.3 *Boatbot* with measurement equipment, either with or without an additional kajak for towing, modified from [2] 4
1.4 Search area for *La Cordelière* at *Le Gullet* at $48°20'$ N, $4°37'$ W 5
1.5 Brique 8 in the westernmost part of the survey area. This brique is later used for the validation of the control system. 5
1.6 Transformed and decoupled nonlinear system, modified from [13]. 9
1.7 The cycle of prediction and correction in a Kalman filter, modified from [26]. . 12

2.1 Curve and vector field of an inverse tangent function. 18
2.2 Curve and vector field of an hyperbolic tangent function. 19
2.3 Boat in a (x_1, x_2)-plane 21
2.4 Boat with additional currents, separating heading x_3 and true course. 22
2.5 PIDsin controller, the feedback here is the heading, provided by the IMU. . . . 24
2.6 Dubin's car approaching the line with a PDsin controller on a hyperbolic tangent vector field as in equation (2.5). 25
2.7 Equilibrium of a Dubin's car following a vector field with currents when not taking the currents into consideration 28

3.1 Size comparison of the old and new embedded computer systems. 34
3.2 **SBG Ellipse 2-A** inertial sensor, modified from [21] 35
3.3 RTK principle [25], *Boatbot* uses a **NEO-M8P-2** as a rover 35
3.4 The sensors are place on a rod beside the helm of the boat. 37
3.5 **Polulu Jrk 12v12** controller board, from [18] 37
3.6 Embedded computer system of *Boatbot* 38
3.7 Transformation function of the steering servo. 39
3.8 Software Architecture 40
3.9 The OpenCPN interface 43
3.10 The **FGM3D TD** magnetometer data logging set 45

4.1 The old controller is a PDsin controller using a vector field for static setpoints. . 46
4.2 The new controller is a feedback linearization based proportional controller. A vector field acts as a setpoint input. 46

4.3 The new controller follows the vector field perfectly. 48
4.4 Converging to the line in a second order model. 49
4.5 Currents make it impossible for the old controller to converge. 50
4.6 Influence of parameters r and K_P on the control, here the new controller. . . . 52
4.7 GPS track of brique 8. 53
4.8 Performance of old and new controllers on brique 8. 54
4.9 Controller output sample from deployment minute 10 to 20. 55
4.10 Offset to track during mission. 56
4.11 Old controller - comparison of desired course with heading and true course. . . 57
4.12 Old controller - visualization of a section of the mission. The green arrow is the true course. 57
4.13 New controller - comparison of desired course with heading and true course. . . 58
4.14 New controller - visualization of a section of the mission. The green arrow is the true course. The black arrow is the estimated currents. 58
4.15 Performance of currents compensating controller on brique 8. 60
4.16 Direction of currents during simulation and deployment. 61
4.17 Currents speed during simulation and deployment. 62
4.18 Estimated speed over water. 63
4.19 The depressor, as seen with a **GoPro** from roughly three meters distance. The attached light is barely visible from farther away. 65
4.20 The depressor, as seen with a **GoPro** from roughly three meters distance. The attached light is barely visible from farther away. 65
4.21 Visualization from the **Data_Processor.py** program in **Google Earth**. 67

List of Tables

1.1 Overview of the discrete matrices and parameters used in a Kalman filter. . . . 12

3.1 Sensor characteristics from manuals provided by manufacturers. Note that accuracy and update rate of the GNSS sensor is heavily dependent on the availability of satellites (GPS,GLONASS,etc.) and the availability of RTK-corrections. 36

3.2 Nodes in `zodiac_command` 43

4.1 First simulation parameters 47

4.2 Second simulation parameters 48

4.3 Third simulation parameters 49

4.4 Estimated environmental conditions during the experiments. 52

4.5 Specifications of missions and controller configurations. 53

4.6 Simulation parameters 59

Chapter 1

Introduction

1.1 Motivation and Goal

Marie la Cordelière was the flagship of the Franco-Bretonic navy during the Battle of Saint-Mathieu, which took place in the Iroise sea in 1512 [7]. After an unexpected attack of the British navy, it is assumed that *La Cordelière* covered the Franco-Bretonic fleet's retreat while engaging *Regent*, the flagship of the British navy. A great explosion onboard *La Cordelière* caused both ships to sink. The exact location of the battle or the sunk ships is unknown to this day. The ships are of great interest though, the *Marie la Cordelière* being one of the biggest ships of her time and also among the most advanced, thus an endeavour for finding the ships has been underway for several years.

An existing boat, *Boatbot* is able to roughly follow a GPS-track using a controller based on GPS and compass data. Restricted by limited sensor data we want to improve the control system to build a reliable system which is able to follow the GPS-track more accurately, using a feedback linearization for control and a Kalman filter for the estimation of disturbances. The aim for this boat is to be set up as a test vehicle for robotics and also as a reliable sensor platform for the use in oceanographic data mapping. Figure 1.1 shows *Boatbot* deployed at "Le gullet" in the vicinity of Brest, France.

Figure 1.1: *Boatbot* deployed on a magnetic data survey on July 18th, 2019.

1.2 Boatbot Project

In this section, the topic dealt with in this thesis will be embedded in an overall context. *Boatbot* is the name of an inflatable boat powered by an outboard motor of the Lab-STICC of **ENSTA Bretagne**. Equipped with a steering servo, GPS receiver, inertial measurement unit, motor control and an embedded computer, it is able to follow a given GPS path. For a comprehensive overview of the setup of hardware and software used in the context of control on *Boatbot*, see chapter 3.1.

The project is thus concerned with the automatic control of small surface vehicles in general and with the improvement and validation of the existing control hardware and software of *Boatbot* in particular. In addition, the validation and the use of a structure for underwater magnetic measurements is of interest. See chapters 4.1 and 4.2 for the validation of the controller and chapter 4.4 for the data mapping setup.

1.2.1 Objective

According to the initiators objectives of the project are to show that such a robot can be used to perform autonomous missions in underwater archaeological research, especially when this is connected to exploring "very large and inhospitable areas". In order to be used as an educational as well as a scientific vessel it is also important to maintain a level of complexity that allows for operators to be switched with low effort. This is, because a rotating crew system, with at least two people on board at all times during deployment, has to be implemented in order to enable *Boatbot* to stay in its deployment area over extended periods of time [2], [3].

Boatbot is used as part of an ongoing search for *La Cordelière*, the Franco-Bretonic flagship of the Battle of Saint-Mathieu, which sunk in the Iroise sea in 1512 [7]. This is done by using a magnetic field sensor to scan a designated portion of the sea and then map the results using also recorded GPS-Data, the implementation of which can be found in chapter 3.2. The main idea is to try and detect the cannons of the ship, which supposedly contain the highest amount of magnetic material aboard the vessel.

1.2.2 Previous Projects

The control software of *Boatbot* is based on an algorithm originally developed for the **VAIMOS** autonomous sailing boat. This algorithm is based on the idea of not following waypoints and thus not having a control law against the use of wind and currents, but using a vector field to follow a line as closely as possible between two waypoints. Among other tests, this approach was validated with an experiment on 17-18 January 2012 between Brest and Douarnez [1], [12].

Figure 1.2: The **VAIMOS** autonomous sailboat during *Submeeting 2019*

Boatbot was already used as an autonomous test vehicle in 2018. A control algorithm has been developed for this purpose based on the **VAIMOS** Autonomous Sailboat algorithm in conjunction with a PID controller [16].

A survey for *La Cordelière* was subsequently performed in the months of June and July in 2018 during *Submeeting 2018*, a conference of robotics and oceanography from 22-23 May 2018 in Trez-Hir, France. The conference encompassed the test of different types of robots, sensors, controllers, techniques, and algorithms to search for buried wrecks. *Boatbot* was tested as an autonomous zodiac designed to tow AUVs or magnetometers. [2].

Students of **ENSTA Bretagne** have taken over the *Boatbot* project and in the process extended software and control algorithms. From 05-07 February 2019 tests were carried out at the lake of Guerledan with experimentation on the hardware setup and the existing line following program. The next step would equipping a towing vehicle (canoe), which had been used during *Submeeting 2018* with GPS and INS, from which in turn the sensor setup should be pulled. [20]

The aim of this extension would be to be able to independently locate the towed vehicle independently and not rely on the logs of *Boatbot* for the post-processing of the sensor data. It would also increase the accuracy of localization. The future of the acquisition of underwater archaeological data at **ENSTA Bretagne** sees Boatbot taken out of the equation and towing the sensors with UAVs.

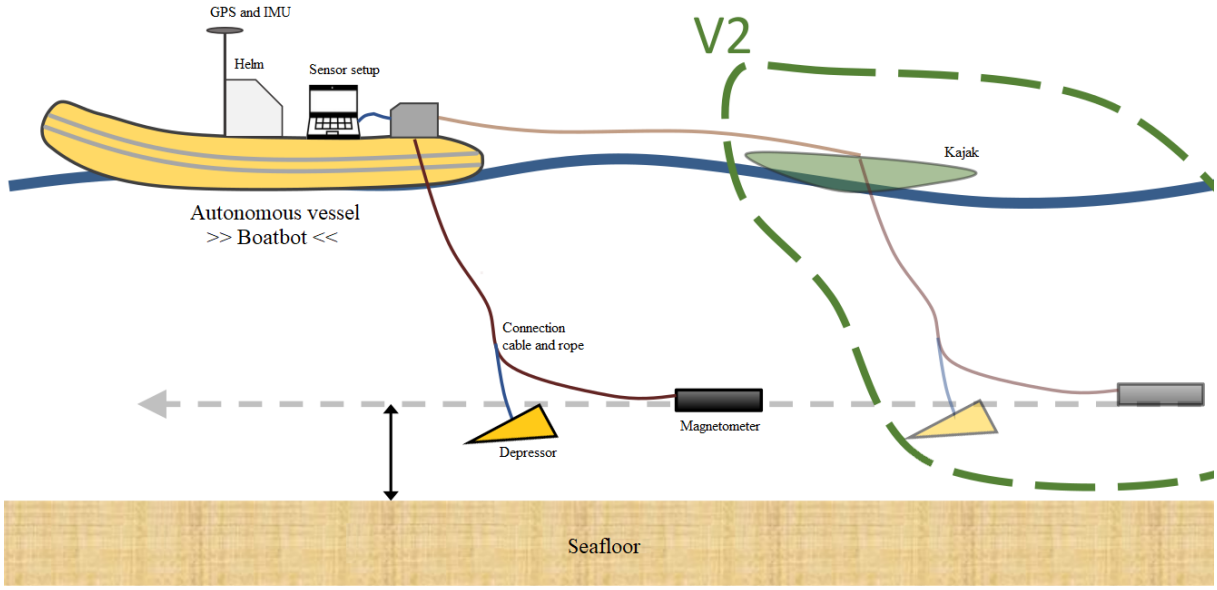


Figure 1.3: *Boatbot* with measurement equipment, either with or without an additional kajak for towing, modified from [2]

However, this was not realized in this work, as the aim here is mainly the improvement of the control algorithm of *Boatbot*. Magnetic sensor data is acquired but these surveys will be treated as experiments for the control setup in this work. The corresponding surveys were conducted as part of *Submeeting 2019*, held from 12-15 June 2019 in Trez-Hir, with the participation of **ENSTA Bretagne**.

1.2.3 Area of Deployment

Submeeting 2019 was again dedicated to testing the control and data collection algorithms of robots, which are also to be used in the search for wrecks in the maritime area southwest of Brest. This includes the *Boatbot* project. The purpose of the boat was to test hardware and software created in the course of this project and also to expand on the amount of magnetic data already collected in the sea area during *Submeeting 2018*.

The main search area is off the coast of Minout, at $48°20'$ N, $4°37'$ W. It is divided into 52 rectangles, which will be referred to as “briques” in the following. It is assumed possible that the Battle of Saint-Mathieu took place here and both *Marie la Cordelière* and *Regent* did sink in this area and thus may be found by using a robot.

The research area is divided into two main areas, west and east of Minout. West of Minout, water depth is relatively constant, ranging from 12-15 meters in the briques nearest to the shore and up to 25 meters at some distance off the coast. The differences inside one brique range up

to three meters, which is beneficial for the acquisition of magnetic data, as a depth control for the magnetic sensor has not been set up. During the course of this project, mainly data from the area west of Minout was acquired due to deployment restrictions on the eastern side. Inside brique 52, which was the only one where data was acquired from the eastern side, depth ranged from 15 to 50 meters, the data therefore not being very meaningful.

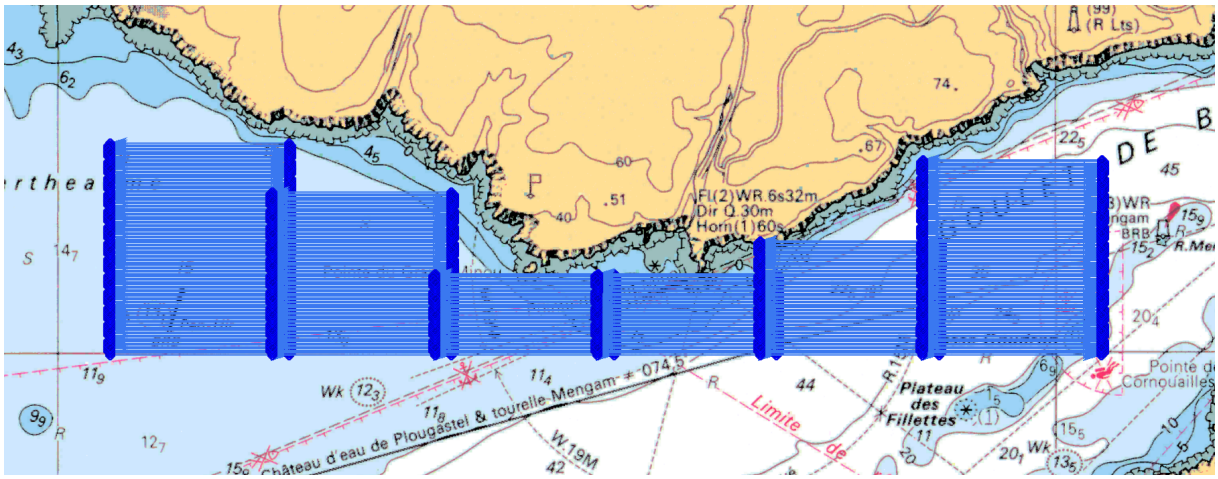


Figure 1.4: Search area for *La Cordelière* at *Le Gullet* at 48°20′ N, 4°37′ W

One brique is approximately 1100 meters long and exactly 100 meters wide. 42 waypoints comprise a set of six lines in west to east direction and five east to west. There is a distance of 10 meters between the lines. For the data this results in a much higher resolution in longitudinal direction than latitudinal. The total distance for one brique comes close to 13km, which roughly corresponds to seven nautical miles.

Figure 1.5: Brique 8 in the westernmost part of the survey area. This brique is later used for the validation of the control system.

Taking the magnet sensor setup and the capabilities of the boat into consideration, surveys on a brique are typically performed at speeds over water around four knots. A survey on one brique therefore usually takes around ninety minutes to two hours to complete, depending also on waves and currents. A limiting factor for the speed during the missions was the attachment of the underwater sensor setup on the boat, which should not be exposed to excessive loads.

1.3 Guidance, Navigation and Control (GNC)

The process of processing data on a robot, such as *Boatbot*, can be abstracted from the data input, i.e. sensor data and missions in the form of waypoint chains, to the control signal and the resulting system response in three stages: Gudiance, navigation and control. Systems built in this way are called GNC systems. They can be found inside a multitude of autonomous and semi-autonomous systems with the purpose of controlling their movements. These include automobiles, aircraft, spacecraft and ships. As such, *Boatbot* follows the principle of guidance, navigation and control.

Guidance

Gudiance encompasses the determination of a desired trajectory from the current position to a designated target. Guidance techniques include but are not limited to waypoint following, line of sight guidance, pure pursuit or constant bearing [4], [10]. *Boatbot* is guided along a GPS-track between two waypoints with a line following algorithm. Thus, in the simplest possible layout, the guidance is a function that determines a heading solely based on the offset to the track. This approach does not originally come from the development of autonomous vehicles. Rather, the pioneering work in this area has been made among other things in guidance systems for missiles [23].

Navigation

Navigation is the determination of the vehicle's own position, velocity and heading at a given time (state vector). Thus, for heading autopilots and dynamic positioning systems, navigation means integrating GPS measurements with inertial navigation systems [10]. Not all state variables can be measured, however. A Kalman-Filter offers one possible solution in estimating missing state variables based on a mathematical model of the vehicle, measurements and previous states. *Boatbot* moves relatively slowly; position (GPS) and heading (IMU) are sufficient to build a GNC-system. But the accuracy can be improved when a Kalman-Filter is used to estimate the speed of the boat and surrounding currents.

Control

Use of the acquired trajectory and system states to manipulate the surrounding forces, here by way of turning the steering wheel of *Boatbot*, in order to fulfill the guidance commands. Different approaches to controal are common for marine vehicles, including PID, linear quadratic, feedback linearization, backstepping, etc. [10]. In this work, two of these approaches will be considered, namely PID and feedback linearization. It will be shown that the feedback linearization is more suitable for the guidance las chosen for *Boatbot*.

1.4 Control Theory

The control theory as a branch of applied mathematics considers dynamic systems whose behavior can be influenced by so-called input variables from the outside. It is a branch of control engineering in which diverse systems of science, engineering and medicine, economics, biology, ecology and social sciences are described using mathematical formulas, i.e. they are analyzed with regard to their stability, controllability and the system variables, which are referred to in the following as states or state variables. Afterwards, mathematical methods are developed which allow the control of these systems.

Mathematically, such systems can be represented in various ways. Common is the representation in the state space representation, where the states occur as variables $\mathbf{x(t)}$, or in the discrete-time case $\mathbf{x_k}$. $\mathbf{u(t)}$ describes inputs and $\mathbf{s(t)}$ system errors. Measurement equations $\mathbf{y(t)}$ are represented in the same manner [27]:

$$\begin{aligned} \dot{\mathbf{x}}(\mathbf{t}) &= f\left(\mathbf{t}, \mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t}), \mathbf{s}(\mathbf{t})\right) \\ \mathbf{y}(\mathbf{t}) &= \mathbf{h}\left(\mathbf{x}(\mathbf{t})\right) + \mathbf{w}(\mathbf{t}) \end{aligned} \tag{1.1}$$

Another form of representation is, for example, differential algebraic equations, which arise from the dynamic systems of robots when additional path restrictions are imposed. [6]. These restrictions are also represented as differential equations. They are additional conditions that the state variables must meet. Problems of this kind are solved with the Optimal Control Theory [11].

$$\begin{aligned} \dot{\mathbf{x}}(\mathbf{t}) &= f\left(\mathbf{t}, \mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t}), \mathbf{s}(\mathbf{t})\right) \\ \mathbf{0} &= \mathbf{h_{res}}\left(\mathbf{t}, \mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t}), \mathbf{s}(\mathbf{t})\right) \end{aligned} \tag{1.2}$$

Boatbot is described by the state space representation. However, it would be conceivable to diversify the scheme in future projects by imposing path limitations, using differential algebraic equations. This might be used for evasive maneuvers.

To precisely describe the system dynamics of a boat, such as *Boatbot*, all occurring forces and disturbances must be taken into account. In this work, many simplifications are assumed, see chapter 2.1.3. For a holistic overview of the external influences acting on a boat and a resulting exact mathematical description, see [5], where a full investigation and development of a small boat mathematical model is performed.

1.4.1 Control of nonlinear systems

A system that moves freely in three-dimensional or two-dimensional space, i.e. being able to be mathematically represented using translations and rotations only by means of nonlinear equations, satisfactory results can only be achieved if these nonlinearities are taken into account accordingly. Boats are such systems. So the goal is to find controllers for non-linear systems.

Modern non-linear controllers are therefore based on a variety of different control approaches, such as PID control, linear quadratic, optimal and stochastic control, H_∞-control methods, fuzzy systems, neural networks, nonlinear control theory and others. Control systems actually used on ships and boats include PID linear quadratic optimal control, state feedback linearization, and integrator back stepping. [10].

In this work we will focus on two of these approaches, namely PID control and feedback linearization. In this thesis PID control will in fact only be expanded upon in the context of pointing out the problems that *Boatbot* had with this approach and, in the following, to explain the advantages of feedback linearization in this project.

1.4.2 Feedback linearization method

The feedback linearization allows us to design the autopilot for yaw dynamics directly on the basis of a non-linearity. We get a yaw-dynamic model that better describes the steering behavior of a boat than would be the case with a linear regulator.

An additional consideration of the rudder saturation and rudder angle slew rate limitation also allows for the execution of a feedback linearization controller that makes very quick course changes without overshooting [24]. This could be considered in a future work as this is one of the main problems found in chapter 4.1.

Feedback linearization is a method of linearizing the output of a nonlinear system in order to find a nonlinear controller. The approach involves coming up with a transformation of the nonlinear system into an equivalent linear system through a change of variables and a suitable control input. The method described here is the the method which was used to find the controller for *Boatbot* and follows the principle from [13], [16].

A nonlinear system can be described by a state equation and a measurement equation, $\mathbf{f}(\mathbf{x})$ being the nonlinear system transition and $\mathbf{g}(\mathbf{x})$ the input equations:

$$\begin{aligned} \dot{\mathbf{x}}(\mathbf{t}) &= f\left(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})\right) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\ \mathbf{u} \\ \mathbf{y}(\mathbf{t}) &= \mathbf{h}\left(\mathbf{x}(\mathbf{t})\right) = \mathbf{h}(\mathbf{x}) \end{aligned} \tag{1.3}$$

The number of inputs $\mathbf{u}$ and outputs $\mathbf{y}$ are both equal to m. The input $\mathbf{u}$ is replaced by a controller of the type $\mathbf{u} = \mathbf{r}(\mathbf{x}, \mathbf{v})$ with $\mathbf{v}$ being the new input, also of dimension m. The system state $\mathbf{x}$ has to be completely accessible for this. [13] points out that if this is not the case "an observer has to be implemented in a nonlinear context, which is a very difficult operation". In our case, we will later use a Kalman filter to estimate parts of the system, namely the currents. This will make all elements of the system state available.

Here, the vector $\mathbf{y}$ is considered as a vector of setpoint variables. For the transformation, we are interested in the derivatives $\mathbf{y}^{(k)}$ of $\mathbf{y}$, to the point where $\mathbf{u}$ is involved in the equation:

$$\mathbf{y^{(k)}} = \mathbf{A}(\mathbf{x}) \cdot \mathbf{u} + \mathbf{b}(\mathbf{x}) \tag{1.4}$$

with $\mathbf{k}$ denoting the number of times $\mathbf{y}$ has to be differentiated before $\mathbf{u}$ appears. Isolating $\mathbf{u}$ now requires the matrix $\mathbf{A}$ to be invertible. Note that in our case the dimension m is equal to 1, thus the matrix $\mathbf{A}$ is singular and can be inverted if it is different from 0.

$$\mathbf{u} = \mathbf{A^{-1}}(\mathbf{x})(\mathbf{v} - \mathbf{b}(\mathbf{x})) \tag{1.5}$$

Remark 1 *Most robots are redundant, meaning that they have more inputs than are needed to reach any state within the scope of their system dynamic. However,* $\dim \mathbf{u} > \dim \mathbf{y}$ *also leads to the matrix* $\mathbf{A}(\mathbf{x})$ *being rectangular. A pseudoinverse can be used for the transformation of* $\mathbf{A}(\mathbf{x})$. $\dim \mathbf{v}$ *will be equal to* $\dim \mathbf{y}$*, leading to a non-redundant linearized system.*

A linear system with m outputs and m inputs is thus acquired with $\mathbf{v}$ as the new input variable:

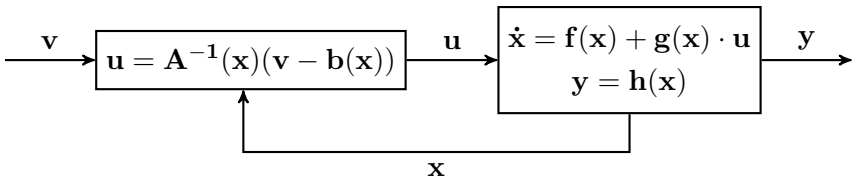


Figure 1.6: Transformed and decoupled nonlinear system, modified from [13].

The acquired system is linear and completely decoupled. This means that every element y_i of $\mathbf{y}$ depends on only one element v_i of $\mathbf{v}$. (It will be shown that in our case there is only one equation of this type, the dimension m being 1.) A PID-type controller can now be used to control each of the decoupled system equations.

1.5 Kalman-Filter

Boatbot was already in use as an autonomous control system [2], [3], [20]. An improved control law should increase the performance of the regulation in the context of this work. Among other things, this is to be achieved by compensating ocean currents and other statistical disturbances by the controller. Knowledge of these factors is required. To achieve this, a Kalman filter is used.

The Kalman filter, presented as "A New approach to Linear Filtering and Prediction Problems" [15], is designed to reduce errors in measurements and provide estimates of non-measurable system states. The prerequisite is that the values of interest can be described by a mathematical model, for example in the form of equations of motion. Also for the filter to work, information about system disturbances, measurement noise and covariances between system states are required. These are the basis of the estimation process. This estimation process can then be compiled into an online function, which makes it particularly well suited for use in robotics. Of course a Kalman filter has uses in many areas, including also the estimation of more abstract state variables.

Since its first conceptions the theory has been steadily expanded. The extended Kalman filter (EKF) allows state estimation with nonlinear system equations by linearizing over the estimate of the current mean and co-variance. Only this allows for unrestricted use in navigation. Therefore, non-linear state transmissions are performed, the covariances on the other hand evolve as linear matrices.

However, the extended Kalman filter (EKF) is difficult to implement, difficult to tune and only reliable for systems that are almost linear on the time scale of updates. Many of these difficulties arise from the use of linearization. The unscented Kalman filter was therefore developed as a method to propagate mean and co-variance information through nonlinear transformations [14].

It is shown later that in the scope of this work a linear Kalman filter is sufficient. The system the Kalman filter estimates has been reduced to only linear equations in chapter 2.5. The formulation of the linear Kalman filter shown here is based on [27] and [26].

A Kalman filter estimates state variables in a feedback process. The state at a given time is

estimated and then receives feedback in the form of measurements. These measurements are statistically noisy. This means that the measured values are ideally subject to a Gaussian distribution. which is the basis for calculating the co-variances and therefore the quality of the estimations. As such, the equations for the Kalman filter fall into two groups:

State evolution → Prediction

Here, the time updates of the process state take place. On the basis of the linear equations of state, an a priori estimate for state and error co-variance is calculated here. This step can be understood as a prediction. The equations below are the system update equation, co-variance update equation and filter matrix equations, all in discrete formulation (1.6).

$$\begin{aligned} \mathbf{x}_{\mathbf{k+1}}^{*} &= \mathbf{\Phi_k}\hat{\mathbf{x}}_{\mathbf{k}} + \mathbf{B_k}\mathbf{u_k} \\ \mathbf{P}_{\mathbf{k+1}}^{*} &= \mathbf{\Phi_k}\mathbf{P_k}\mathbf{\Phi}_{\mathbf{k}}^{\mathbf{T}} + \mathbf{D_k}\mathbf{Q_k}\mathbf{D}_{\mathbf{k}}^{\mathbf{T}} \\ \mathbf{S_{k+1}} &= \mathbf{C_{k+1}}\mathbf{P}_{\mathbf{k+1}}^{*}\mathbf{C}_{\mathbf{k+1}}^{\mathbf{T}} + \mathbf{R_{k+1}} \\ \mathbf{K_{k+1}} &= \mathbf{P}_{\mathbf{k+1}}^{*}\mathbf{C}_{\mathbf{k+1}}^{\mathbf{T}}\mathbf{S}_{\mathbf{k+1}}^{-\mathbf{1}} \end{aligned} \tag{1.6}$$

The matrix $\mathbf{S_{k+1}}$ is a substitution to reduce the length of the equation for $\mathbf{K_{k+1}}$.

Measurement → Correction, Innovation

The update of the measurements is responsible for introducing the current sensor data into the a priori estimate in order to obtain improved a posteriori estimates. This step can be understood as a correction [26] or innovation [27]. The equations (1.7) are the state correction equation, where measurement and prediction are combined and the co-variance correction equation, decreasing the state co-variance based on the quality of the measurements.

$$\begin{aligned} \hat{\mathbf{x}}_{\mathbf{k}} &= \mathbf{x}_{\mathbf{k}}^{*} + \mathbf{K_k}\left(\mathbf{y_k} - \mathbf{C_k}\mathbf{x}_{\mathbf{k}}^{*}\right) \\ \mathbf{P_k} &= \mathbf{P}_{\mathbf{k}}^{*} - \mathbf{K_k}\mathbf{C_k}\mathbf{P}_{\mathbf{k}}^{*} \end{aligned} \tag{1.7}$$

State vector		State transition	Control input	Estimation		Disturbance
Estimate	Prediction	matrix	matrix	covariance matrix		input matrix
$\hat{\mathbf{x}}_\mathbf{k}$	$\mathbf{x}_\mathbf{k}^*$	$\mathbf{\Phi_k}$	$\mathbf{B_k}$	$\mathbf{P_k}$	$\mathbf{P}_\mathbf{k}^*$	$\mathbf{D_k}$
Disturbnace		Measurement	Measurement	Kalman filter		Measurement
covariance matrix		input matrix	covariance matrix	matrix		vector
$\mathbf{Q_k}$		$\mathbf{C_k}$	$\mathbf{R_k}$	$\mathbf{K_k}$		$\mathbf{y_k}$

Table 1.1: Overview of the discrete matrices and parameters used in a Kalman filter.

The result is a prediction-correction algorithm for solving numerical problems. Usually, the prediction can be carried out at much higher frequencies than the correction. The rate of update for the correction on the other hand, is bound to the refresh-rate of the measurements, i.e. the sensors. Therefore, the state co-variance estimates $\mathbf{P}_\mathbf{k}^*$ usually increase while only predictions are carried out and is set back to much smaller values when a measurement arrives.

The algorithm of the Kalman filter as a dialog between prediction and correction now mainly deals with the steady adjustment of the filter matrix $\mathbf{K_K}$, which is essential for the weighting between measured values $\mathbf{y_k}$ and prediction values $\mathbf{x}_\mathbf{k}^*$, see (1.6). It is, in fact a matrix that puts a bias on the quality and therefore the reliability of measurement vs. prediction. If, for example, the measured values are very noisy, the prediction is weighted more heavily, while the measured values are of greater importance if the prediction is far from the measured values.

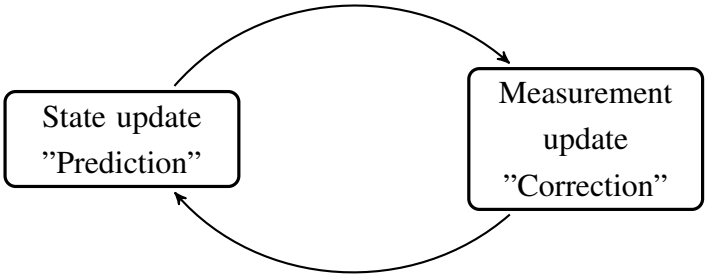


Figure 1.7: The cycle of prediction and correction in a Kalman filter, modified from [26].

$\mathbf{\Phi_k}$ is the discrete transition matrix that follows from the system matrix $\mathbf{A(t)}$ of the continuous system (1.8) after discretization. $\mathbf{G(t)}$ and $\mathbf{\Gamma(t)}$ are input matrices of the control signals $\mathbf{u(t)}$ and the disturbances $\mathbf{s(t)}$, which can be represented as white noise.

$$\dot{\mathbf{x}}(\mathbf{t}) = f\left(\mathbf{t}, \mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t}), \mathbf{s}(\mathbf{t})\right) = \mathbf{A}(\mathbf{t})\,\mathbf{x}(\mathbf{t}) + \mathbf{G}(\mathbf{t})\,\mathbf{u}(\mathbf{t}) + \mathbf{\Gamma}(\mathbf{t})\,\mathbf{s}(\mathbf{t})$$
$$\mathbf{y}(\mathbf{t}) = \mathbf{h}\left(\mathbf{x}(\mathbf{t})\right) + \mathbf{w}(\mathbf{t}) \tag{1.8}$$

The state variables $\mathbf{x}(\mathbf{t})$, which usually can not be measured directly, are are derived from the measurement equation $\mathbf{y}(\mathbf{t})$. The measured variables are subject to the measurement errors $\mathbf{w}(\mathbf{t})$. The measurement matrix $\mathbf{C_k}$ is the linearized representation of $\mathbf{y}(\mathbf{t})$.

From the known input matrices $\mathbf{G}(\mathbf{t})$ the discrete input matrix $\mathbf{B_k}$ follows from the discretization:

$$\mathbf{B_k} = \int_{\mathbf{0}}^{\mathbf{T}} \mathbf{\Phi}(\tau)\mathbf{G_k}\,\mathbf{d}\tau \tag{1.9}$$

The discrete input matrix $\mathbf{D_k}$ of the disturbances $\mathbf{s}(\mathbf{t})$ can be calculated in the same way from the input matrices $\mathbf{\Gamma}(\mathbf{t})$ of the statistical disturbances. The covariance matrix of the discrete disturbances then follows as:

$$\mathbf{Q_k} = \mathbf{D_k}\mathbf{Q'_k}\mathbf{D_k^T} \text{ and } \mathbf{Q'_K} = \mathbf{E}\{\mathbf{s_k}\mathbf{s_k^T}\} \tag{1.10}$$

For simple applications, where parameters can be adjusted quickly and are limited to a first-order model, such as the Kalman filter used on *Boatbot*, which will be shown in the following chapters, it is often sufficient to dispense with this derivation and rather estimate the co-variance matrices $\mathbf{Q_k}$ for the discrete noise and $\mathbf{R_k}$ for the measurement noise directly.

Thus, $\mathbf{R_k}$ is first derived from an analysis of the measurement method. Often the data sheets of the sensor manufacturers contain suitable information on accuracy. These can be further adapted in test procedures. However, this only makes sense if actually physically meaningful measurements are used. Using preprocessed data at the basis of the measurement model of a Kalman filter is a common mistake that often leads to erratic results as the error distribution of this data is usually not Gaussian [27].

Initialization also requires initial values for the co-variance matrix of the estimation errors $\mathbf{P_0^*}$. These can usually be very large, i.e. $\mathbf{P_0^*} \to \infty$. The size of the values in the co-variance matrix of the estimation errors $\mathbf{P_k^*}$ then automatically adjusts to the quality of the estimates $\hat{\mathbf{x}}_\mathbf{k}$. The

closer to $\mathbf{P}_k^*$, $\mathbf{P}_0^*$ is initially chosen at the beginning, the less time the Kalman filter will take to converge.

Meaningful diagonal values for the co-variance matrix $\mathbf{Q_k}$ of the discrete perturbations often follow from an error estimation of the individual state variables [27]. In practice, the ratio between the size of the values in $\mathbf{Q_k}$ and $\mathbf{R_k}$ decides on the dynamic of the filter. If the covariances of the perturbations are assumed to be larger, the Kalman filter responds with higher volatility as a response to changes while, on the other hand, noise is very well filtered out, i.e. a strong smoothing occurs when the covariances of the measurements are assumed to be larger.

1.6 Robot operating system (ROS)

ROS is an open source software framework which provides libraries and tools to build robots. A ROS software system is a system with subprograms called nodes performing individual tasks while messages are passed between them through topics. This can be represented in the form of a graph structure. There are more functions, such as services and parameters, but for the scope of this work, a basic understanding of nodes and topics is sufficient.

There is a background process called the ROS Master [19] which provides a decentralized process architecture by registering each node to itself and establishing the topics as peer-to-peer connections between them. Another initial component of ROS is the `rosbag_record` node, which creates the data logs that contain data needed for post-processing. The data used for the validation in chapter 4 is based on these logs.

Nodes

A process running within ROS is called a node. A node is an executable file within a ROS package [19] which is typically identified by a given name, although it is possible to create anonymous nodes with automatically generated identifiers. Nodes are at the core of programming with ROS, as most of the code used on a robot will be based around nodes receiving messages from topics, processing the data and passing it on to other nodes.

Nodes can also receive and send messages to and from peripheral software and devices. Drivers are usually available for the implementation of sensors, custom messages can also exported through the use of virtual serial ports.

Topics

Topics are named buses over which nodes send and receive messages. [19] This is done by creating publishers and subscribers inside of nodes and specifying the structure of the messages in the ROS framework, making it possible to use both predefined and user-defined message types. The content of these messages can be sensor data, motor control commands, state information, actuator commands, or anything else.

The system of ROS topics is anonymous, information about the publishing node is not incorporated into the message, i.e. a node which subscribes to a topic does not know where the information came from unless the programmer explicitly includes it into the message.

1.7 Structural Overview

This work will be divided into four parts. The first part will focus mainly on the theory of the control of an autonomous boat along a GPS-track. In the second part, the technical implementation will be discussed. Additions and optimization carried out as a part of this work, will be pointed out. The third part is an evaluation of the complete software and hardware setup based on simulations and deployed tests. In the final part, the acquired data gathered in the search for *La Cordelière* will be shown.

Chapter 2

Control Problem

2.1 Control of a boat in a vector field

A very simplistic representation of a boat is an entity which floats in the water and is capable of moving in any direction given that it has propulsion and it is maneuverable. While the first is a prerequisite for any sea-keeping entity, the other two are exclusive to controllable vehicles. In this work the entity is a rubber boat (Zodiac) with an outboard motor which provides propulsion as well as, by turning, maneuverability. We want to use these capabilities to make the boat follow a given track between two waypoints.

2.1.1 Restrictions

A boat can normally be described as a system with an input vector $\mathbf{u}(\mathbf{t}) \in \mathbb{R}^m$ with $m = 2$. This applies to a vessel where the turning rate and the speed are controllable. However, *Boatbot* is set up in a way that a controller can only control the turning rate of the boat, using a servo motor that controls the angle of the outboard motor, and not the speed. Thus, the system is restricted in a way that it can only use the turning of the motor as an input u. The propulsion of the motor is set to a constant value and can only be changed by a human operator.

2.1.2 Vector field control

When a vector is assigned to each point in a subset of a given space, this field is called a vector field. In a plane it can be visualized as a collection of arrows with a given magnitude and direction, each attached to a point in the plane. There are many applications in physics and mathematics. For instance, vector fields are commonly used to model the speed and direction of moving fluids or the distribution of forces in an area.

In the context of navigation, a vector field can be described as a plane with vectors inside an earth-fixed coordinate system, showing a desired direction and speed depending of the orientation and size of the vectors attached to individual points.

We define a vector field v in the space $\Omega \subset \mathbb{R}^2$ and assign a vector $v(x_1, x_2) \in \mathbb{R}^2$ to every point $(x_1, x_2) \in \Omega$, thus $v : \Omega \to \mathbb{R}^2$. In the case of *Boatbot*, we are only interested in the orientation of the vectors in this vector field, which is mapped to latitudinal and longitudinal coordinates and thus returns a desired direction at the given point. Thus, the control of *Boatbot* in a vector

field is the control of it's orientation in relation to the direction of the vectors comprising the vector field.

Here, we want to follow a straight line. For the control in relation to a line of undefined length, only the orientation of *Boatbot* and the offset to the line are relevant. Thus, a function can be chosen for the generation of the vector field v that returns a desired direction ψ for an offset d. In this work the offset d usually aligns with the x_2 coordinate. Note that this is only the case as long as the line to be followed and the x_2 axis are identical. On *Boatbot* a coordinate transformation for the controller was implemented to account for this problem.

$$\psi : \mathbb{R} \to \mathbb{R}, x_2 \mapsto \psi\left(x_2\right) \tag{2.1}$$

A set of restrictions further specifies the desired function. The angle which is returned for $x_2 = 0$ has to be 0. An entity on the line should head in the direction of the line to follow it.

$$\psi\left(x_2 = 0\right) = 0 \tag{2.2}$$

The angles which are returned for rising values of x_2 - both in negative and positive direction - have to rise equally and continuously. The further the entity is from the line the more it should change its heading towards the line in order to reach it.

$$\psi\left(-x_2\right) = -\psi\left(x_2\right) \tag{2.3}$$

In an east-north-up coordinate frame (ENU) this means that ψ has to rise for negative values of x_2 and vice versa. Also, when an entity is heading towards the line at 90 degrees, this is the maximum angle the function should reach, both in positive and negative directions. A greater angle would mean the entity is heading in the opposite direction of the line to be followed.

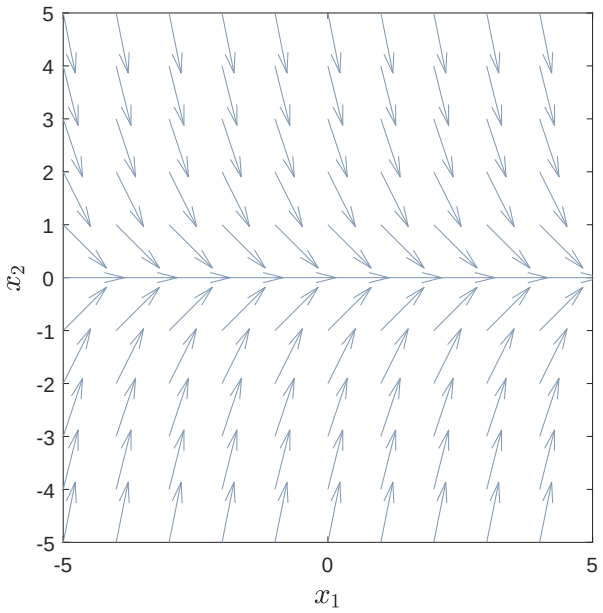
$$\begin{aligned} \lim_{x_2 \to \infty} \psi(x_2) &\geq -\tfrac{\pi}{2} \\ \lim_{x_2 \to -\infty} \psi(x_2) &\leq \tfrac{\pi}{2} \end{aligned} \tag{2.4}$$

Two such function are the inverse tangent function and the hyperbolic tangent function (2.5) which are both used in this work:

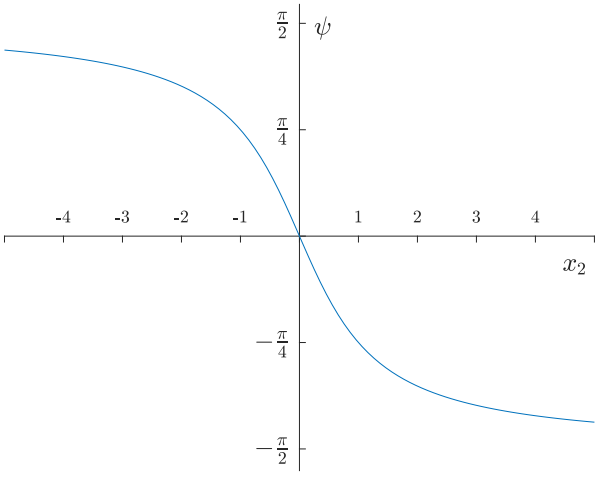
$$f(x_2) = -\arctan(x_2)$$
$$f(x_2) = -\tanh(x_2) \tag{2.5}$$

These are the functions that will be used to derive the controllers later. It is apparent that the x_1 coordinate is meaningless in this context. The orientation of the vectors in the vector field v results solely from the function $\psi(x_2)$, d respectively, the offset to the track in the real system. The vector fields v can be drawn as:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \cos(-\arctan x_2) \\ \sin(-\arctan x_2) \end{pmatrix}$$
$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \cos(-\tanh x_2) \\ \sin(-\tanh x_2) \end{pmatrix} \tag{2.6}$$



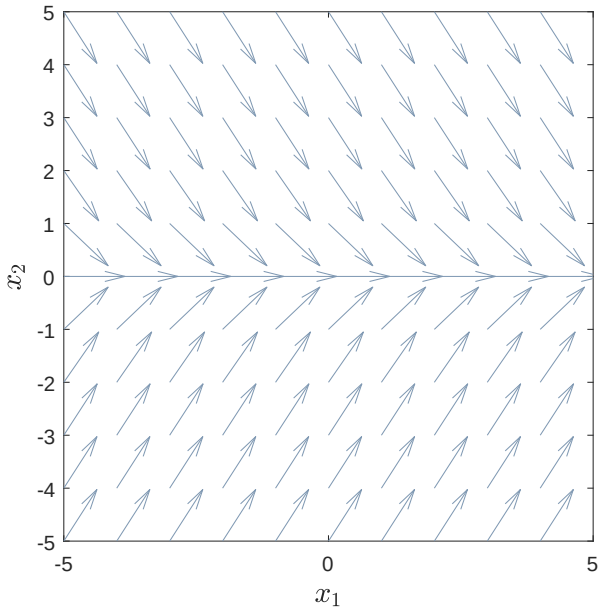
(a) Arctan vector field



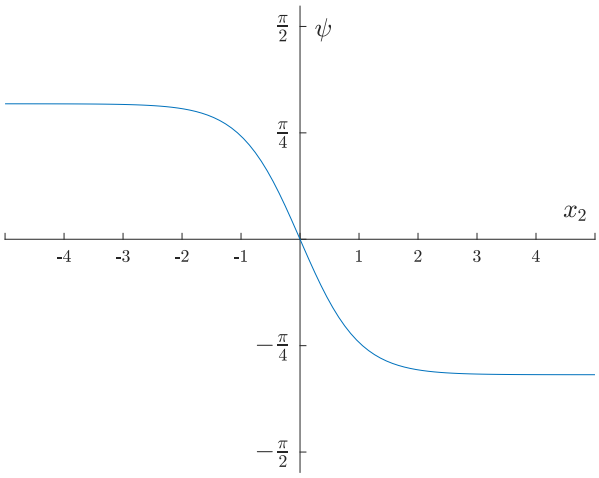
(b) Arctan function curve

Figure 2.1: Curve and vector field of an inverse tangent function.

The inverse tangent function, as in figure 2.1, has a natural incidence of $90°$ but the transition to $0°$ is much smoother as compared to the hyperbolic tangent function in figure 2.2. For *Boatbot* the hyperbolic tangent function has been finally used because it is easier to modify the smoothness of the orientation than the incidence angle. In fact, it is easy to implement either.



(a) Tanh vector field



(b) Tanh function curve

Figure 2.2: Curve and vector field of an hyperbolic tangent function.

2.1.3 Path following

It has been shown in [16] that for a system of the form

$$\begin{aligned} \dot{\mathbf{x}}(\mathbf{t}) &= f\left(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})\right) \\ \mathbf{y}(\mathbf{t}) &= \mathbf{h}\left(\mathbf{x}(\mathbf{t})\right) \end{aligned} \tag{2.7}$$

with an input vector $\mathbf{u} = (u_1, ..., u_m)$ and a state vector $\mathbf{x} = (x_1, ..., x_n)$, where $\mathbf{y}$ is considered a pose vector $\mathbf{y} = (y_1, ..., y_{m+1})$ with $n \geq m + 1$ it is possible to find a controller that follows a vector field in the $\mathbf{y}$ space.

Here, for the dimensions of the system, we have $m = 1$, one control input $\mathbf{u}$, which is the rudder angle. $n = 3$ is the number of state variables $\mathbf{x}$, namely x_1, x_2 and x_3, which are position and heading. The pose vector $\mathbf{y}$ is of dimension $m + 1 = 2$. It consists of the position (x_1, x_2), measured by GPS.

It is pointed out that it is possible to control $m + 1$ state variables and not only m of them, due to the fact that a path following is performed and not a trajectory tracking, where the time is Involved. Thus, we will not control $\mathbf{y}$ to be equal, but co-linear to the vector field v.

2.2 Mathematical model of the boat

The motion of a vehicle is usually described in Cartesian coordinates. Thus the equations of motion are derived in the (x_1, x_2)-plane. For any advanced control method, an accurate and stable mathematical model of the process to be controlled has to be found.

It is in our interest to find a simple controller that can easily be transported to any boat, thus the models we choose here are strongly simplified. The most important feature is the reduction from a 6-DOF system to a 3-DOF system. Heave, pitch and roll are discarded. The models only use the components in the x_1 and x_2 coordinate frame, yaw, sway and surge.

The purpose of this chapter is to find a mathematical model which can be used to describe a boat both as a basis for a controller and also to simulate it in the form of (2.7).

$$\dot{\mathbf{x}}(\mathbf{t}) = f\left(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})\right) \tag{2.8}$$

with

$$\mathbf{x}(\mathbf{0}) = \mathbf{x_0},\ \mathbf{u}(\mathbf{t}) \in \mathbb{R}^m,\ \mathbf{x}(\mathbf{t}) \in \mathbb{R}^n \tag{2.9}$$

We will then find controllers which are capable of controlling $m + 1$ state variables to perform a path tracking, the first of which will be an illustration of the PDsin controller that has been used on *Boatbot* in previous years.

2.2.1 Dubin's car

We consider an entity, here a boat, moving in a (x_1, x_2)-plane. For a basic model, we want to be able to describe this boat's position and orientation. The simplest mathematical representation of a boat is a Dubin's car. It is a first order model with $n = 3$ state variables, position and orientation, and $m = 1$ input variable, the rudder.

$$\mathbf{u}(\mathbf{t}) \in \mathbb{R}^1,\ \mathbf{x}(\mathbf{t}) \in \mathbb{R}^3 \tag{2.10}$$

Here, x_1 is defined as the boat forward direction, e.g. the direction of surge. x_2 corresponds to the direction of sway and x_3 is the heading in mathematical coordinates $\rightarrow$ ENU.

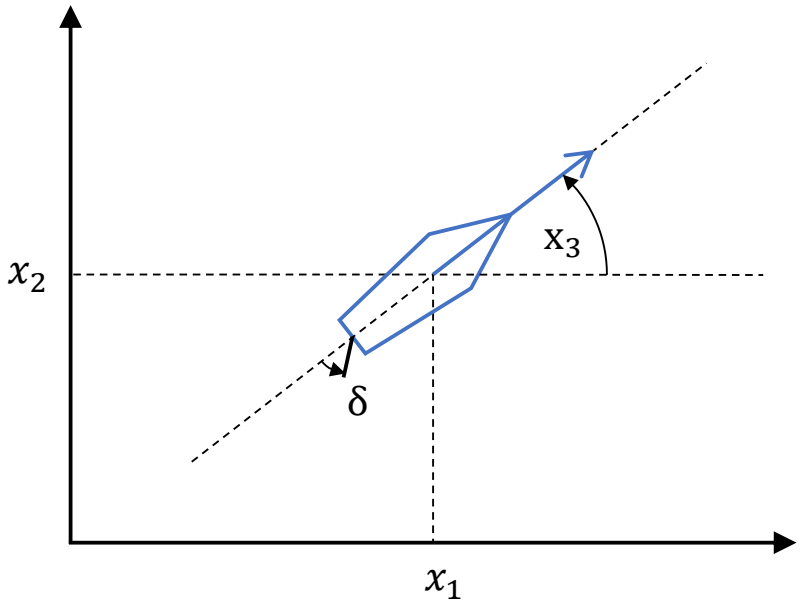


Figure 2.3: Boat in a (x_1, x_2)-plane

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} \cos x_3 \\ \sin x_3 \\ u \end{pmatrix} \tag{2.11}$$

Rather than a target angle for the rudder, u gives us the turning rate of the boat directly. The model assumes that for a given control input u the motor or rudder instantaneously jumps to the designated angle.

According to (2.7), we have a pose vector $\mathbf{y}$ of dimension $m + 1$ which consists of x_1 and x_2, which we want to control with $\mathbf{u} = u$ of dimension $m = 1$. Here, the pose vector $\mathbf{y}$ corresponds to the position of the center of the boat.

2.2.2 Sea currents and speed over water

Adding currents and speed over water to the model requires a state equation of the following form:

$$\dot{\mathbf{x}}(\mathbf{t}) = f\left(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t}), \mathbf{p}(\mathbf{t})\right) \tag{2.12}$$

Introducing currents to the system requires us to distinguish between true speed, defined by $\dot{x}_1$ and $\dot{x}_2$ and speed over water, $\dot{p}_1$. On the boat, the speed over water cannot be measured directly,

whereas the true speed corresponds to the speed measurements of the GPS.

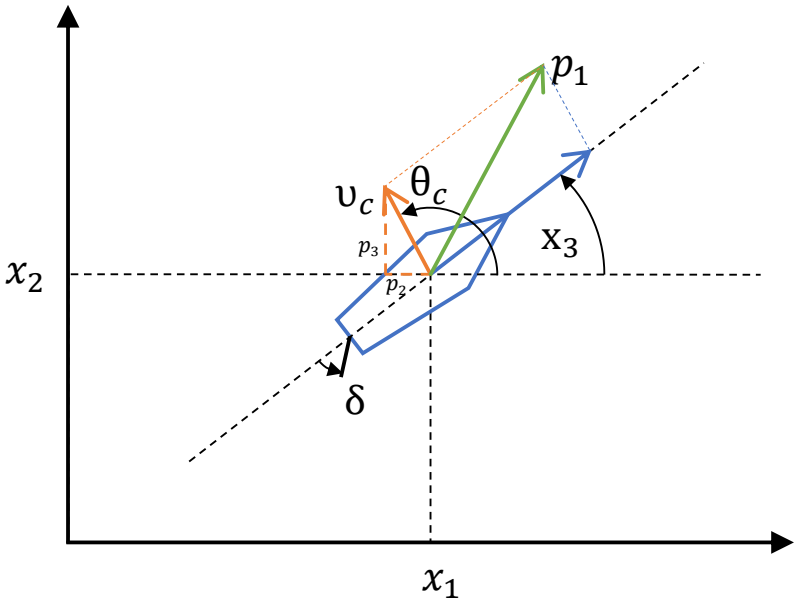


Figure 2.4: Boat with additional currents, separating heading x_3 and true course.

The currents and speed over water can also be considered as additional state variables, more in line with (2.12), rather than the constants as which they are shown in (2.13), thus it would be possible to consider $\mathbf{p}$ as a part of the state vector $\mathbf{x}$ and thus the dimension of the state vector $n = 6$. This does not change anything for the controller, which retains $m = 1$ and wills still be able to control $m + 1 = 2$ states, namely the pose x_1 and x_2. But as the currents and speed over water are unknown they have to be described as system states in order to estimate them as shown in chapter 2.5.

Currents are divided into components in x_1 and x_2 direction, p_1 and p_2 respectively. Note that a controller based on this model will always converge to $x_2 = 0$. This will require a coordinate transformation of the currents from a global coordinate frame to a local coordinate frame of the line to be followed later.

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} p_1 \cos x_3 + p_2 \\ p_1 \sin x_3 + p_3 \\ p_1 u \end{pmatrix} \tag{2.13}$$

In the model (2.13) the input u corresponds directly to the angle of the rudder or in our case motor δ. This impacts the motor in such a way that the turning rate $\dot{x}_3$ of the boat is directly

affected by the input u without latency, which would arise from the rudder having to slowly turn before it induces any forces. This is one of the main inaccuracies of this model.

Also, while a real boat can produce a self induced sway, the model only allows sway through currents. As this is the model used for the derivation of the controllers, resulting effects can be seen in chapter 4.1.

2.2.3 Second order model

Most of the aforementioned inaccuracies can be reduced by using a second order model for the boat. More specifically, this means, adding accelerations for the linear velocities $\dot{x}_1$ and $\dot{x}_2$ and a simple turning rate model for $\dot{x}_3$.

Simplified second order model where δ corresponds to the angle of the motor and L corresponds to the length of the boat:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x_4} \\ \dot{x_5} \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ \frac{\sin\delta}{L/2}\left(x_4 \cos x_3 + x_5 \sin x_3\right) \\ p_1 \cos x_3 + p_2 - x_4 \\ p_1 \sin x_3 + p_3 - x_5 \end{pmatrix} \tag{2.14}$$

with δ being a function of the control input u. Whereas $\dot{x}_3$ is the turning rate of the entity, the control input u does not translate directly to this turning rate, since it takes some time for the motor or rudder to turn to the angle set by the input variable.

$$\delta : \mathbb{R} \to \mathbb{R}, u \mapsto \delta\left(u\right) \tag{2.15}$$

The system shown in (2.14) is very strongly simplified. The equations for $\dot{x}_4$ and $\dot{x}_5$ for the acceleration in the x_1 and x_2 directions correspond to the equations of (2.14) with a normalized damping term, but have no evident physical meaning.

2.3 PD control problems

For the boat, in the past, a PD type controller has been chosen with a sinus on the derivative term. It was originally designed to follow an inverse tangent vector field with an incidence angle of 70 degrees. It has been modified here to follow a hyperbolic tangent function for the purpose of comparison with the new controller. The controller shown here has been modified to work

inside the east-north-up (ENU) reference frame.

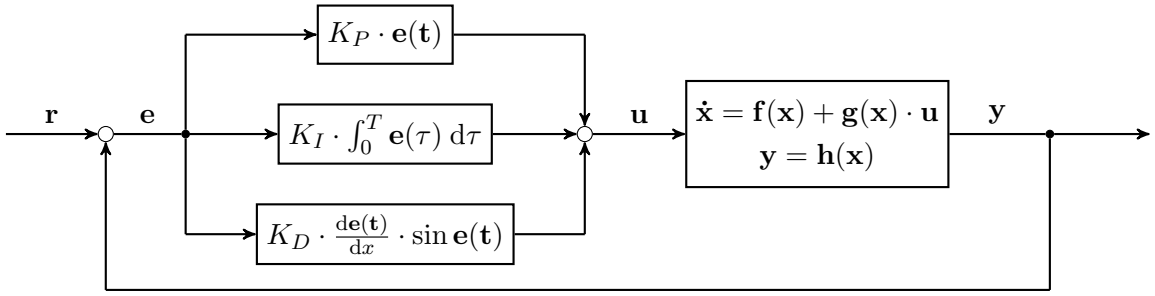


Figure 2.5: PIDsin controller, the feedback here is the heading, provided by the IMU.

First, a setpoint $\mathbf{r(t)}$,the target course ψ, which has been defined in the last section, is calculated based on the angle of the line to be followed φ and the distance to that line d. A parameter r_d specifies the "smoothness" of the vector field:

$$\mathbf{r(t)} = \psi = \varphi + \tanh(d/r_d) \tag{2.16}$$

$$\mathbf{r(t)} = \psi = \tanh(x_2/r_d) \tag{2.17}$$

Note that (2.16) only applies when φ is different from 0 and the line is not the x_2 axis. In that normalized case, which is the basis for the control equations in this work, we get (2.17).

Next, an error function $\mathbf{e(t)}$ is defined. This is the difference between the current heading of the boat x_3 and the target heading defined by the vector field ψ:

$$\mathbf{e(t)} = \psi - x_3 \tag{2.18}$$

This error function closes the control loop. We get a direct feedback controller with a proportional gain of 1. After adding a derivative term with a sinus we get the PDsin controller previously used on *Boatbot*:

$$\begin{aligned} \mathrm{P} &= K_P\ (\psi - x_3) \\ \mathrm{D} &= K_D\ \sin\left(\tfrac{z-1}{z}\ (\psi - x_3)\right) \\ \mathbf{u(t)} &= u = \mathrm{P} + \mathrm{D} \end{aligned} \tag{2.19}$$

The gains that have been used on *Boatbot* previously are $K_P = 1$ and $K_D = 0.2$. This controller does not follow a given path optimally but rather compensates an error, here an offset, after it is created, i.e. it does not take the curvature of the vector field into account. This is a classical feedback control law. The main problem of this controller is the overshoot it creates in this manner when it controls the boat to approach a path. A simulation has been carried out in **Matlab** with a Dubin's car model. Note that the rudder angle has been limited to 15 degrees to get a closer representation of the real system:

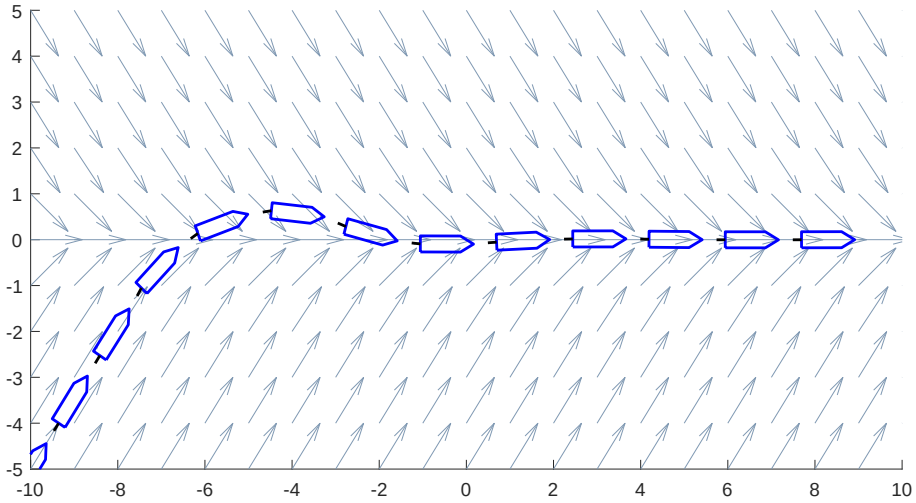


Figure 2.6: Dubin's car approaching the line with a PDsin controller on a hyperbolic tangent vector field as in equation (2.5).

Of course, this behaviour could be improved by adjusting the gains and smoothing the vector field, which has been done on *Boatbot*, but a better controller which avoids this problem can be found, as shown in the next section. Another problem of this controller is that it does not compensate for systematic errors, i.e. currents. For this purpose an integrator could be introduced and the controller thus expanded to a PIDsin controller. Again, a better way is to find a controller that is capable of compensating known currents which are in turn estimated by using a Kalman filter.

There are also some inherent flaws in using a PID type controller for this problem. The most apparent is the adaptivity. The PID controller works best in the environment in which it has been tuned. This can work well in a simulation. But if, for an instance the velocity of the boat or outside disturbances change, it will not work as well anymore.

2.4 Feedback linearization

In this section a feedback linearization method is used to demonstrate how a controller for a boat in a vector field can be obtained. Subsequently, the method is used to find first a simple controller and later. The problems of such a controller for our control purpose are shown and an improved controller is obtained.

2.4.1 Field following controller

To obtain a field following controller for the Dubin's car model, a feedback linearization is carried out. A setpoint for the control output has to be chosen. We want the output of the system to converge to 0. This means we want the boat to be heading in the same direction as the vector field.

$$y = x_3 + \arctan x_2 \tag{2.20}$$

This y can be interpreted as a control error. Now we have to find a controller for which y converges to 0. Differentiation of (2.20) yields:

$$\dot{y} = \dot{x}_3 + \frac{\sin x_3}{1 + x_2^2} = u + \frac{\sin x_3}{1 + x_2^2} \tag{2.21}$$

Note that (2.21) contains $\dot{x}_3$ which is defined as u in the Dubin's car model. This allows us to choose a first order equation for the error y. This closes the feedback loop.

$$\dot{y} = -y \tag{2.22}$$

Isolating u yields an equation which satisfies the error equation after inserting (2.22) in (2.21):

$$u = -y - \frac{\sin x_3}{1 + x_2^2} \tag{2.23}$$

Finally, a line following controller is obtained, after inserting (2.20) in (2.23):

$$u = -x_3 - \arctan x_2 - \frac{\sin x_3}{1 + x_2^2} \tag{2.24}$$

In the same manner, a controller can also be obtained for a hyperbolic tangent vector field. The controller is:

$$u = -x_3 - \tanh x_2 - \frac{\sin x_3}{\cosh(x_3)^2} \tag{2.25}$$

2.4.2 Offset due to currents

There is a problem with the controller we have just obtained, that is an offset which is generated by systematic errors. The most prevalent of these errors is sea currents and waves, and to a latter degree also wind and inaccuracies in the implementation of the robot. Currents generate a static offset when a simple field following controller like (2.24) or (2.25) is used, which is demonstrated in the following example of a Dubin's car with currents.

Let us take for the speed over water and the currents a vector $p = \left(\begin{smallmatrix} 1 \\ 0 \\ 0.2 \end{smallmatrix}\right)$. We assume that we are on the line, thus $x = \left(\begin{smallmatrix} 0 \\ 0 \\ 0 \end{smallmatrix}\right)$. The boat is facing in the direction of the line and has no offset. The controller (2.24) gives an output of 0 and thus the currents create an offset in the direction of x_2.

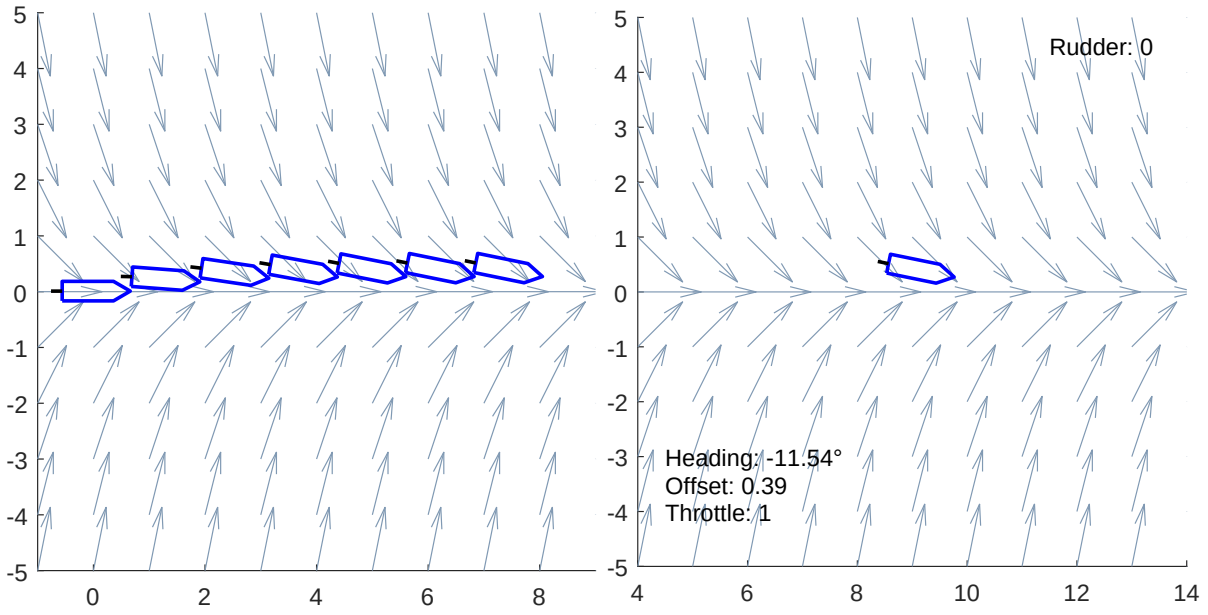
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} p_1 \cos 0 + p_2 \\ p_1 \sin 0 + p_3 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 + 0 \\ 0 + 0.2 \\ 0 \end{pmatrix} \tag{2.26}$$

The equilibrium of forces from currents and control output is reached when $\dot{x}_2 = 0$, thus

$$\begin{gathered} \sin x_3 + 0.2 = 0 \\ x_3 = \arcsin -0.2 = -11.54^\circ \end{gathered} \tag{2.27}$$

The offset can be computed from the control equation. Here, u can be set to equal 0, thus $\dot{x}_3 = u = 0$. This means that the turning rate of the boat is 0 when the equilibrium state is reached. The following equation can be solved numerically.

$$\begin{gathered} u = -x_3 - \arctan x_2 - \tfrac{\sin x_3}{1+x_2^2} \\ 0 = 11.54^\circ - \arctan x_2 - \tfrac{\sin(-11.54^\circ)}{1+x_2^2} \Rightarrow x_2 \approx 0.39 \end{gathered} \tag{2.28}$$



(a) Boat drift due to currents

(b) Equilibrium state

Figure 2.7: Equilibrium of a Dubin's car following a vector field with currents when not taking the currents into consideration

The state equation updated with the currents states at equilibrium shows that the impact on the velocity in direction of the line is small.

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} \cos(-11.54°) \\ \sin(-11.54°) + 0.2 \\ u \end{pmatrix} \approx \begin{pmatrix} 0.98 \\ -0.2 + 0.2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.98 \\ 0 \\ 0 \end{pmatrix} \tag{2.29}$$

In this example the effects of currents on a boat using a vector field controller without taking the currents into consideration, have been demonstrated. Consequently, a controller which compensates the effects of the currents has to be found.

2.4.3 Currents compensating controller

The derivation of the controller for the currents compensating controller is similar to the controller for the Dubin's car model. Again, a setpoint for the control output has to be chosen. We want the output of the system to converge to 0. Here, the heading will not be identical with the true course due to the currents. Following the line perfectly does not necessarily mean that the boat has to be heading in that direction. As soon as there is an x_2-component in the currents, this is in fact impossible with a finite speed.

Instead we want the true course to align with the direction of the vector field. The true course can be represented by the inverse tangent function of $\dot{x}_1$ and $\dot{x}_2$. Also, this time, we choose to use the hyperbolic tangent function as the target function for the vector field control:

$$y = \arctan\left(\frac{\dot{x}_2}{\dot{x}_1}\right) + \tanh x_2 = \arctan\left(\frac{p_1 \sin x_3 + p_3}{p_1 \cos x_3 + p_2}\right) + \tanh x_2 \tag{2.30}$$

Differentiation of the control error (2.30) yields:

$$\dot{y} = \frac{\dot{x}_1\ddot{x}_2 - \dot{x}_2\ddot{x}_1}{\dot{x}_1^2 + \dot{x}_2^2} + \frac{\dot{x}_2}{\cosh(x_2)^2} \tag{2.31}$$

With the substitutions

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} = \begin{pmatrix}p_1 \cos x_3 + p_2\\ p_1 \sin x_3 + p_3\end{pmatrix} = \begin{pmatrix}a(x)\\ b(x)\end{pmatrix} \tag{2.32}$$

$$\begin{pmatrix}\ddot{x}_1\\ \ddot{x}_2\end{pmatrix} = \begin{pmatrix}-p_1\dot{x}_3 \sin x_3\\ p_1\dot{x}_3 \cos x_3\end{pmatrix} = \begin{pmatrix}-p_1^2 \sin x_3\\ p_1^2 \cos x_3\end{pmatrix} u = \begin{pmatrix}\partial a(x)\\ \partial b(x)\end{pmatrix} u \tag{2.33}$$

we get:

$$\dot{y} = \underbrace{\frac{a \cdot \partial b - b \cdot \partial a}{a^2 + b^2}}_{\mathbf{a(x)}} \cdot u + \underbrace{\frac{b}{\cosh(x_2)^2}}_{\mathbf{b(x)}} \tag{2.34}$$

Now, u can be isolated, thus completing the feedback linearization.

$$u = \mathbf{a}(\mathbf{x})^{-\mathbf{1}}(\dot{y} - \mathbf{b}(\mathbf{x})) \tag{2.35}$$

This allows us to choose a first order equation for the error y:

$$\dot{y} = -y \tag{2.36}$$

We obtain the controller (2.37) which consists of the error function and substitutions carried out in this chapter:

$$u = \mathbf{a}(\mathbf{x})^{-\mathbf{1}}(-y - \mathbf{b}(\mathbf{x})) \tag{2.37}$$

This controller is now able to compensate currents provided the speed over water p_1, i.e. the propulsion of the boat is great enough to move the boat significantly faster than the currents.

2.5 Currents and speed over water estimation

2.5.1 Mathematical model

Including the currents and speed over water to our mathematical model adds three unknown variables to our model. These can described as state variables additionally to position and heading. Assuming that the speed over water and the currents are constants, We get an expanded nonlinear system for the Dubin's car:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \end{pmatrix} = \begin{pmatrix} p_1 \cos x_3 + p_2 \\ p_1 \sin x_3 + p_3 \\ p_1 u \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.38}$$

We now want to use a Kalman filter to estimate p_i. A Kalman filter will allow us to estimate system states using an algorithm that observes a series of measurements over time. Therefore, a measurement equation which contains our system states is needed, in order to get a system of the form:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\ \mathbf{u} + \mathbf{g}'(\mathbf{x})\ \mathbf{s} \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}) \end{aligned} \tag{2.39}$$

with $\mathbf{s}$ as the vector of continuous disturbances on the plant. The equations (2.38) and (2.39) belong to a nonlinear system. In order to use a basic Kalman filter, a linear system is needed. In this case, estimating x_i is not necessary, because these states are known. Thus, we choose a Kalman Filter that only estimates p_i. The system can be described in the form:

$$\begin{aligned} \dot{\mathbf{p}}(\mathbf{t}) &= \mathbf{A}(\mathbf{t})\ \mathbf{p}(\mathbf{t}) + \mathbf{G}'(\mathbf{t})\ \mathbf{s}(\mathbf{t}) \\ \mathbf{y}(\mathbf{t}) &= \mathbf{h}(\mathbf{p}(\mathbf{t}), \mathbf{x}(\mathbf{t})) \end{aligned} \tag{2.40}$$

Note that the vector $\mathbf{x}$ consists of the elements x_i now and remains part of the system dynamic. Currents (in longitudinal and latitudinal direction) and speed over water are independent of the heading of the boat, thus the system is also independent of the input u. The model for $\dot{x}_1$ and $\dot{x}_2$ is then used as the new measurement equation. The sensor data which will be used here is the GPS-speed, which is the true speed described by $\dot{x}_1$ and $\dot{x}_2$:

$$\begin{pmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$
$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} p_1 \cos x_3 + p_2 \\ p_1 \sin x_3 + p_3 \end{pmatrix} \tag{2.41}$$

Also, we assume the unknown system disturbances only change slowly, i.e. there is no direct correlation between the system disturbances and our first order model, allowing us to choose the matrix $\mathbf{G}'$ as a matrix of zeros.

Note that the system states estimated by the Kalman filter will be referred to as $\mathbf{p}$ in the following to avoid confusing currents and speed over water with the position and heading of the boat. In most literature it is common to use $\mathbf{x}$ as the vector of system states.

2.5.2 Discretization

In the scope of robotics, a Kalman filter is usually implemented as an online estimation process in the software of the robot. It is therefore implemented as a discrete algorithm working within set time steps. This makes it necessary to discretize the system dynamic and find a Kalman filter of the form.

Innovation

$$\begin{aligned} \hat{\mathbf{p}}_\mathbf{k} &= \mathbf{p}^*_\mathbf{k} + \mathbf{K}_\mathbf{k} \left(\mathbf{y}_\mathbf{k} - \mathbf{C}_\mathbf{k} \mathbf{p}^*_\mathbf{k} \right) \\ \mathbf{P}_\mathbf{k} &= \mathbf{P}^*_\mathbf{k} - \mathbf{K}_\mathbf{k} \mathbf{C}_\mathbf{k} \mathbf{P}^*_\mathbf{k} \end{aligned} \tag{2.42}$$

Prediction

$$\begin{aligned} \mathbf{p}^*_{\mathbf{k+1}} &= \mathbf{\Phi}_\mathbf{k} \hat{\mathbf{p}}_\mathbf{k} \\ \mathbf{P}^*_{\mathbf{k+1}} &= \mathbf{\Phi}_\mathbf{k} \mathbf{P}_\mathbf{k} \mathbf{\Phi}^\mathbf{T}_\mathbf{k} + \mathbf{D}_\mathbf{k} \mathbf{Q}_\mathbf{k} \mathbf{D}^\mathbf{T}_\mathbf{k} \\ \mathbf{S}_{\mathbf{k+1}} &= \mathbf{C}_{\mathbf{k+1}} \mathbf{P}^*_{\mathbf{k+1}} \mathbf{C}^\mathbf{T}_{\mathbf{k+1}} + \mathbf{R}_{\mathbf{k+1}} \\ \mathbf{K}_{\mathbf{k+1}} &= \mathbf{P}^*_{\mathbf{k+1}} \mathbf{C}^\mathbf{T}_{\mathbf{k+1}} \mathbf{S}^{\mathbf{-1}}_{\mathbf{k+1}} \end{aligned} \tag{2.43}$$

The measurement matrix $\mathbf{C}_\mathbf{k}$ is:

$$\mathbf{C}_\mathbf{k} = \begin{pmatrix} \cos x_3 & 1 & 0 \\ \sin x_3 & 0 & 1 \end{pmatrix} \tag{2.44}$$

The transition matrix $\mathbf{\Phi}$ can be obtained from the system dynamic matrix $\mathbf{A}$ through the Laplace transformation:

$$\mathbf{\Phi(T)} = \mathcal{L}^{-1}\left[\mathbf{sI} - \mathbf{A}\right]^{-1} = \mathcal{L}^{-1}\left[\begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{pmatrix}\right]^{-1} = \mathcal{L}^{-1}\begin{pmatrix} 1/s & 0 & 0 \\ 0 & 1/s & 0 \\ 0 & 0 & 1/s \end{pmatrix} = \mathbf{I} \tag{2.45}$$

In this context our understanding of the continuous disturbances is limited, thus we will not be able to find an accurate mathematical representation of $\mathbf{D_k}$ and therefore also $\mathbf{Q_k} = \mathbf{D_k Q'_k D_k^T}$ which represents the co-variance matrix of discrete disturbances. In the technical implementation it is entirely sufficient to provide a rough estimate for this matrix.

The co-variance matrix of discrete disturbances:

$$\mathbf{D_k Q_k D_k^T} = \begin{pmatrix} 0.000001 & 0 & 0 \\ 0 & 0.000001 & 0 \\ 0 & 0 & 0.000001 \end{pmatrix} \tag{2.46}$$

The co-variance matrix of measurements:

$$\mathbf{R_k} = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix} \tag{2.47}$$

The initial co-variance matrix the state estimations:

$$\mathbf{P_0^*} = \begin{pmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{pmatrix} \tag{2.48}$$

Chapter 3

Implementation

The idea of *Boatbot* is to have an integrated solution of an automatic data acquisition tool. It is, when deployed with all hard- and software able to not only follow a given path but also gather magnetic data in the process and provides the possibility of online monitoring. The capabilities of *Boatbot* can be divided into two main categories which are when deployed, completely independent from each other:

Robot
: The boat with motor, steering servo, control setup and navigation sensors. This is the part of *Boatbot* which follows a given path using a feedback linearization controller.

Magnet sensor setup
: The sensor setup for data acquisition is completely decoupled from the boat itself. A magnetometer is towed under water; the data is written to an isolated computer.

Note that GPS-data acquired by the robot is later used in an offline process when merging the track of *Boatbot* with the magnetic data gathered. This will be expanded on in chapter 4.4.

The responsibility of **ENSTA** is limited to an autonomous zodiac capable of properly scanning an area for a long time [2]. This opens up the possibility for other sensors to be used on the setup, as long as they have their own geo-location, synchronization and recording capabilities.

In the following chapter the software and hardware setup for both the robot and the magnet sensor setup are explained in depth.

3.1 Robot

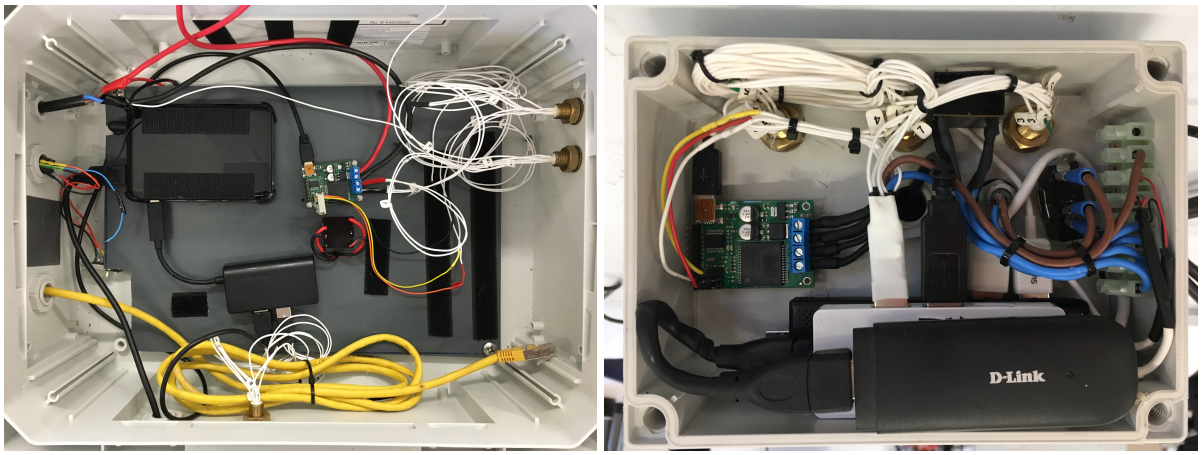
A robot is “a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer” [8]. That is, in the case of *Boatbot* navigation and control along lines defined by a set of GPS-waypoints. These actions are carried out automatically by an integrated system of sensors, processors and actuators. In this chapter, the technical application of the system at hand will be explained.

3.1.1 Hardware

The hardware on the boat can be roughly divided into three categories. There are the sensors and peripherals, which pass the information they receive on to an embedded computer system which processes the data according to the scheme of guidance, navigation and control and give control commands to the actuator as well as giving feedback to the peripherals.

Some of the hardware that was finally used on the robot had already been in use in previous *Boatbot* projects. An earlier version used a different IMU (**Razer**), which was replaced in 2018 because it was not self-calibrating and less accurate than the one now in use. Also the hardware setup of the embedded computer was completely overhauled.

Two boxes containing embedded computer systems are in use during the project, a big one, which has been used in previous years, and a small one which is a recent and improved version. During the month of June, 2019, most data was acquired using the big box while the small box was not yet in a working state. This was mainly caused by bugs on the embedded computer of the small box which had not been previously encountered using the embedded computer in the big box.



(a) Big box (old) (b) Small box (new)

Figure 3.1: Size comparison of the old and new embedded computer systems.

3.1.1.1 Sensors

Boatbot is equipped with a GPS-receiver and an IMU for position and heading. GPS-data used for the control system includes latitudinal and longitudinal position, x_1 and x_2, as well as true speed and course over ground. GPS-timestamps are logged and later used for merging the magnetometer data with the positional data. The only data used from the IMU is the heading, which is computed directly within the driver.

IMU (Compass)

The IMU used on *Boatbot* is an **Ellipse 2-A** manufactured by **SBG Systems**. It includes 3 axis gyroscopes, accelerometers and magnetometers, as well as temperature sensors [21]. The primary interest for *Boatbot* is the z-axis magnetometer which is used to measure the heading of the boat. All data acquired by the sensors is preprocessed in an internal CPU. See also 3.2.

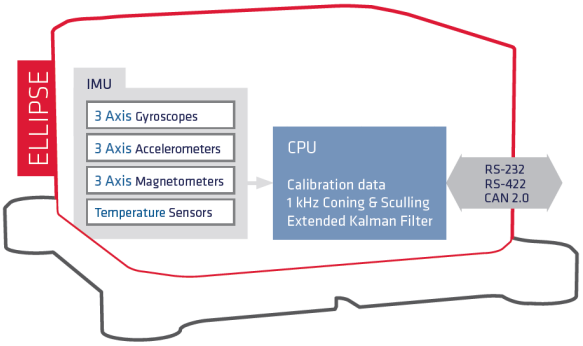


Figure 3.2: **SBG Ellipse 2-A** inertial sensor, modified from [21]

The **Ellipse 2-A** has one main data port which connects to a USB port on the embedded computer of *Boatbot* directly. Magnetometer data is transferred as Euler angles.

RTK GPS

The GPS chip used is a **NEO-M8P-2** RTK GPS made by **u-blox** [25]. It is placed on a **DP0501** board sold by **Drotek** [9]. It is equipped with Real Time Kinematic (RTK) technology which allows it to be combined with another static GNSS module as a rover and thus greatly improve it's accuracy. In the context of *Boatbot*, such RTK corrections are usually received through an online service.

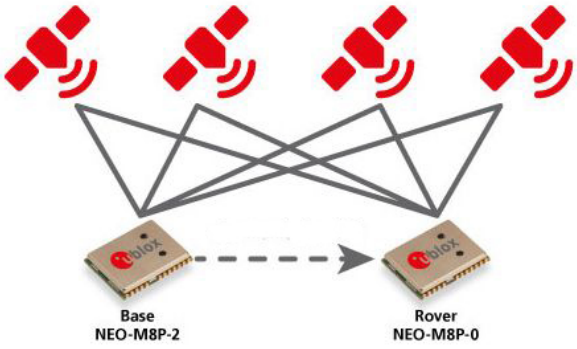


Figure 3.3: RTK principle [25], *Boatbot* uses a **NEO-M8P-2** as a rover

Data used from the GPS include positioning in longitudinal and latitudinal degrees, GPS-speed and course, and a GPS timestamp. GPS-speed and course are later converted to speeds in latitudinal and longitudinal direction in the **ROS**-driver The GPS module is also plugged in to the embedded computer via a USB port. Information is sent through NMEA-sentences.

Remark 2 *NMEA sentences are a standard for communication between navigation devices on ships, defined by the National Marine Electronics Association (NMEA) and also used for communication between GPS receivers and PCs as well as mobile devices [17]. An example for an NMEA sentence used in navigation is given below. There are many different message types. The positions marked with numbers are different for every message type.*

a b c 1 2 3 4 5 6 7 (...) n d e
`$--GSV,x,x,x,x,x,x,x,...*hh<CR><LF>`

Field Number: a) Message begin marker, b) Device ID, c) Message type, 1) Data record begin, 2) Message number, 3) Satellites in view, 4) Satellite number, 5) Elevation in degrees, 6) Azimuth in degrees to true, 7) SNR in dB, ...) more satellite infos like 4-7, n) Checksum, d) Carriage return e) Line feed

These are all the sensors used on *Boatbot* on which positioning and control are based. Additionally, RTK-corrections for the GPS are technically received via a separate internet stick, but they are then incorporated into the GPS-data and don't make a difference to the functionality of the control, although they do improve the accuracy of the GPS.

Method	Sensor	Obs	Accuracy	Update rate
Positioning	u-blox NEO-M8P-2	x_1, x_2	$0.025 - 2.5\ m$	$5 - 10\ Hz$
Speed	u-blox NEO-M8P-2	$\dot{x}_1, \dot{x}_2$	$0.05\ ^m/_s$	$5 - 10\ Hz$
Heading	u-blox NEO-M8P-2	$\dot{x}_1, \dot{x}_2$	$0.5°$	$5 - 10\ Hz$
Heading (Compass)	SBG Ellipse2-A	x_3	$0.8°$	$200Hz$

Table 3.1: Sensor characteristics from manuals provided by manufacturers. Note that accuracy and update rate of the GNSS sensor is heavily dependent on the availability of satellites (GPS,GLONASS,etc.) and the availability of RTK-corrections.

The sensors are both placed on an aluminum rod at the side of the helm of *Boatbot*. The GPS was placed on the top and the IMU at the bottom. Note that both sensors are not placed at

the center of the boat. This is also not mathematically compensated. This should not have an impact on the IMU as only the magnetic data for the heading is used. For the GPS, an additional systematic error of $\sim {}^{1}/_{2}m$ should be expected. This error is neglected.

(a) IMU (Compass) (b) RTK GPS

Figure 3.4: The sensors are place on a rod beside the helm of the boat.

3.1.1.2 Embedded Computer System

The embedded computer system is the main processing unit of the robot. It is contained in a box attached to the front of the helm of *Boatbot* and houses all programmable hardware components.

USB Motor Controller

The embedded computer system houses the motor controller **Polulu Jrk 12v12** from **Polulu Corporation**. It has an operating range from 6 V to 16 V and a continuous output current of 12 A (30 A peak) which allows it to control the servo used to turn the steering wheel [18].

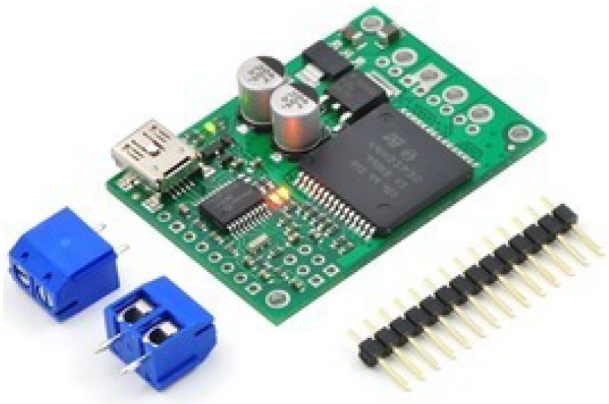


Figure 3.5: **Polulu Jrk 12v12** controller board, from [18]

It connects to the embedded computer via a USB Mini-B port and controls the servo with the values from 0 to 4095 it receives as an input from the **ROS** driver in **zodiac_auto**.

Embedded Computer

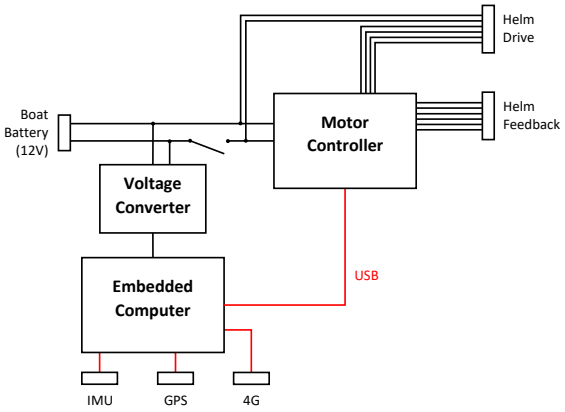
The embedded computer is an **Intel Compute Stick STK1A32SC**. It runs on **Ubuntu 16.04 Xenial (LTS)** and has two USB ports which are used to connect the **Polulu Jrk 12v12** and a USB hub connecting IMU, GPS and a internet stick from **D-Link**. **ROS** runs on top of **Ubuntu** among smaller software packages, the most important of which will be pointed out in the next section.

Voltage Converter

A voltage converter is used to convert the 12 V from the battery of the boat to the 5 V required for the compute stick.

GPS, IMU can be separately connected to the box with waterproof plugs, as well as a command (Helm Drive) and a feedback line (Helm Feedback) for the steering servo. A manual switch is located at the front of the box which allows the user to switch between manual and automatic control of the steering wheel. A 4G-internet-module was added to allow for the reception of RTK-corrections. See figure 3.6.

(a) New embedded computer housing



(b) Connection chart

Figure 3.6: Embedded computer system of *Boatbot*

Previously, a much bigger box had been in use with slightly different hardware. It has been upgraded during the preparations for *Submeeting 2019* and has been in use since then. However, most of the data from *Submeeting 2019* was still gathered using the old embedded computer system. Apart from the hardware of the main processing unit the hardware changes mainly drastically reduced the size of the complete setup.

3.1.1.3 Actuators

Usually, a boat would be described as a system with two inputs $\mathbf{u(t)}$, namely heading control and speed control. As already pointed out earlier, the control setup only uses u as an input to control the motor angle and therefore the heading. This is also due to the restriction that *Boatbot* is only equipped to allow for automatic control of the steering wheel. In fact, the capability of automatic speed control has not been added, because it is unnecessary for the type of mission *Boatbot* performs which require a relatively stable speed over water.

Steering servo

A steering servo is located inside the helm behind the steering wheel of the boat. A switch controlling the servo, located on the box housing the embedded computer, disables and enables the servo and locks the steering wheel wheel when automatic control is enabled. The servo takes commands from 0 to 4095 which correspond to the range from maximal rudder angle from starboard $\sim (-25°)$ to port $\sim 25°$.

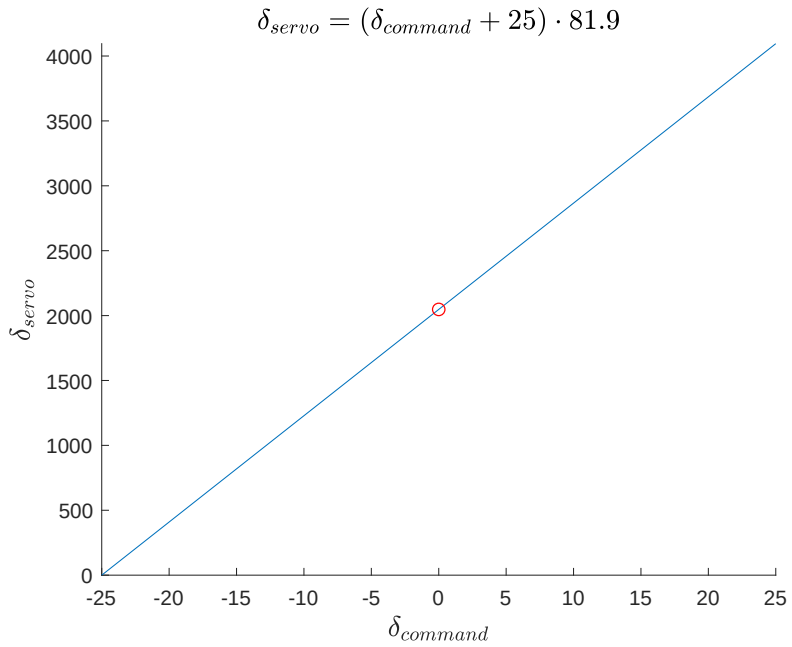


Figure 3.7: Transformation function of the steering servo.

This transformation function is a part of the **ROS**-node `motorCtr` in the package `zodiac_auto`. A bias can be added through a configuration file.

Speed lever

Although the speed is very important to the control system as a whole, a way to extend the controllability of the system to the propulsion has not yet been added. *Boatbot* still relies on manual speed control. During a mission, speed is usually set to a constant value of around four knots, speed over ground.

3.1.1.4 Peripherals

A laptop connects to a wireless network originating from the embedded computer inside the box. This laptop is used to monitor the mission and to set GPS-waypoints in a navigation software called **OpenCPN**. Since the embedded computer system is equipped with an internet stick, it is also possible to monitor or set waypoints from any device connected to the internet.

3.1.2 Software

The software used for the control of *Boatbot* can be mainly divided into three categories. There is the **Robot Operating System (ROS)** which serves as the main data processing and distribution hub. Some additional software serves to connect the data streams from **ROS** to sensors and peripherals. A peripheral Laptop uses **OpenCPN** and **rqt** to monitor the process. A comprehensive overview of the main software architecture is given in 3.8.

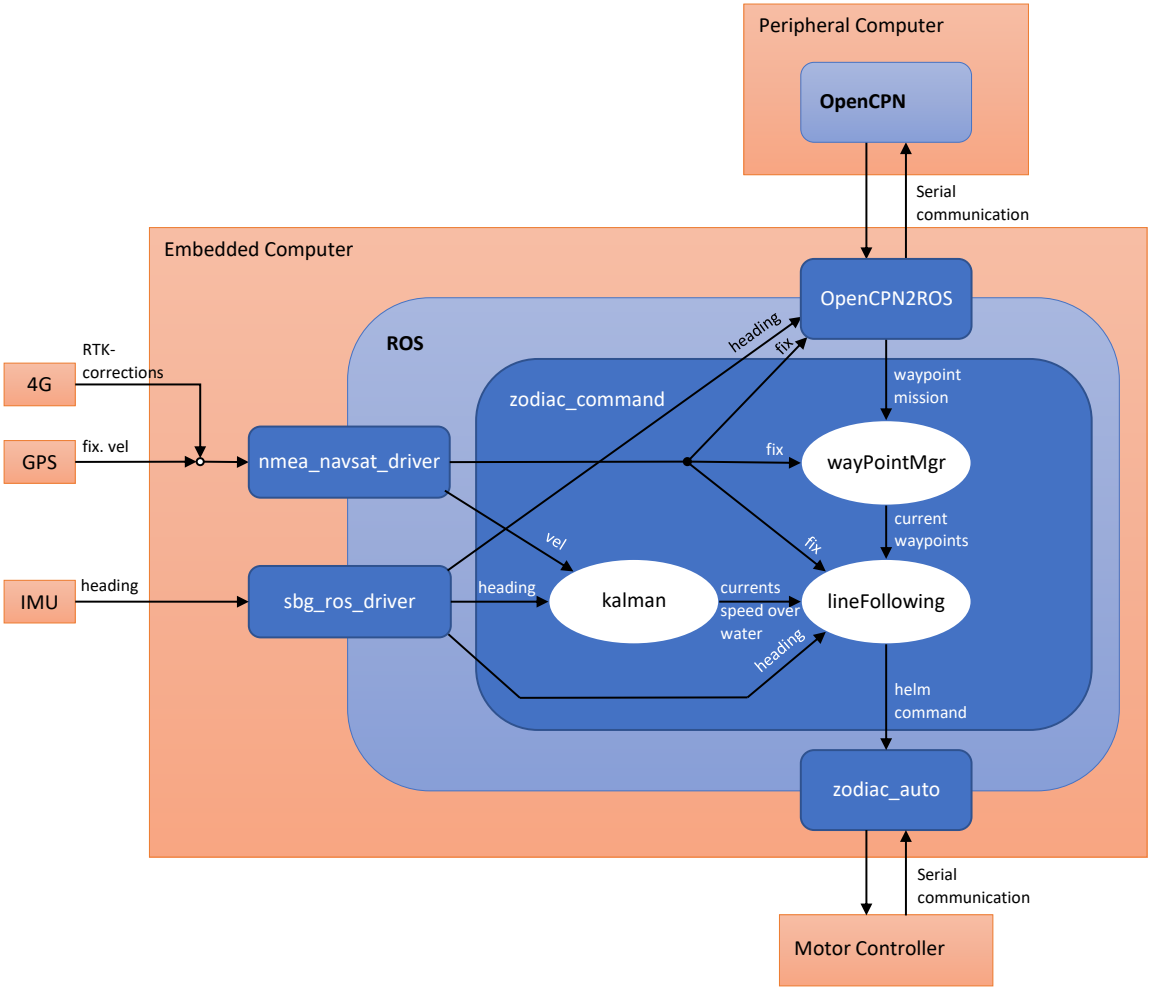


Figure 3.8: Software Architecture

3.1.2.1 Robot operating system (ROS)

On top of *Ubuntu*, most of the data processing and control is done by **ROS**. **ROS** is a software framework for robots in which multiple subprograms can be run simultaneously as ”nodes”. These nodes subscribe to and publish messages as ”topics” for other nodes to use. Messages can also be read from and written to serial ports to allow for hard- and software to be incorporated outside of **ROS**.

The **ROS** system for *Boatbot* consists of five sets of nodes. These are the drivers for IMU and GPS, the communicator nodes between **ROS** and the **Polulu Jrk 12v12** board, as well as between **ROS** and the peripheral computer running **OpenCPN** and the command nodes. Additionally, configuration and launchers have been extracted.

Note that the following table only illustrates sets of nodes. Individual nodes can be identified in the figure 3.8.

nmea_navsat_driver

Language: Python 3

Documentation: `http://ros.org/wiki/nmea_navsat_driver`

This package contains the driver for the GPS sensor. It publishes the GPS-data topics `fix`, `time_reference`, `gps_speed` and `gps_course`.

OpenCPN2ROS

Language: Python 3

Documentation: `https://img.shields.io/badge/ROS-Kinetic--Kame-green.svg`

This package creates a link between **OpenCPN** and **ROS** throughout a serial port using the NMEA standard. It subscribes to the `nmea_sentence` topic which contains GPS and IMU data and publishes `nmea_sentence_opencpn`, and `new_waypoint_mission` which contains the waypoints the robot wants to follow.

sbg_ros_driver

Language: C++

Documentation: `https://github.com/SBG-Systems/sbg_ros_driver/blob/master/src/ellipse.cpp`

This package contains the driver for the inertial measurement unit (IMU). It publishes the IMU-data topics `imu_data` which contains the linear accelerations and angular velocities and `ekf_euler` which contains the angles yaw, pitch and roll. The only value used in this work is the yaw, contained in `ekf_euler`, which is measured by an integrated magnetic compass.

zodiac_auto

Language: C++

Documentation: `https://github.com/EnstaBretagneClubRobo/ZodiacAuto`

This package contains the driver for the **Polulu** control board which sends actuation signals to the steering servo. A transfer function, see 3.7, maps the message in the `helm_angle_ cmd` topic to the actuation values of the servo.

zodiac_command

Language: C++

Documentation: `https://github.com/EnstaBretagneClubRobo/ZodiacAuto`

This package the state estimation and control nodes for *Boatbot*. Most of the programming in the project was done inside this package. There are four nodes, `kalman` which provides the estimation of currents and speed over water, `wayPointMgr` which passes the two current waypoints to `lineFollowing` where the controller computes the controller output. `mathUtility` is not a node but contains mathematical functions used in the other nodes. An overview of subscribers and publishers of these nodes can be found in the table below 3.8.

zodiac_launchers

Language: C++

Documentation: `https://github.com/EnstaBretagneClubRobo/ZodiacAuto/zodiac_launchers`

This package contains launchers and scripts for the whole software package. All nodes and virtual serial ports are launched here.

The main control nodes for *Boatbot*, i.e. the nodes which contain the controller and estimator setup shown the **Control Problem** chapter, are contained in **zodiac_command**. The following table 3.2 gives a comprehensive overview of the functionality of the nodes contained.

The mathematical controller is contained in the node `lineFollowing`. The only topic it publishes which is used by another node is the `helm_angle_cmd`. All other topics published by this node are only for the purpose of surveillance and logging. The `kalman` node contains the Kalman filter which is used to estimate the currents and speed over water which are published in the `currents` topic along with their estimated errors. It also publishes the `boat_heading` topic. The node `wayPointMgr` takes the GPS position from `fix` and `new_waypoint_mission` in the NMEA-format from **OpenCPN** as an input and constantly checks if the next waypoint has been reached as well as publish the two current waypoints.

Node	wayPointMgr	kalman	lineFollowing
Subscribers	fix new_waypoint_mission	vel ekf_euler imu_data	fix waypoint_line currents boat_heading
Publishers	waypoint_line status_waypoint_mission	currents boat_heading	signed_distance line_angle desired_course helm_angle_cmd
Frequency	...	100 Hz	10 Hz

Table 3.2: Nodes in `zodiac_command`

3.1.2.2 Peripheral

As already pointed out, a peripheral computer is used to monitor the automation of *Boatbot* and to set missions in the from of sending waypoints. As on the embedded computer, **socat** is also used here to create virtual serial ports. These are needed to establish a connection between **OpenCPN** and the **ROS** program running on the embedded computer.

OpenCPN is an open source GPS-navigation software by **OpenCPN.org** used here to monitor *Boatbot* on a map in real time, as well as send missions in the form of waypoints. Some additional features used are importing and exporting waypoint missions and GPS-tracks, monitoring the rudder angle and the true speed. It is also possible to see currents when displaying both GPS-course and the heading provided by the IMU (compass).

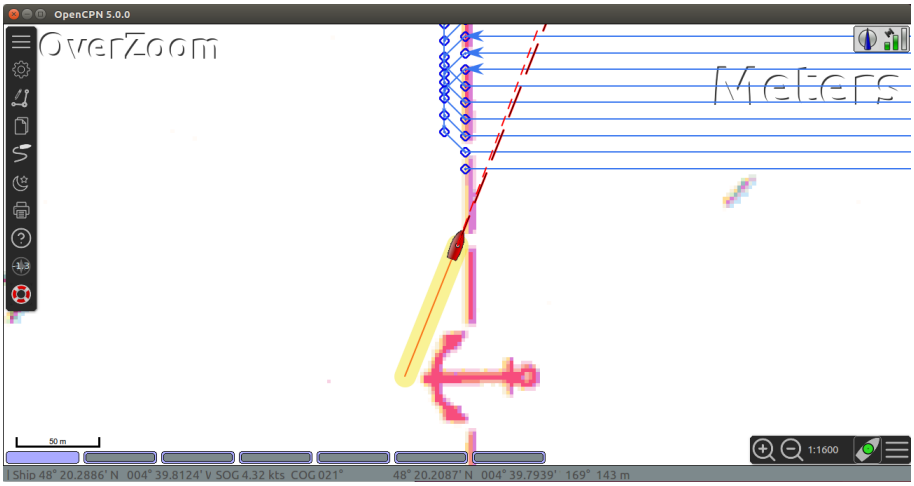


Figure 3.9: The OpenCPN interface

Monitoring the **ROS** processes was carried out either by using the **ROS** subprogram **rqt** or by

using the terminal command `rostopic echo`. This was only possible by installing **ROS** and all the nodes running on the embedded computer on the peripheral device as well. Normally, installing only **ROS** on the peripheral device would be sufficient, but custom messages have been used, which are only available if the corresponding nodes are available and built on the peripheral device.

While using `rostopic echo` is uncomplicated and easy to use when monitoring one topic, **rqt** brings more functionality and troubleshooting capabilites. It is also easier to monitor multiple topics at once.

3.2 Magnet sensor setup

As *Boatbot* has been conceptualized as a data mapping robot, a sensor has to be added. The sensor data is only used for offline post processing and has no impact on the control, thus it is possible for it to be completely independent from the robot. In the case of *Boatbot*, magnetic data is acquired with a single towed magnetic sensor. In this chapter the hard- and software setup of the sensor will be shown and some difficulties pointed out. The main idea is to tow a magnetometer at a set depth under the water surface and by extension also to be capable of estimating its depth.

3.2.1 Hardware

The data acquisition comprises of a Magnetometer, which is inserted into a tube. The tube has four fins at its bottom to increase stability in the water. The Magnetometer is connected to a Depressor at a distance of seven meters. The depressor is needed for the Magnetometer to stay under water while the boat is moving. It is shaped to allow water to flow under it more easily than over it, thus increasing the pressure on its upper side pushing it downwards. This results in a sensor depth of approximately seven to eight meters at a deployment speed of three to five knots, when the length of the rope between the attachment on *Boatbot* and the depressor is 12.5 meters for the provisional setup. (The length was limited by the cable connecting the magnetometer with the onboard hardware.) This setup was used in all the experiments that were carried out during the month of June, 2019. See figure 1.3.

During *Submeeting 2019* the depth of the magnetometer was crudely estimated by estimating the angle between the rope connecting the frame of the boat with the depressor and the horizon, thus estimating α. To reduce the load on the frame of the boat, a rope was added between the bow and the frame, which was tightened depending on the speed over water when the depressor was deployed. The force on the frame coming from the depressor also was the main limitation on the speed of *Boatbot*.

The sensor used on *Boatbot* is a **FGM3D** fluxgate magnetometer from **Sensys**. It is a magnetic field sensors and measures in three axes x, y, z [22]. As a system to monitor or control the orientation of the magnetometer was not the main purpose of this work, usually values from all three axes were normalized. The analogue data is then passed on to a **FGM3D TD** analogue to digital converter from **Sensys**, see 3.10.

Figure 3.10: The **FGM3D TD** magnetometer data logging set

The Magnetometer is connected to an analogue to digital converter on *Boatbot*, which is connected to a computer via USB. The analogue to digital converter is powered by an external battery.

3.2.2 Software

A software from **SENSYS** completes the **FMG3D** packa isand used to log the incominge g data, which consists of magnetic measurements in x, y and z directions, as well as a GPS-timestamps form a built in GPS receiver. This data is consecutively merged with the GPS-readings from *Boatbot* in an offline process. This will be expanded upon in Chapter .

Chapter 4

Validation

For the validation of the new control law, one brique (brique 8) was selected as a basis for comparing the performance of the old and new controllers, both in simulation and in a series of tests on the sea. Simulations have been performed in **Matlab** in a normalized system and in **UxVSim** which uses the same software as *Boatbot*, simulating only sensors and system response. The usability of the data acquisition hardware also will be evaluated in this chapter.

4.1 Path following

The validation process of the control algorithm aims to showcase that the new controller, which is a feedback linearization based proportional controller taking currents into account, performs better than a more simple PD-based controller with a sinus on the derivative term.

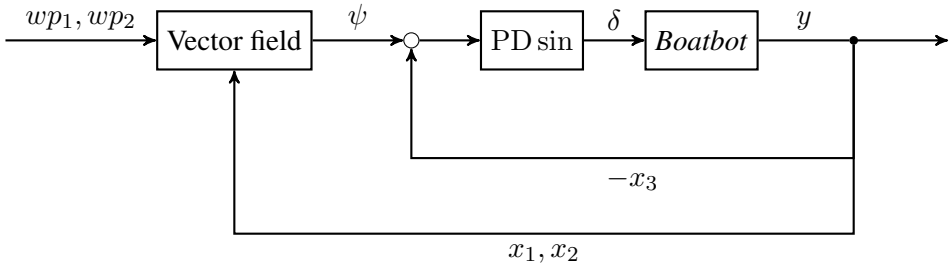


Figure 4.1: The old controller is a PDsin controller using a vector field for static setpoints.

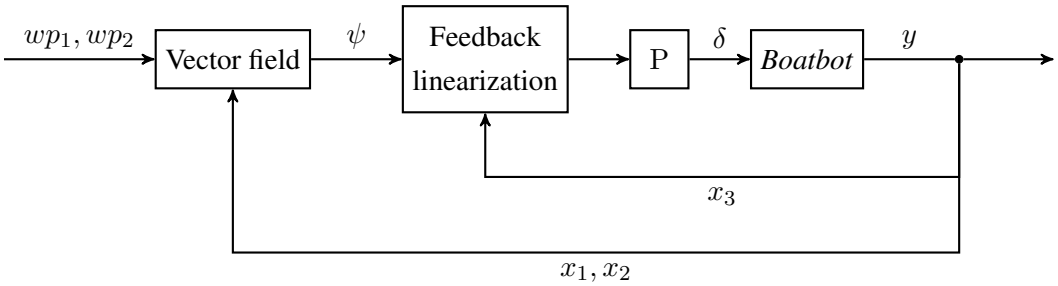


Figure 4.2: The new controller is a feedback linearization based proportional controller. A vector field acts as a setpoint input.

In the first part of the path following validation a **Matlab** simulation will compare the results from both controllers in a normalized environment. Thus, the basic functionality of both controllers can be showcased and expectations on the behaviour of the real system formulated.

Of course, the simulations are much simplified in comparison to a deployed system. They use a 3-DOF model which also doesn't take environmental conditions like waves or wind into account. Therefore, the validation would not be complete without adding a comparison of *Boatbot* being deployed on the sea using both the old and new controllers. Therefore, the second part of this validation compares the performance of the controllers in two real experiments.

4.1.1 Simulation

These simulations were carried out using **Matlab**. The Dubin's car model which is a first order dynamic model for the boat, with and without currents is used for this purpose, as well as a second order model, taking longitudinal and lateral accelerations into account. This allows us to demonstrate how the controllers work in a perfect environment. The control equations are also normalized. This means there are no additional parameters and all gains are set to 1.

4.1.1.1 Converging to the line

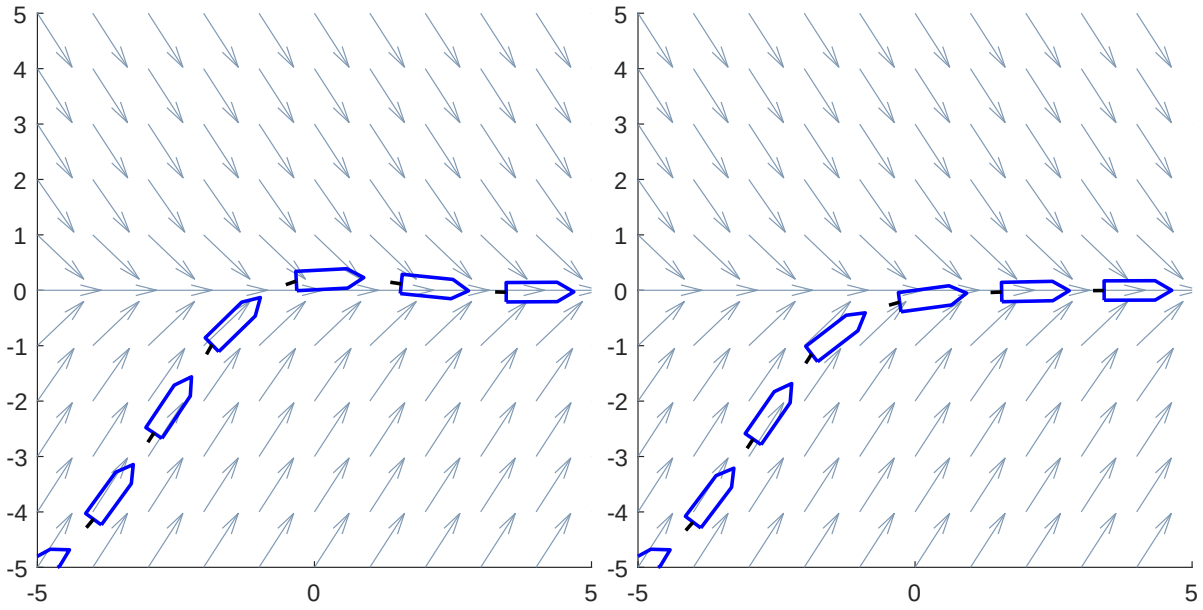
For the first simulation the parameters were chosen to represent an idealized environment on the sea. This means there are neither currents nor statistical errors on the positional and heading data. The table 4.1 shows the parameters that were used in the simulation. The boat starts south of the line, heading northeast.

Dynamic model	Speed over water	Currents	
		Speed	Direction
Dubin's car	$1.0\ ^{m}/_{s}$	$0.0\ ^{m}/_{s}$	...

Table 4.1: First simulation parameters

The objective of this simulation is to show that the new controller follows the vector field perfectly, because it inherently aligns the heading of the boat with the desired direction. The old controller is based on the error between the heading of the boat and the desired heading, thus reacts only after an error has already been created.

This corresponds to the normal operating principle of the proportional part of a PID controller. In this case, it is already known how the desired course will develop though, the vector field being known, which is why it makes sense to use a controller that uses this information. This additional information used by the new controller can be roughly described as taking the curvature of the vector field into account.



(a) Old controller (b) New controller

Figure 4.3: The new controller follows the vector field perfectly.

In this instance, the boat using the new controller fulfills the requirements of anticipating the curvature of the vector field and thus turning so that the true course aligns with the direction of the vector field and then converges to the line.

To avoid overshoot the vector field at the basis of the old controller could be smoothed out or the gains of the PDsin controller adjusted. On *Boatbot* some optimization regarding these parameters has been done to improve the performance of the PDsin controller. It stays inferior to the new controller though in that it cannot anticipate the curvature of the vector field.

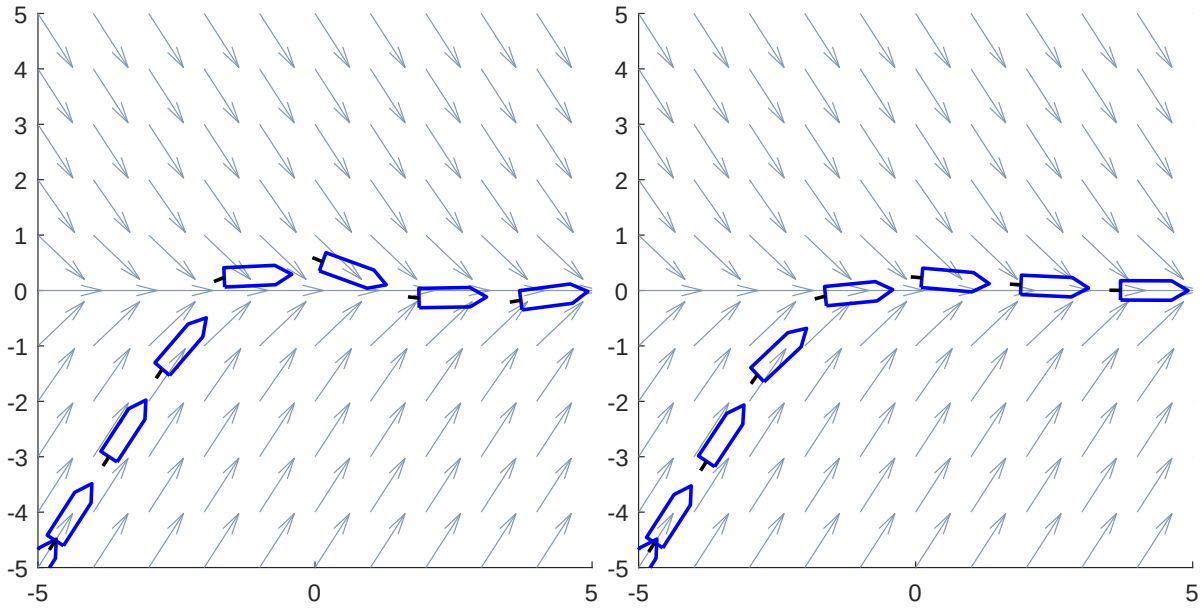
For the second simulation a second order model was used. It now includes a model for accelerations in x_1 and x_2 direction. Therefore, the speed over water in table 4.2 is now a desired speed which is affected from deceleration at turning.

Dynamic model	Speed over water	Currents	
		Speed	Direction
Second order	1.0 $^m/_s$	0.0 $^m/_s$	...

Table 4.2: Second simulation parameters

As the model used in this case has acceleration also in lateral directions, both controllers should perform less well than with the Dubin's car. Both controllers are derived from Dubin's car and

should therefore be unable to compensate for anything not present in the model.



(a) Old controller (b) New controller

Figure 4.4: Converging to the line in a second order model.

As expected, this time, the new controller also creates an overshoot, resulting from sideways drift. The error for the old controller has also increased. Eliminating this error would require us to build a controller on top of the second order model or any other model that is more accurate than Dubin's car. But for the scope of this work, the performance of the controller obtained, is sufficient.

4.1.1.2 Offset compensation

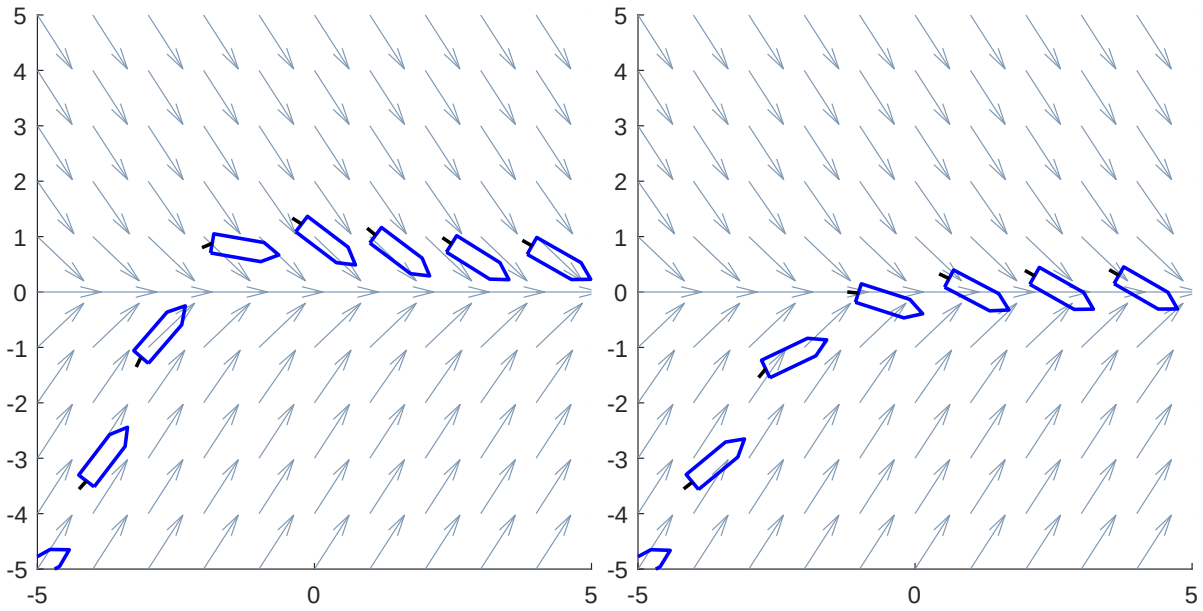
A third simulation has been performed with **Matlab**. Now currents have been added, which can, within limits, also showcase the influence of winds and waves and other statistical disturbances.

Dynamic model	Speed over water	Currents	
		Speed	Direction
Dubin's car	1.0 m/s	0.5 m/s	90° (ENU)

Table 4.3: Third simulation parameters

Firstly, the simulation should prove that the boat using the old controller approaching the line to be followed generates an offset to the line after crossing it or without ever reaching it, depending on the what side of the line the boat starts on. The new controller will quickly converge to the

line instead when currents and speed over water are estimated correctly. The results of these tests are shown in figure 4.5.



(a) Old controller

(b) New controller

Figure 4.5: Currents make it impossible for the old controller to converge.

Note that for this simulation, the currents were given to the controller directly, without using a Kalman filter to estimate them. Here, one of the main differences in the working principle of the controllers can be seen. The old controller tries to turn the boat for heading and desired course to align, while the new controller turns the boat to align the vector field with the true course and so quickly converges to the line, compensating the currents with merely some loss of speed.

In the initial simulations, the performance of the new controller is better. It is able to take the curvature of the vector field into account and therefore compensates for changes in the desired direction before they change. Combining it with a Kalman filter to estimate speed over water and currents allow it to follow the path almost perfectly in any simulation. Therefore, it should be expected, that the new controller also performs better in a real experiment especially in two regards:

- The statistical offset to the line over the course of a mission should be much lower.
- True course and desired course should align after the controller converges.

4.1.1.3 Parameters r and K_P

Boatbot was initially equipped with the working PDsin controller from previous projects. This controller was not further optimized in the course of this project. However, there are two parameters that were previously used to significantly affect the performance of the controller. These two parameters are also used to optimize the new controller.

The first parameter r is the flatness of the vector field that lies at the basis of both the old and new controllers. Thus, this parameter does not directly affect the control, but rather the setpoints to which the controller converges, i.e. the desired course ψ. Assuming that the line to follow from west to east is parallel to a degree of latitude, thus $\varphi = 0$, (4.1) applies:

$$\psi = \tanh\left(\frac{x_2}{r}\right) \tag{4.1}$$

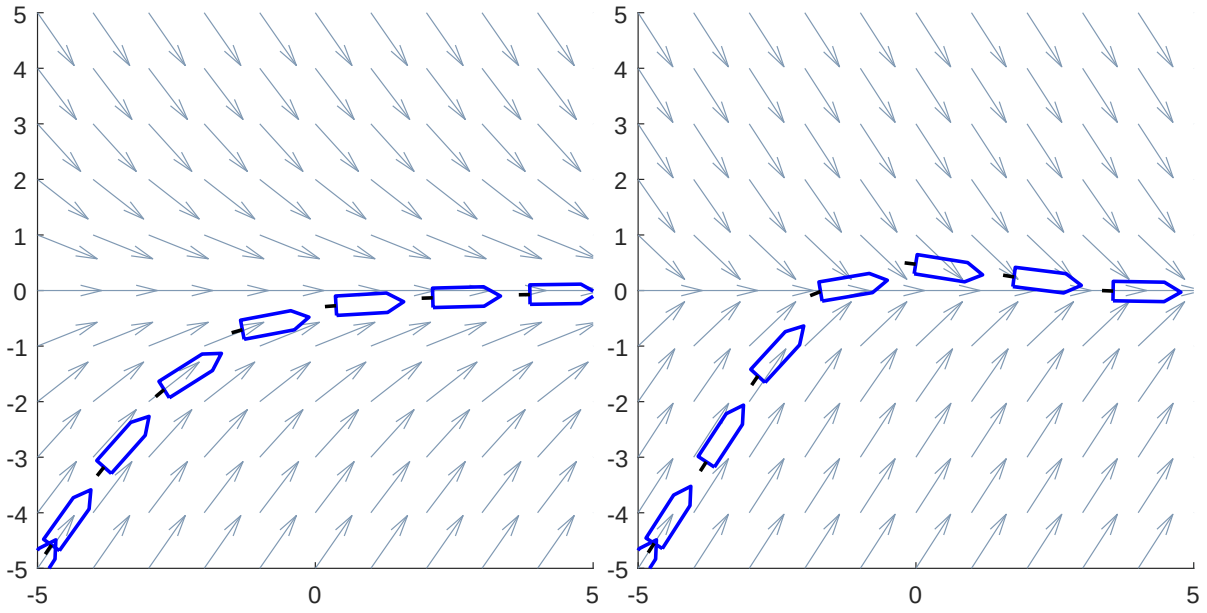
An increase of r will result in less changes to the desired course and thus lower control signals. In this way, for example, oscillation can be prevented from occurring as the speed increases. However, it also means that *Boatbot* takes a little longer to converge to the line. The regulation is softened overall.

The second parameter K_P is a gain that is applied directly to the output signal of the controller. It is used to adjust the output signal to the conditions of the steering servo. In the simulation, this leads to lower rudder deflections. The volatility of the controller output signal with small changes in the external conditions can be reduced. Also, for *Boatbot* relatively small control signals as compared to the simulation are sufficient, as shown in the next section. Therefore, the parameter is usually chosen to be smaller than 1 in this project.

$$\overline{u} = K_P \cdot u \tag{4.2}$$

For the old controller, control works well with these parameters set to $r = 15$ and $K_P = 0.35$. For the new controller they were set to $r = 10$ and $K_P = 0.7$ after some testing.

The following figure 4.6 illustrates the effects of changing the two parameters from the output value $r = K_P = 1$. The simulations uses the new controller with the second-order dynamic model, see also table 4.2. It should therefore be compared with the figure 4.4b.



(a) Parameter $r = 2.5$ (b) Parameter $K_P = 0.5$

Figure 4.6: Influence of parameters r and K_P on the control, here the new controller.

4.1.2 Deployment

In the following, the performance of the old controller (PDsin) will be compared with the performance of the new controller (Feedback linearization) based on two real experiments. The data from the old controller comes from an experiment carried out during *Submeeting 2019* on brique 8 on June 12, 2019. This experiment was used for the comparison because the old controller achieved the best results there. The data of the new controller were collected in an experiment on 18 July 2019 in the same place. The weather conditions were similar on both days with little wind and waves, and a small but noticeable sea current.

The conditions in the table 4.4 are estimated values. The currents data from July 18 comes from the estimate of the Kalman filter, which is integrated in the new controller. The values shown are subject to fluctuations and, among many others, are just three of the most important environmental factors that influenced the experiments.

Day	Controller	Waves	Currents	
			Speed	Direction
June 12th	PDsin	$\sim 0.3\ m$	$\sim 0.5\ {}^{m}/_{s}$	$\sim 60°$ (ENU)
July 18th	Feedback linearization	$\sim 0.4\ m$	$\sim 0.55\ {}^{m}/_{s}$	$\sim 70°$ (ENU)

Table 4.4: Estimated environmental conditions during the experiments.

The following table 4.5 shows the configuration and some mission specifics of the controllers used during the experiments. The time specifically refers to the amount of data used in this validation. During *Submeeting 2019*, the brique was in fact completed.

Controller	Mission duration	Average speed	Control signal gains		Vector field smoothness
PDsin	$58\ min$	$2.01\ ^{m}/_{s}$	$K_P = 0.35$	$K_D = 0.0$	$r = 15$
Feedback linearization	$53\ min$	$2.23\ ^{m}/_{s}$	$K_P = 0.7$	...	$r = 10$

Table 4.5: Specifications of missions and controller configurations.

4.1.2.1 Controller performance

First, the overall results of the regulation are considered. For comparison, data from the first hour of the experiments are used. Due to time constraints, the experiment of July 18 was canceled before the end of brique 8, so the June 12 recordings beyond this point are neglected. This has no influence on the assessment, since this part of the experiment is identical to the previous parts and the external influences did not change significantly during the experiments. In fact, during both experiments the environmental conditions slowly deteriorated over time.

The figure 4.7 schematically shows the process of a mission on a brique. Note that this and all other representations are heavily compressed. An x-axis section in 4.8 represents the same distance as the entire y-axis. *Boatbot* starts at the southwestern edge of the brique and then heads towards the southeastern corner. From there it turns towards the lowest available westward line. A the western side, it turns back to the lowest available eastward line. This process is repeated until the brique is completed. A mission usually takes around 90 minutes, depending on the speed of the boat.

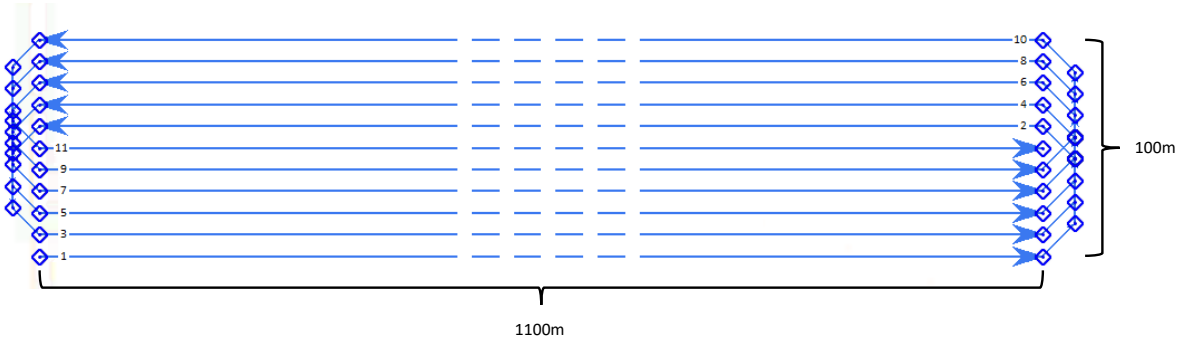


Figure 4.7: GPS track of brique 8.

When looking at 4.8, it immediately becomes clear that the new controller follows the track much more closely than the old one. The oscillations are therefore exaggerated in this representation. For the old controller, these are due to the irregular influence of waves, wind and currents, which are not compensated with this controller.

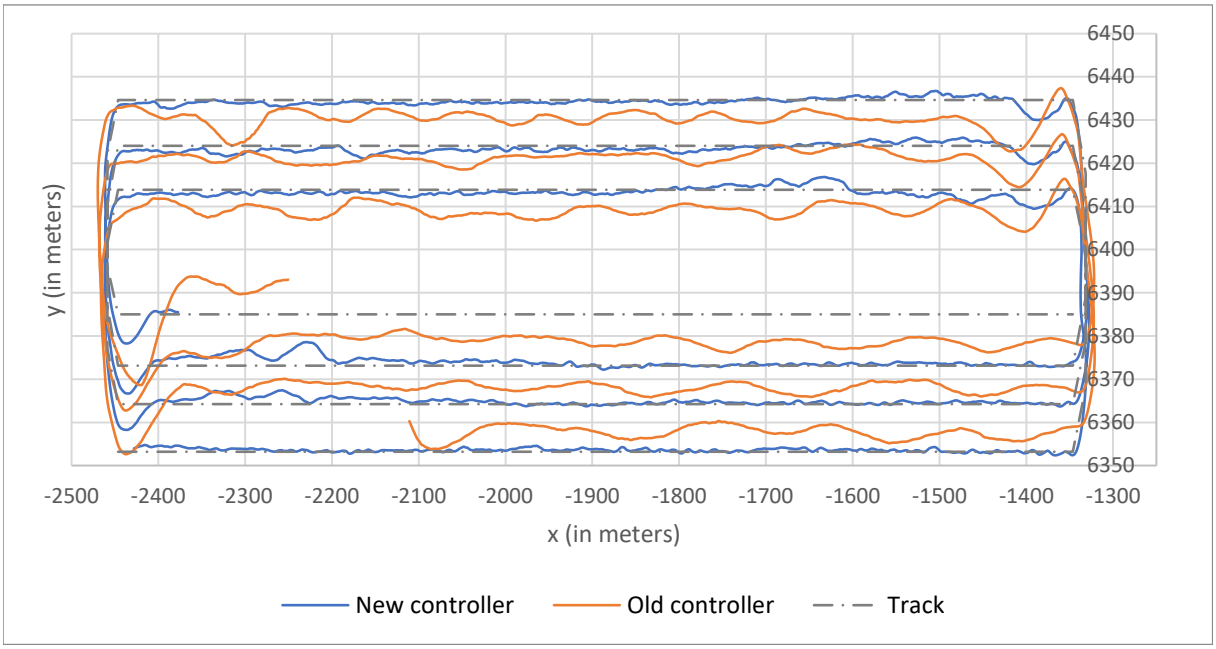


Figure 4.8: Performance of old and new controllers on brique 8.

The transient responses to changes in direction are also noticeable. These are also bigger with the old controller, because here a waypoint is only reached after it's position has been exceeded, whereas with the new controller, waypoints are breached ten meters earlier. The waypoints are at the corners of the track and thus indicate the directional changes. If the waypoints are not reached until they are crossed, this leads to a wide overshoot after the waypoint. This is mainly due to the fact that the system response takes several seconds after a control signal has been sent due to the inertia of the boat and because of the time the steering servo takes to turn the engine. This is an effect that only has been observed during the experiments with *Boatbot*. Therefore, also the optimal values for the parameters r and K_P are different for simulation and the real system.

The figure 4.9 shows the controller output in minutes 10 to 20 of the missions. This includes part of the first straight and the first change of direction taking place at minutes 16 and 17. Noticeable here is a significantly higher amplitude in the oscillations of the fundamental frequency of the new controller, which is probably due to the fact that the targeted course changes more strongly when the track is crossed. With the old controller, this is not the case because the line between the waypoints was never crossed except after turning around.

A larger amplitude is visible as well, which superimposes the signal of the old controller with a lower frequency. Again, the influence of wind and waves becomes visible here, as before in 4.8 and later in 4.10.

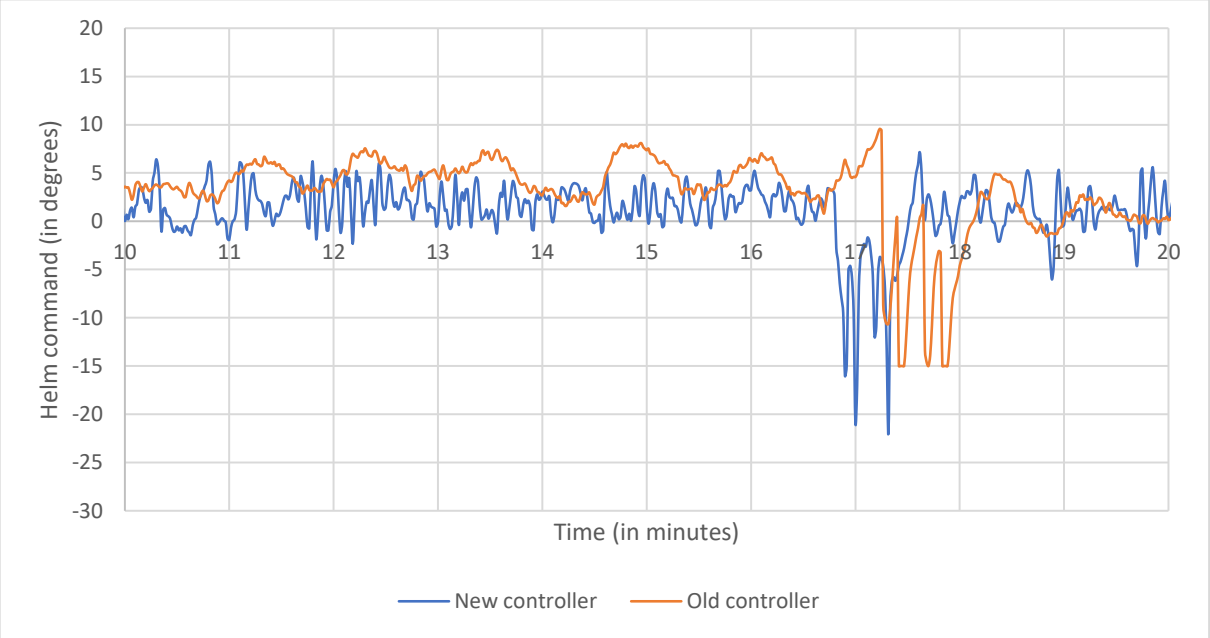


Figure 4.9: Controller output sample from deployment minute 10 to 20.

It can be surmised from figure 4.9 that the old regulator essentially responds to waves and currents that sometimes cause large deflections but require only low-frequency control. On the other hand, the new controller, which compensates for these influences in the first place, has to regulate much more closely in observance of an absolute track, the line between the waypoints, which leads to significantly higher amplitudes in a higher frequency of the control signal.

There is still room for improvement for both controllers, especially while turning after reaching a waypoint. The volatility of the control signal in the event of abrupt changes of the desired course is too great and leads to avoidable oscillations.

The figure 4.10 shows the offset to the track during the missions. The average deviation is calculated from the sum of the amount of the total deviations. On average, the two-sided offset to the track when using the new controller amounts to $1.03m$ in this experiment. Taking this value as an indicator for the performance improvement of the new controller over the old, the result is an improvement of roughly 443%.

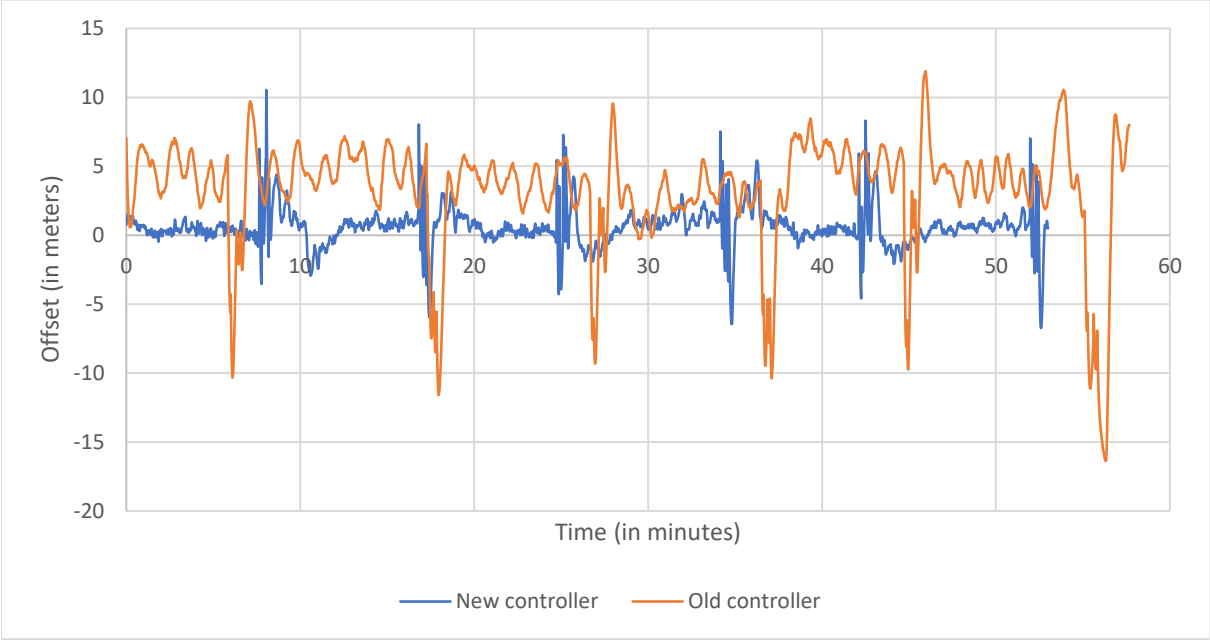


Figure 4.10: Offset to track during mission.

$$\begin{aligned} \overline{d}_{new} &= \tfrac{1}{n}\sum_{i=1}^{n}|d_i| = 1.03m \\ \overline{d}_{old} &= \tfrac{1}{n}\sum_{i=1}^{n}|d_i| = 4.56m \end{aligned} \tag{4.3}$$

In addition to the offset, this representation also shows shorter mission time during the second experiment due to the the slightly higher speed. In both controllers, an increase in speed was usually met with an occurrence of oscillations, which can be prevented by an adjustment of the control parameters r and K_P. For a future continuation of the project, it should be considered to implement an operational amplifier or make the parameters already implemented dependent on the speed of the boat, to better account for speed changes.

4.1.2.2 Course and heading

The old controller is designed to eliminate the difference between heading x_3 and the desired course ψ. The true course $\arctan\left(\frac{\dot{x}_2}{\dot{x}_1}\right)$ was not considered in the definition of this controller. This causes it to generate an offset due to currents. The offset is the distance to the track at which the controller is in equilibrium, i.e. where the offset no longer changes when the control input is 0.

Figure 4.11: Old controller - comparison of desired course with heading and true course.

This results in the true course, invisible to the controller, pointing in the direction of the track, while the desired course corresponds to the inverse tangent of the distance to the track. The heading x_3 is usually between these two values, or converges with the true course if the lateral currents are relatively small, as is the case here on the way west. The figure 4.12 shows a part of the experiment with the old controller. True heading and heading are relatively close together with a visible offset to the north.

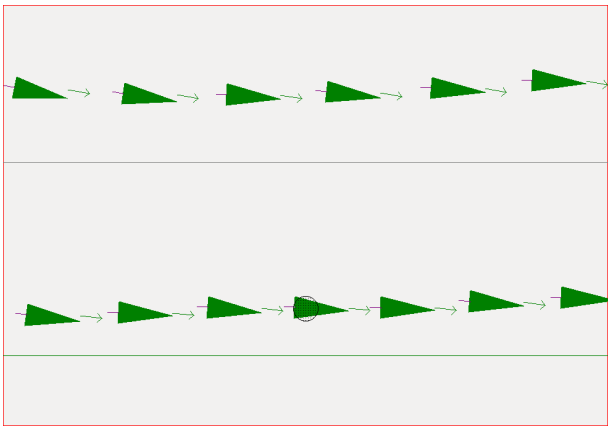


Figure 4.12: Old controller - visualization of a section of the mission. The green arrow is the true course.

Unlike the old controller, the new one controls the difference between the desired course ψ and true course $\arctan\left(\frac{\dot{x}_2}{\dot{x}_1}\right)$. Here, an equilibrium is reached only when for one thing, the boat is

actually on the track and on the other hand turned so that it compensates for the currents without a control input to the angle of the motor.

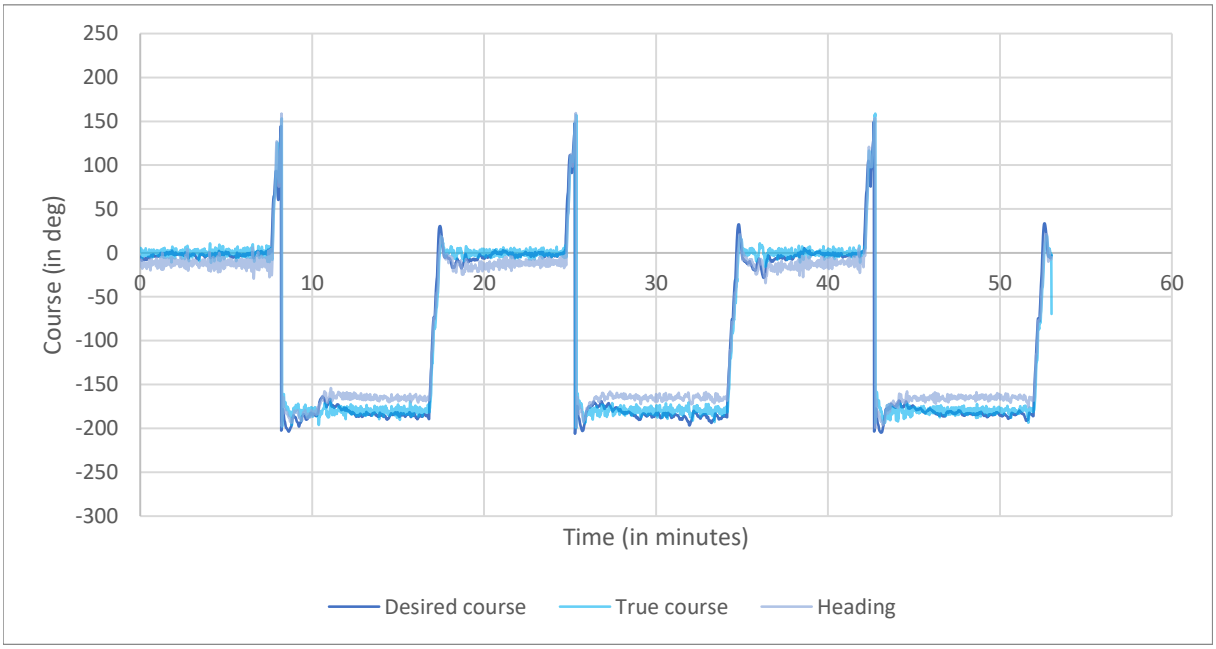


Figure 4.13: New controller - comparison of desired course with heading and true course.

As a result, as shown in figure 4.13, the true and desired courses converge. The boat is automatically rotated against the current from the southwest. The heading is therefore slightly smaller than $0°$ on in eastward direction and somewhat greater than $-180°$ westward. Figure 4.14 shows a noticeable difference between true course and heading. Also, the offset to the track is very small in this instance. It should be noted that the control input, visualized as the rudder, oscillates more compared to the old controller, as already observed in figure 4.9.

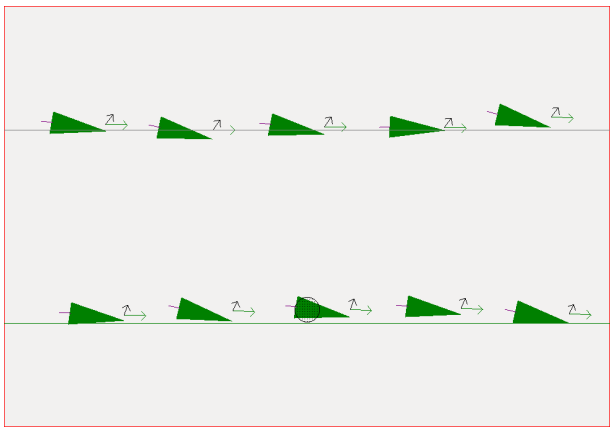


Figure 4.14: New controller - visualization of a section of the mission. The green arrow is the true course. The black arrow is the estimated currents.

4.2 Currents estimation

This part of the validation aims to show that for the control of *Boatbot* a simple Kalman filter on the assumption of static currents, see (4.4), is sufficient to compensate for systematic disturbances that influence the control. For this purpose, a simulation is compared with the experiment of 18 July, which has already been used to validate the controller.

$$\begin{pmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \tag{4.4}$$

The Kalman filter uses as state variables only the velocities of currents p_1 and p_2, as well as the speed of the boat over water p_3. The basis for this are the measurements for GPS speed and direction, which can be decomposed into these components. Other state variables are not estimated and assumed to be known, even if they are erroneous measured values.

Propulsion	Currents		Currents variation		Statistical error	
	Speed	Direction	Speed	Direction	GPS	IMU (compass)
$4.5\ N$	$0.7\ ^{m}/_{s}$	$1\ rad$ (ENU)	$0.3\ ^{m}/_{s}$	$0.1\ rad$ (ENU)	$0.25\ m$	$0.05\ rad$

Table 4.6: Simulation parameters

The simulation was carried out using the same ROS-controller as in the experiment in conjunction with *UxVSim*, an external simulation program which provides simulated GPS and IMU sensors and the state transitions for the boat. The simulations are based on the second order model from (2.14). The parameters, see table 4.6, were chosen to resemble the conditions on the sea more closely than in the **Matlab** simulations. Currents have been implemented which, within limits, vary over time in both direction and speed. The currents have been set to flow in northeastern to northern direction, similar to what has been observed in the area during the experiments with *Boatbot*.

4.2.1 Controller performance

First, the overall result should be viewed. At a first glance, in figure 4.15, there is no systematic offset from the track for either the simulation or the real system. Therefore, as we saw in the previous section, the controller is in fact able to compensate for the currents. However, it is noticeable that after a change of direction the controller needs some time to get back on track.

The absence of any visible offsets in the simulation can probably be explained by the fact that there is no delay between the control input and the system response and also the system is not under the influence of waves.

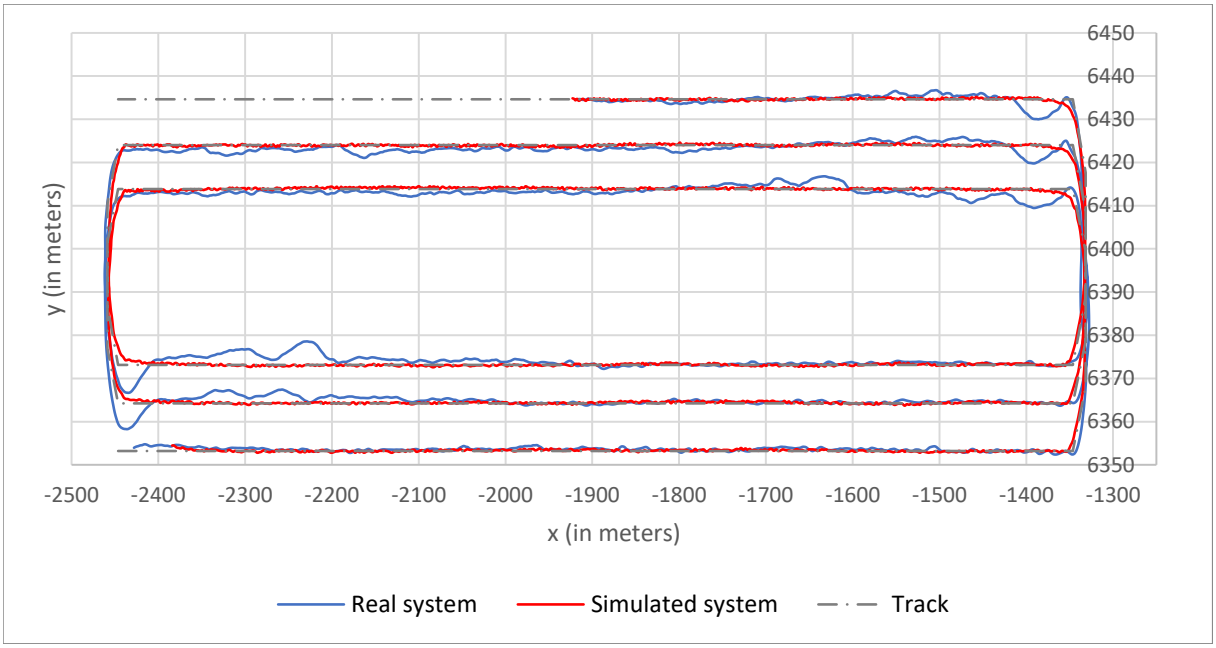


Figure 4.15: Performance of currents compensating controller on brique 8.

Note that this representation is again extremely compressed. A subsection of the x-axis corresponds to the total length of the y-axis. The larger deflections of the real system after changes of direction, seen at the southwest and northeast waypoints, suggest that at these points the state estimate may not have converged, since the controller should be able to return a deflected system without oscillations to bring about a stable state.

4.2.2 Estimation results

The results of the state estimation $\hat{p}$ will now be discussed. The estimation is composed of the speed over water $\hat{p}_1$ and the currents in longitudinal $\hat{p}_2$ and latitudinal $\hat{p}_2$ direction. Subsequently, the currents were combined into velocity v_c and direction θ_c, see (4.5).

$$\begin{aligned} v_c &= \sqrt{\hat{p}_2^2 + \hat{p}_3^2} \\ \theta_c &= \text{atan2}\left(\hat{p}_3, \hat{p}_2\right) \end{aligned} \tag{4.5}$$

4.2.2.1 Currents direction

First, the flow direction is considered. It is noticeable in figure 4.16 that the value in the estimation of the real system is much more volatile. This is to be expected, because many disturbances affect the real system, which are not taken into account in the simulation, but it also becomes clear here that the Kalman filter in the real system reacts more sensitively to the state changes carried out by the controller. The large jumps in the estimates, in the real as well as in the simulated system, are situated where the changes of direction take place at the edges of the brique.

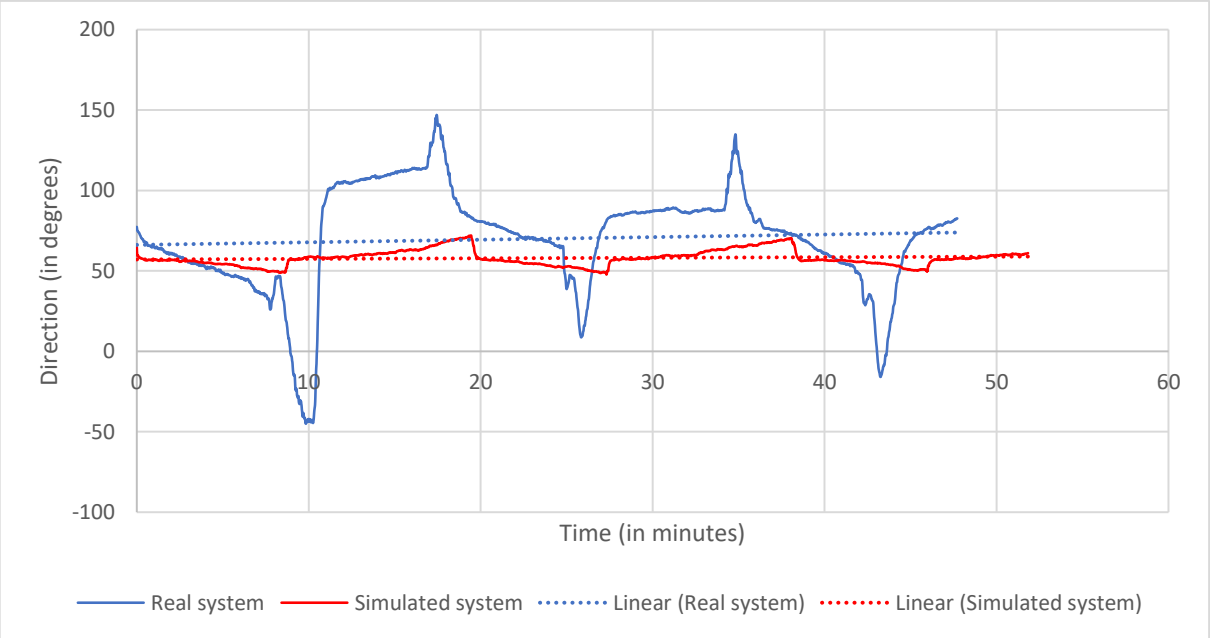


Figure 4.16: Direction of currents during simulation and deployment.

In contrast to these strong deflections in directional changes, in between, while the boat is on the long straight, the convergence of the algorithm becomes visible. Again, however, it is noticeable that the estimated direction of the systematic perturbation, a.k.a. the currents, converges in different directions depending on the direction in which *Boatbot* is moving. For example, the directional estimate for the eastbound path converges to the northeast, while it tends to turn northward on the route to the west. This can be observed for the simulation as well as for the real system, though the effect is stronger in the real system. A possible reason for this effect is that the estimate is only able to distinguish exactly between the speed proportions of boat and currents if the direction of travel changes regularly. This will be discussed in more detail in the next sections.

4.2.2.2 Currents speed

Similar to the estimation of the flow direction, as shown in Figure 4.17, a much greater volatility of the estimation of the real system can be observed. However, some of the estimation plateaus on the straight lines are relatively stable, so it can be assumed that the currents actually reached about $0.55\ ^{m}/_{s}$ during the mission of the real system.

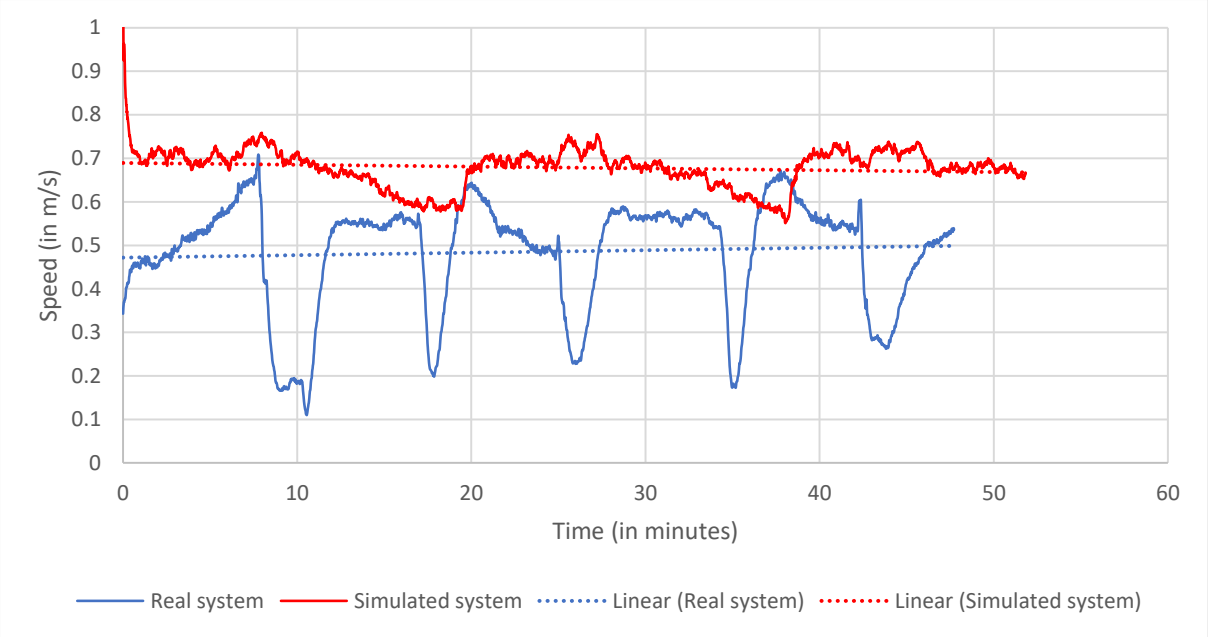


Figure 4.17: Currents speed during simulation and deployment.

Another effect occurs in the results of the simulation. Like the direction, the current changes only to a much smaller extent than in the real system. However, here too, a difference in the estimate depending on the direction in which the boat moves can be seen. Thus, the estimates decrease as the boat moves toward the currents, such as in the minutes 10 to 20 or 28 to 38, while being largely stable in the other periods, or very slightly increasing.

4.2.2.3 Speed over water

Finally, the speed over water should be considered. This is the speed of the boat in relation to the water. If there were no currents, it would be identical to the actual speed. At first glance, the graph of the real system in representation 4.18 seems the least volatile of all graphs so far. The values are all in a range between $1.8\ ^{m}/_{s}$ and $2.4\ ^{m}/_{s}$. Now one would think that the velocity above water should be, more even than the currents, a largely constant value. It should be noted that the significant increases in the minutes 8 and especially 18, but also in 35, and the associated decreases in the real system can be attributed to the fact that the speed was controlled manually on *Boatbot*. The speed was adapted several times during the mission. At minute 11, the speed was throttled significantly to match the speed of another boat, after which it was

increased significantly in the 18th minute. Of course, this makes it difficult to interpret this data in a meaningful way.

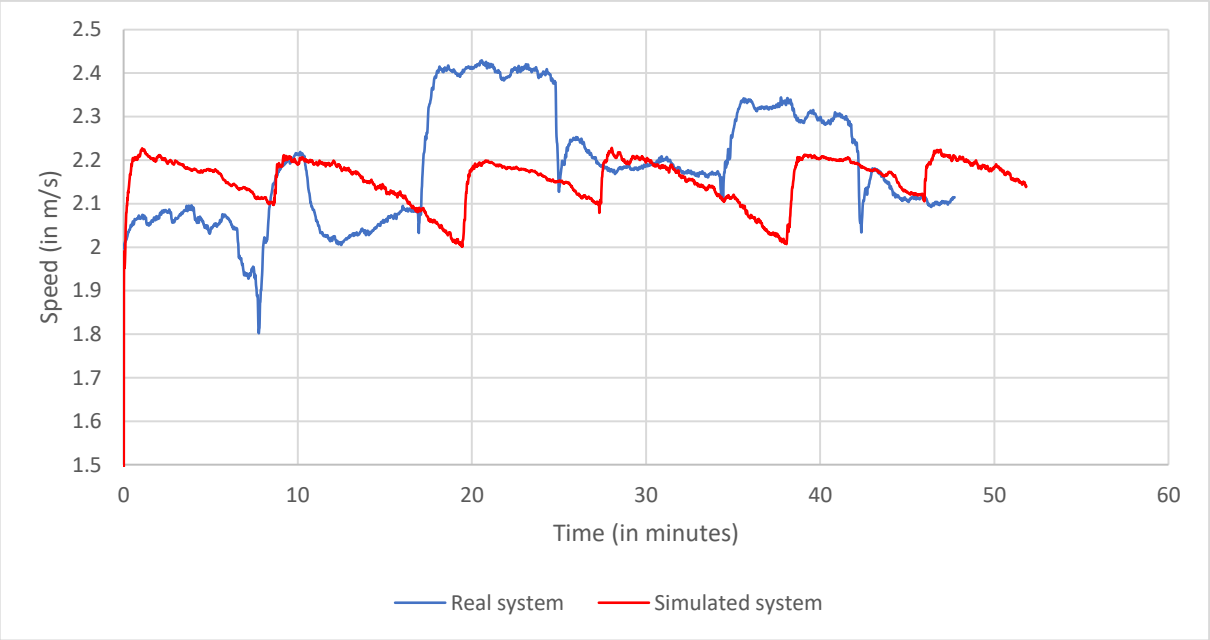


Figure 4.18: Estimated speed over water.

However, a previously observed effect is again evident when looking at the graph of the simulation. The speed decays. So here too, the estimated speed decreases steadily while moving along a straight line. This is particularly evident when going against the flow direction of the currents. If one compares this with the previously considered currents speed, it becomes clear that this decay is mutually dependent. The Kalman filter thus erroneously assumes that the speed of boat and currents decrease equally when going against the current. As a result, the lateral flow component relatively increases here and the flow direction supposedly changes from northeast to north. This effect is similar when going in the direction of the currents. However, here only the speed of the boat decreases slightly and the current turns east to make up for the difference. Summed up, the currents and the speed over water always correspond to the speed measured by the GPS:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} p_1 \cos x_3 + p_2 \\ p_1 \sin x_3 + p_3 \end{pmatrix} \tag{4.6}$$

However, in this experiment, accompanied by a simulation, the limits of the estimate used become clear. The estimation depends on a regular change of states to get a definite result. If the boat travels too long in a straight line, it is no longer able to correctly distinguish between the

speed of the boat and the currents. On the other hand, however, the volatility of the estimates of a change in direction in the real system is far too great, suggesting that values are likely to come closest to reality within the first few minutes after such a change in direction. However, this effect may be reduced by adjusting the covariance matrices of the measurements and disturbances.

4.3 Depth estimation

In order to properly validate the data gathered by the magnetometer, an idea was to measure or estimate the depth of the magnetometer more accurately than it is possible by only the length of the rope between the boat and the depressor.

4.3.1 Sonar

A sonar has been used on *Boatbot* for the purpose of getting a better estimation for the depth of the depressor and the magnetometer. This idea was discarded after some testing when it was discovered that the disturbances in the sonar image were so great that it was impossible to determine the position of the magnetometer or the depressor.

4.3.2 Camera

Another approach to estimating the depth of the depressor and thus also get an estimation for the depth of the magnetometer was to position underwater lights on the depressor. A camera, attached to the bottom of the boat would record the lights. Knowing the depth and mounting angle of the camera, an estimation of the depth of the depressor would be based on the distance between the lights. The calculation could be carried out in an offline or online process.

The idea was discarded after testing the visibility of one underwater light at roughly five meters depth with a camera from below the surface. The light was not visible.It is possible that the light was not strong enough. Further testing has to be carried out in this instance.

Alternatively, it could be feasible to attach the camera to the depressor and record the shadow of the boat. Although this would come with different problems. Implementing this into an online estimation problems would be impossible without another long underwater cable, complicating the setup for the experiment. Also, if the boat is visible from a depth of ten or more meters still has to be tested.

4.4 Magnetic data collection

The data, collected by the magnetometer as a by-product of this work, was not analyzed in detail. However, the quality of the measurements seems sufficient to detect magnetic disturbances under water. This is favored in any case by the large distance between the sensor and

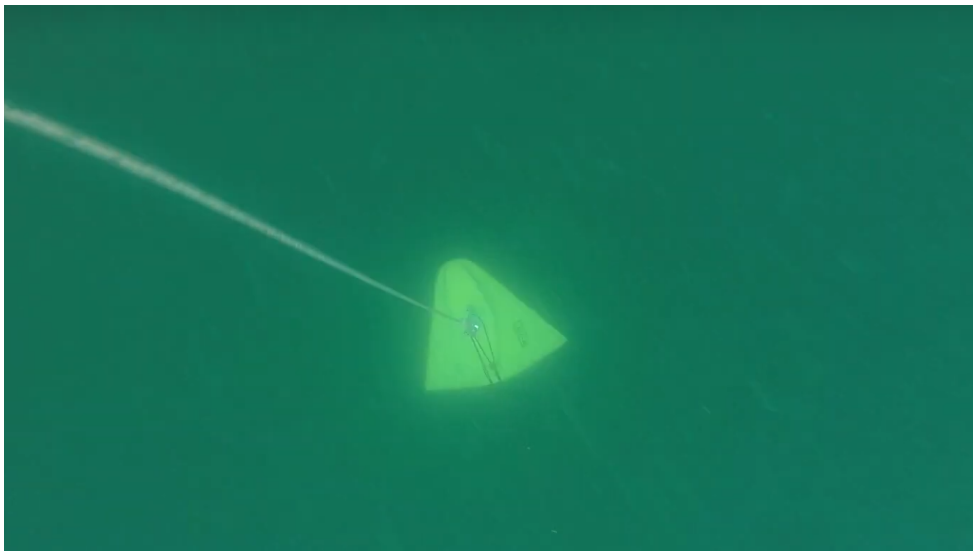


Figure 4.19: The depressor, as seen with a **GoPro** from roughly three meters distance. The attached light is barely visible from farther away.

any magnetic parts. The Depressor, which was in all tests about seven meters away from the magnetometer, contains hardly any magnetic parts and the boat was at all times about twenty meters away from the Magnetometer.

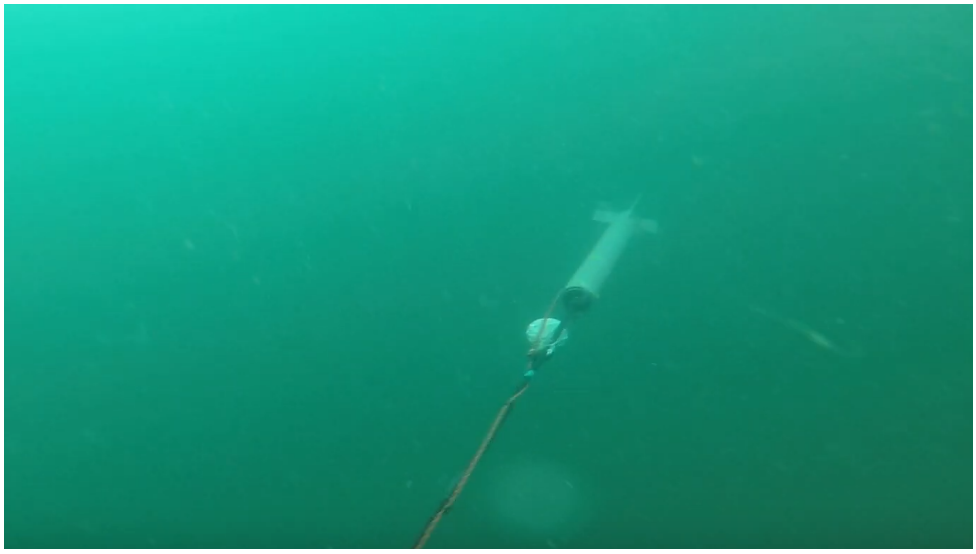


Figure 4.20: The depressor, as seen with a **GoPro** from roughly three meters distance. The attached light is barely visible from farther away.

4.4.1 Magnetometer problems

Nevertheless, the implementation of the magnetometer continues to be provisional. Accordingly, there were many problems that should be considered for a more professional approach to

collecting magnetic data.

- The localization of the magnetic data depends largely on inaccurate estimates based on the boat's GPS position.
- The independence between the hardware and software of the boat and the sensor setup has the advantage that the overall system is less complex and therefore easier to modify. However, it also requires a post-processing of the data, since the magnetic data must be merged with the position data from the GPS.
- The sensor itself has three axes. However, since the orientation of the magnetometer under water is not known exactly, at the moment only the absolute magnetic field strength can be used as a reliable value.
- Connecting cable and rope between boat and magnetometer currently limit the depth of use with a total length of about twenty meters between boat and sensor. The resulting depth is only about five to seven meters.
- The depth of the magnetic sensor under the water surface depends very much on the speed of the boat. It would be better to develop an adaptive depth control or to tow the magnetometer with a UAV.

These are just a few of the many small issues that lie between the data collection project with *Boatbot* and a real professional search for shipwrecks. However, it should be noted that *Boatbot* should rather be understood as a scientific project with a specific purpose as a test platform for various ideas in water robotics and hydrography. A polished overall system was never a goal of the project.

4.4.2 Visualization and findings

During the field testing of *Boatbot* along with the data from ROS, also magnetic data of briques six to ten has been recovered. Magnetic data has also been recorded for brique 52, but as it is not connected to any of the other covered briques, the area is comparatively small. Also, the depth of the oceanfloor varied between 18 and 50 meters - this also depends on the current tide - so that the gathered data is discarded as meaningless.

The setup for the control of *Boatbot* and the sensors is independent. The magnetometer data is given a GPS-timestamp but in order to be visualized it has to be merged with the GPS positions which are logged in ROS. A python program has been written that fulfills that purpose. This program matches the GPS-timestamps of the logs and assigns each GPS-position a magnetic value. It has to be noted that the native sample-rate of the magnetometer at 400Hz is much higher than that of the GPS which samples at 2Hz. For the data already gathered this requires

an offline post-processing in which the magnetometer data is sampled down to 10Hz with a mean function. The GPS data-points are subsequently interpolated to match the total amount of magnetic data. Finally, the values of the magnetometer data are written on a map at their respective coordinates and graphically interpolated.

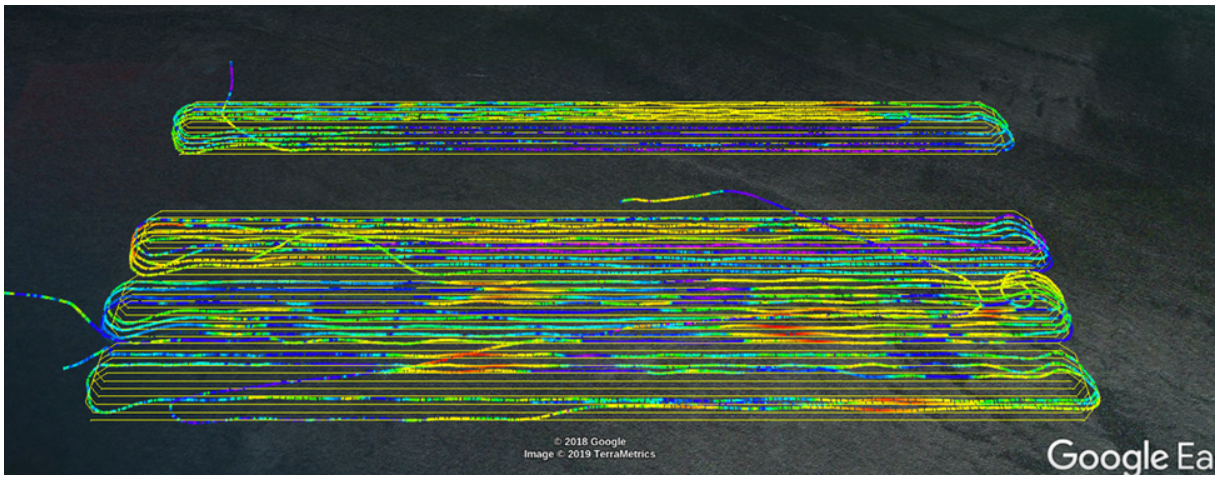


Figure 4.21: Visualization from the **Data_Processor.py** program in **Google Earth**.

In figure 4.21 some findings of the data collection missions from *Submeeting 2019* nare visualized. It becomes clear that there were several problems with these missions. For example, brique 6 was broken off on June 14, 2019 due to excessive waves. It is also not known where the color shift in the data at brique 10 come from.

Nevertheless, at least one can guess from the data, what was already known in parts before. Three red areas in the bottom third of the picture are easy to recognize. From left to right here is the wreckage of a small steamboat named “Penhir”, which sank on the way from Brest to Nantes in 1935, a geological fault, and possibly, in the right-hand section of the image, the remains of an old underwater cable.

Chapter 5

Conclusion

In this work *Boatbot* was presented, a project that explored approaches to the autonomous control and collection of maritime data on a boat. For the autonomous control, data from GPS and compass were used in a controller, which was developed from a feedback linearization method in order to allow for *Boatbot* to follow a vector field as closely as possible. A Kalman filter was also implemented to compensate for disturbances from currents, wind and unforeseen influences.

In addition to theoretical considerations, the technical implementation with regard to hardware and software was also discussed in this work. It has been demonstrated that *Boatbot* is able to locate itself and translate this information into real-time control. In addition, an existing system for data collection has been extended and tested.

It has also been shown in simulations and real experiments that the new controller developed for *Boatbot* is superior to the old one. This was particularly clear in the statistical offset to the track. In addition to the improved controller, this can be attributed to the estimation of the environmental influences by the Kalman filter. Although the Kalman filter was able to fulfill its purpose in terms of control, it will be necessary to optimize this in the future to deliver good result under any circumstances. It has been noted that the quality of the currents estimations depend to a large degree from the quality of the control and the stability of movement of the boat.

Finally, the results of collected magnetic data were considered. It has become clear that the collection of magnetic or other data by following a predefined grid with *Boatbot* is basically possible, but there is still more room for improvement in the approach to the survey, as well as in the evaluation of the data.

Appendix

As an appendix to this work, a digital disk was created. It is enclosed and contains the following data:

1_Literature
: All papers and web pages, which were quoted in the work indirectly or directly, as well as further literature.

2_Implementation
: The ROS software packages used on *Boatbot* and additional software. It contains the entire source code from before and after the implementation of the new controller.

3_Simulation
: The Matlab program created to simulate a boat in a perfect environment. In the file `control.m` the controller can be specified. `main.m` allows to switch between recording and simulation modes.

4_Experiments
: All the data collected during experiments with *Boatbot*. This includes all data generated by ROS and the magnetometer. Also contained are the experimental reports created for *Boatbot* during *Submeeting 2019*.

5_Validation
: The post-processed data from the experiments of June 12, 2019 and July 18, 2019, which were used for the validation in the chapter 4 and the visualizations contained in the chapter.

6_Images
: All images and graphics that were used in the work.

7_Media
: Pictures and videos taken or recorded while working with *Boatbot*.

Bibliography

[1] Fabrice Le Bars and Luc Jaulin. An experimental validation of a robust controller with the vaimos autonomous sailboat. In *5th International Robotic Sailing Conference*, pages 74–84, Cardiff, Wales, England, 2012. Springer.

[2] Boatbot 2018. https://www.ensta-bretagne.fr/jaulin/boatbot2018.html. Accessed: 2019-06-06.

[3] Boatbot 2019. https://www.ensta-bretagne.fr/jaulin/boatbot2019.html. Accessed: 2019-06-06.

[4] Morten Breivik and Thor Fossen. Guidance laws for planar motion control. In *Proceedings of the IEEE Conference on Decision and Control*, Dec 2008.

[5] Andrew W. Browning. A mathematical model to simulate small boat behaviour. *SIMULATION*, 56(5):329–336, 1991.

[6] Michael Burger and Matthias Gerdts. *Applications of Differential-Algebraic Equations: Examples and Benchmarks*, chapter DAE Aspects in Vehicle Dynamics and Mobile Robotics, pages 37–80. Springer International Publishing, Cham, 2019.

[7] La cordelière. https://www.lacordeliere.bzh/. Accessed: 2019-06-06.

[8] Definition robot. https://www.lexico.com/en/definition/robot. Accessed: 2019-07-29.

[9] Drotek store. https://store.drotek.com/DP0501. Accessed: 2019-08-02.

[10] Thor I. Fossen. *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*. Marine Cybernetics, 2002.

[11] Matthias Gerdts. Mathematische Methoden in den Ingenieurswissenschaften. Unpublished Lecture, 2018.

[12] L. Jaulin and F. Le Bars. A simple controller for line following of sailboats. In *5th International Robotic Sailing Conference*, pages 107–119, Cardiff, Wales, England, 2012. Springer.

[13] Luc Jaulin. *Mobile Robotics*. Elsevier, 2015.

[14] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, Mar 2004.

[15] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–journal of Basic Engineering*, 82(Series D):35–45, 1960.

[16] Maël Le Gallic, Joris Tillet, Luc Jaulin, and Fabrice Le Bars. Tight slalom control for sailboat robots. In *International Robotic Sailing Conference (IRSC)*, Southampton, United Kingdom, Aug 2018.

[17] Nmea sentences. http://www.nmea.de/nmea0183datensaetze.html. Accessed: 2019-08-15.

[18] Polulu Corporation. *Pololu Jrk USB Motor Controller Users Guide*, 2019.

[19] Ros tutorials. http://wiki.ros.org/ROS/Tutorials. Accessed: 2019-07-27.

[20] Simon Rouhou. Hydrographes et roboticiens explorent guerldan saison 3. http://guerledan.ensta-bretagne.fr/, 2018. Accessed: 2019-08-10.

[21] SBG Systems EMEA. *Ellipse 2 Series - Miniature high performance inertial sensors*.

[22] SENSYS GmbH. *Data Logging and analysis device for FGM3D sensors FGM3D TD*, 2017.

[23] N.A. Shneydor. Introduction. In N.A. Shneydor, editor, *Missile Guidance and Pursuit*, pages xiii – xvi. Woodhead Publishing, 1998.

[24] C.Y. Tzeng, G. C. Goodwin, and S. Crisafulli. Feedback linearization design of a ship steering autopilot with saturation and slew rate limiting actuator. In *International journal of adaptive control and signal processing*, Keelung, Taiwan, 1999.

[25] u-blox. *NEO-M8P - u-blox M8 High Precision GNSS Modules*, May 2017.

[26] Greg Welch and Gary Bishop. An introduction to the kalman filter. *Proc. Siggraph Course*, 8, January 2006.

[27] Hans Joachim Wünsche. Filter- und Schätzverfahren. Unpublished Lecture, 2019.