GUIDANCE, NAVIGATION AND CONTROL OF AUTONOMOUS VESSELS

An Implementation using a Control-Based Framework

H.L.J. Taams

Delft University of Technology

TU Delft
DAMEN

Guidance, Navigation and Control of Autonomous Vessels

An Implementation using a Control-Based Framework

by

H.L.J. Taams

Student number:	4169131	
Thesis committee:	Dr.ir. T.J.J. Van den Boom	TU Delft
	Dr.ir. S.A. Miedema	TU Delft
	Ir. P. Rampen	Damen Shipyards
	Ir. M. Bharatheesha	TU Delft
	Dr. P. Mohajerin Esfahani	TU Delft
	Dr.ir. R.L.J. Helmons	TU Delft

TU Delft

List of Symbols

$(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$	A continuous or discrete state and input trajectory pair that solves a planning problem. It is used as a time-varying reference during path execution.
α_i	Azimuth angle of thruster i.
$\bar{\boldsymbol{\theta}}$	Estimated parameter vector.
$\bar{\boldsymbol{y}}(t, \bar{\boldsymbol{\theta}})$	Estimated model output for the estimated parameter vector $\bar{\boldsymbol{\theta}}$.
$\dot{\boldsymbol{x}}$	The time derivative of the state $\boldsymbol{x}$.
$\boldsymbol{\eta}$	Vector containing the x, y and ψ states of the vessel expressed in the earth-fixed coordinate frame.
$\boldsymbol{\nu}$	Vector containing the u, v and r velocity states of the vessel expressed in the body-fixed coordinate frame.
$\boldsymbol{\tau}$	The summation of forces acting on the vessel.
$\boldsymbol{\theta}$	True parameter vector.
$\tilde{\boldsymbol{u}}$	A continuous or discrete path in the action space.
$\tilde{\boldsymbol{x}}$	A continuous or discrete path in the state space.
$\boldsymbol{B_T}$	The thruster configuration matrix.
$\boldsymbol{C_{RB}}$	The rigid-body Coriolis and centripetal forces matrix.
$\boldsymbol{I}$	Identity matrix.
$\boldsymbol{M_A}$	The added mass matrix.
$\boldsymbol{M_{RB}}$	The rigid-body mass matrix.
$\boldsymbol{T}$	Vector of projected thrust forces.
$\boldsymbol{u}$	Input of a system.
$\boldsymbol{x}$	A point in the state space. Also referred to as state.
$\boldsymbol{x}_G$	Goal state.
$\boldsymbol{x}_I$	Initial state.
$\boldsymbol{x}_{rand}$	A randomly sampled state.
$\boldsymbol{y}$	Model output.
Δt	A time duration of a sampling period.
Δ_m	Vessel displacement.
$\hat{\bullet}$	The observer estimate of $\bullet$.
$\mathscr{U}$	Set of allowable action trajectories.
$\mathscr{U}_d$	Set of motion primitives.

ψ	Heading of a vessel.
$A(\boldsymbol{x})$	Subset of the workspace occupied by the agent for a state $\boldsymbol{x}$.
B	Breadth of the vessel.
CF	Center of floatation.
COG	Center of Gravity.
D_p	Propeller diameter.
E	An edge or set of edges.
$f(\bullet)$	A function operator.
G	A graph with vertices V and egdes E.
J	Propeller advance speed.
$K_T(J)$	non-dimensional thrust-coefficient.
L	Vessel length.
$L_{i,x}$	The x coordinate of the ith azimuth expressed in the body-fixed coordinate frame.
$L_{i,y}$	The y coordinate of the ith azimuth expressed in the body-fixed coordinate frame.
L_{WL}	Length of the waterline.
n_i	Propeller rotation speed in revolutions per minute, of thruster i .
O	The obstacle region.
p	Pitch, the rotation speed of a vessel about y_b.
P/D_p	Propeller Pitch/Diameter ratio.
q	Roll, the rotation speed of a vessel about x_b.
r	Yaw, the rotation speed of a vessel about the vertical earth-fixed axis.
T	Draft of the vessel.
U	Action space of a system.
u	Surge, the forward speed of a vessel.
u_c	Constant cruise speed.
U_d	A discrete action trajectory or motion primitive.
V	A vertex or set of vertices.
v	Sway, the sideways speed of a vessel.
V_{ctg}	Cost-to-go.
V_{near}	The nearest neigbouring vertex to $\boldsymbol{x}_{rand}$.
VAF	Varience Accounted For.
W	The workspace.
w	Heave, the speed of a vessel along the z_b axis.
X	The state space of a system.

x_b	Axis of the body-fixed coordinate system {b}, positive towards the bow.
X_G	Goal region.
x_{CF}	Distance from the bow to CF.
x_{COG}	Distance from the bow to COG.
X_{free}	The obstacle free state space.
X_{obs}	The state space obstacle region.
XT	The state × time space.
y_b	Axis of the body-fixed coordinate system {b}, positive towards portside.
z_b	Axis of the body-fixed coordinate system {b}, positive pointing upwards.
{b}	Body-fixed coordinate system.
{e}	Earth-fixed coordinate system.
APF	Artificial Potential Field.
ASD	Azimuth Stern Drive, vessel propulsion consist of azimuth thrusters installed at the stern.
BVP	Boundary Value Problem.
DASh	Damen Autonomous Ship, a 1:25 scale model of an ASD3111 Damen tug.
DP	Dynamic Positioning.
DT	Dynamic Tracking.
EKF	Extended Kalman Filter.
GNC	Guidance, Navigation and Control.
LWPR	Locally Weighted Projection Regression, a supervised learning algorithm.
MPC	Model Predictive Control.
NMPC	Non-linear model predictive control.
PRM	Probabilistic Roadmap.
RDM	Rotterdamsche Droogdok Maatschappij.
RRT	Rapidly-exploring Random Tree.

Abstract

This thesis report proposes a framework to implement Navigation, Guidance and Control (GNC) systems, that enable point-to-point autonomy for displacement vessels. A model-based control approach is chosen as the basis of the GNC systems. The resulting algorithms are implemented for verification in a 1:25 scale model of a Azimuth Stern Drive (ASD) 3111 Damen tug named "Damen Autonomous Ship", *aka* DASh.

First, a compact maneuvering model that captures relevant dynamics of displacement vessel is formulated. The dynamic model of DASh is identified using system identification. the propulsion, surge dynamics and sway-yaw dynamics are identified separately by performing bollard pull tests, straight-line acceleration tests and zigzag tests. The parameter estimation problem is formulated as a nonlinear optimization, using a trust-region based solver in Matlab.

Secondly, the guidance system is automated such that it connects an initial state to a goal state with a collision free path that satisfies all input and differential constraints of the vessel model. To this end, the kinodynamic Rapidly-exploring Random Tree (RRT) algorithm is extended to use a maneuver automaton and optimal motion primitives in its steering function. A learned cost-to-go distance metric for the state space is formulated to efficiently calculate distance between states, which is used to search for nearest neighbors in the kinodynamic RRT algorithm. The Locally Weighted Projection Regression (LWPR) learning algorithm is used to approximate the true cost-to-go of an optimal control steering method. The performance of the planner using the learned cost-to-go distance metric is compared to a minimal curve length distance metric based on Dubins Curves and the commonly used straight-line Euclidean distance metric. It is shown that the learned cost-to-go and the minimal curve length distance metric result in paths of similar performance while the Euclidean metric performs severely worse.

Lastly, the navigation and control systems are implemented on DASh. Due to disturbances present in real world environments, the paths must be tracked using feedback control. State estimation for navigation, based on position and heading measurements is performed by implementing an observer using a Extended Kalman Filter (EKF). Non-linear model predictive control (NMPC) in combination with thrust allocation is used to control the vessel during path execution. Due to real time requirements of DASh, the EKF, NMPC and the thrust allocation algorithm are directly implemented in efficient C++, on the on-board computer of DASh.

Model tests for static reference tracking and time-varying reference tracking are performed on DASh. It is shown that NMPC converges to static reference positions faster and with less control inputs compared to traditional non-linear PD control. It is also shown that time-varying trajectories created by the kinodynamic RRT can be executed successfully. This shows that the identified model is suitable for use in model-based control and that the planned paths indeed satisfy the input and differential constraints of the vessel.

Acknowledgments

Performing the research culminating in this thesis has been a great learning experience. I would like to thank my thesis supervisor ir. Mukunda Bharatheesha for all the enlightening conversations we had about path planning and optimal control. Your sharp commentary brought out the best in me. Although the agendas were packed, the doors of dr.ir. Ton van den Boom, dr.ir. Sape Miedema and prof.dr. Rudy Negenborn were always open. Thank you for your guidance and feedback throughout this research project.

Special thanks to my supervisor ir. Peter Rampen from Damen Shipyards. You supported me greatly and helped me out whether I was stuck on a technical question or at a train station. I would also like to acknowledge my Damen colleagues Tim van den Heuvel, Ernst-Jan Goslinga, Kees Custers, Wouter Joosten and Sándor Iváncsics for helping me realize the Damen Autonomous Ship scale model and maintaining the RDM office. It was a delight to work alongside Luc Prawoto, Thom Sneep and Matheus Terrivel, colleagues whom I now consider friends. In addition, I am grateful that Damen Shipyards allowed me to pursue my combined technical interests in offshore and control engineering during this endeavor.

I would like to thank Noor Taams, Margriet Taams and Wim Wagner for their support and help during the scale model tests, even though the water was freezing. Thank you to Daphne Dotsios, for all her love, encouragement and for being an excellent camera operator during the model tests.

Contents

1 Introduction

Ships have been sailing around the world for many centuries. Navigation, Guidance and Control (GNC) of vessels has always been the responsibility of humans, whom determined their own position and intuitively set out routes. The helm was continuously manned to execute subtle course corrections. Over the last century, advances in technology produced detailed digital maps, accurate position measuring systems and steering controllers. Implemented in every new-build industrial vessel, these technologies are referred to as GNC systems and help captains and helmsmen make informed decisions. Over time, individual GNC subsystems have been automated resulting in autopilots for course-keeping and the Global Position System (GPS) for reliable navigation. As a result, the human role aboard vessels has become less critical and perhaps, in the future, obsolete. The benefits of unmanned ships could be substantial. Crew cost are mitigated and vessels could be built without crew accommodations, saving weight and costs. On top of that, automation may lead to safer shipping, as accidents and casualty situations are often the result of human errors [61]. A concept design of an unmanned Damen tug in figure 1.1 reveals what vessels may look like in the future.

Figure 1.1: A concept design of an unmanned Damen tug.

The technological push towards unmanned vessels resulted in the automation of station keeping by so called Dynamic Positioning (DP) systems and the automation of sailing between predetermined way-points by Dynamic Tracking (DT) systems [9]. Guidance for DT systems however remains simplistic compared to the robotics and automotive industry that include system dynamics and behavioral planning of surrounding vehicles [47, 56]. Navigation and Control on the other hand, has for decades attracted considerable attention within the maritime industry [22, 29].

The currently installed individual guidance, navigation and control systems already largely elevate tasks that were previously performed by skippers. Detailed and constantly updated digital maps shown the position of

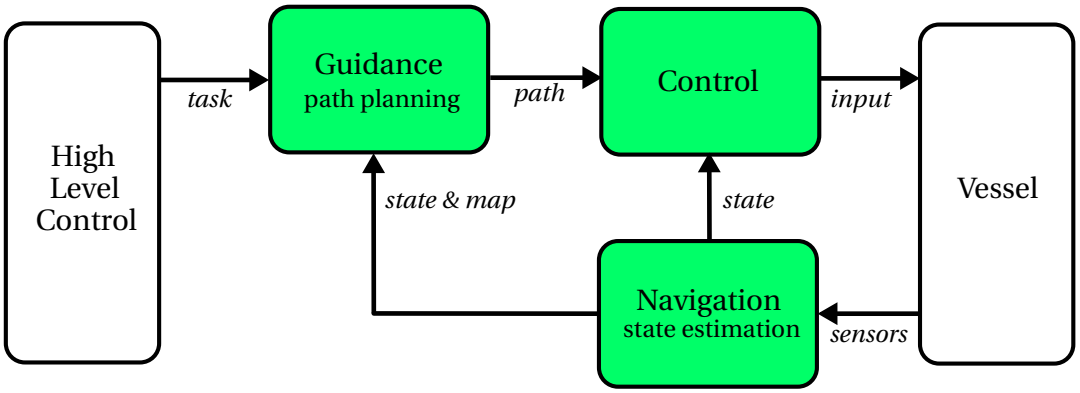

Figure 1.2: A simplified representation of the GNC system to achieve autonomous sailing. GNC subsystems are displayed in green.

the vessel while also displaying information on other ships in the vicinity [72]. Course keeping control systems account for environmental forces to maintain a steady course over ground. To fully automate a vessel, one must let the GNC systems communicate their results and give it permission to act, without the interference of humans. The resulting system is then able to execute tasks, given by a high-level controller, by itself.

Automating the GNC systems is a only small step towards truly unmanned vessels. Tasks performed by humans such as maintenance, communications, loading, unloading and mooring will still have to be tackled. Intermediate steps like partial automation and the tele-operation of vessels are explored in the industry. In the drone community however, small autonomous vessels are being build for application of shore and inland water monitoring [20, 33]. In the research community many more test platforms exist which were recently surveyed in [63].

1.1. The Guidance, Navigation and Control Problem

This work is concerned with the automation of guidance and control systems using a model-based approach. It is assumed a high-level controller provides a point-to-point navigation task consisting of: The initial position of the vessel, a goal position we would like to sail to and a map of static and moving obstacles. The vessel has to autonomously reach the goal position. The GNC subsystems must execute the following task: The navigation system estimates the vessels current state and updates moving obstacles on the map. The guidance systems plans a collision free path from the current state to the goal position. The control system determines what control forces and moments must be exerted by the propellers and rudders, in order to follow the path. If the tracking error becomes too large or if an unforeseen obstacles would cause a collision, a new path must be planned. For autonomous operation, the GNC system is closely integrated and responsible for self-state estimation, path planning and motion control. The simplified representation of the architecture and data flow of an autonomous GNC system is shown in figure 1.2.

In the robotics community, guidance is more commonly referred to as *motion planning* or *path planning*. Path planners for vessels do exist, but mostly to perform single operations such as station keeping or low speed way-point tracking. A recurring approach in path planning for ships is to *decouple* the planning into two stages; The first stage plans a geometrically constrained global path indexed by way-points. The global paths avoids collision with static obstacles. The second stage performs way-point following using a local controller. The controller only uses information about its immediate surrounding to reach the next way-point. It is also responsible for path execution and to avoid collisions with moving obstacles, a functionality referred to as Collision Avoidance (CA).

Decoupling decreases the complexity of the planning problem by breaking it up into two subproblems. The reactive part of the planner makes the approach robust to moving obstacles and input, sensor and model disturbances. The decoupling also results in lower computational time before path execution, as geometrically constrained global paths require significantly less computational power compared to differentially con-

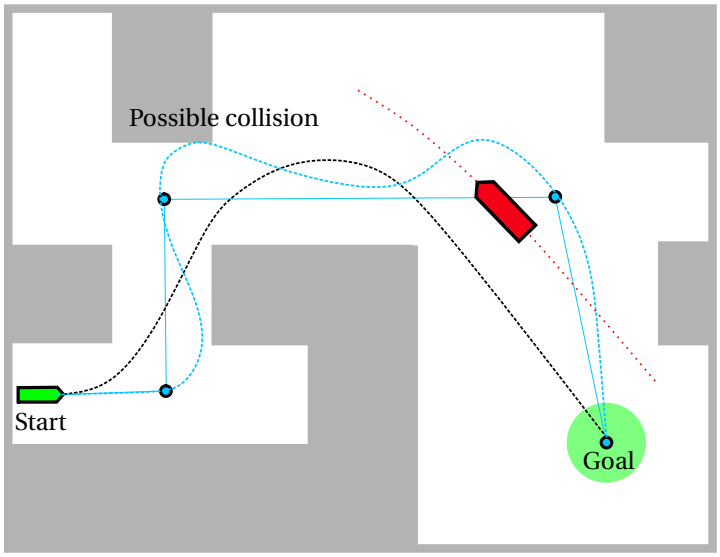

Figure 1.3: The decoupled approach finds a way-point indexed global path (solid blue) that is tracked using a motion controller and CA algorithms, resulting in the dashed blue path. Directly accounting for the vessel dynamics and moving obstacles (red) results in the dashed black path.

strained paths [48]. Yet this is also its mayor downside as differentially constrained planning approaches account for the vessel dynamics that locally restrict the movement of the vessel such that it is *guaranteed* that the control systems is able to reliably track the path. This guarantee is missing using the decoupled approach, which could result in collisions. Secondly, the decoupled approach may find longer paths compared to differentially constrained paths that would be found by directly taking into account the moving obstacles. Lastly, differentially constrained planning is able to optimize desirable attributes of paths, such as maximum accelerations or fuel consumption.

All in all, the decoupled approach is simple to implement and works well at open water, but is not suited for narrow or congested waters where the vessel dynamics play a more significant role. An illustration of the decoupled approach and the dynamically constrained approach is shown in figure 1.3.
This thesis proposes a point-to-point planning framework that directly accounts for the vessel dynamics and moving obstacles during path planning, while keeping the computational loads low. Therefore the following research questions are formulated:

- How should the vessel dynamics be modelled and identified if it is to be used for guidance, navigation and control?
- How can the dynamics of the vessel be taken into account during point-to-point path planning?
- Can the computational loads of dynamically constrained path planners be reduced to allow on-line implementation?

The research questions will be answered based on the following assumptions

- Environmental forces acting on the vessel during path execution may be neglected.
- It is assumed perfect knowledge on the location of static and moving obstacles is available.

To underwrite the theoretical results of this thesis, the GNC algorithms will be implemented on a scale model. Therefore the following secondary goals are formulated:

- Design and construct a scale model outfitted with hardware to enable autonomy.
- Implement the updated GNC algorithms on the scale model.
- validate the updated GNC algorithms using model tests.

1.2. Report Outline

This report consists of two parts. The first part concerns the ship modelling and system identification of the scale model using knowledge obtained in the *Offshore & Dredging Engineering* master track. The hardware of the scale model is presented in chapter 2. A mathematical vessel model that captures relevant vessel dynamic with a minimal set of model parameters is proposed in chapter 3. The model parameters are identified in chapter 4.

The second part concerns the model-based guidance and control systems predominately based on theory learned in the *Systems & Control* master track. The fundamentals of guidance for our scale model is presented in chapter 5. A control-based planning framework is presented in chapter 6 to allow for fast path planning under differential constraints. The control performance of the scale model for static and moving reference tracking during model testing is presented in chapter 7. The overall thesis conclusions and recommendations for future research are given in chapter 8.

1.3. Notation

The following notational rules are implemented in this report. Vectors are denoted by bold lower-case Latin or Greek characters (e.g. $\boldsymbol{x}$). Matrices are denoted by Latin bold capital letters (e.g. $\boldsymbol{M_{rb}}$). The identity matrix is always denoted by $\boldsymbol{I}$ and a zero matrix or vector by $\mathbf{0}$. The size of the identity or zeros matrices is given in superscript, e.g. $\mathbf{0}^{n\times m}$ is a zero matrix with n rows and m columns. Scalars values are denoted by lower *or* upper-case, non-bold, non-italic Latin or Greek characters. If parameter A is a function of b, it is displayed as $A(b)$. Note that A and or b may be a matrix or vector in the correct character style (e.g. $\boldsymbol{C_{rb}}(b)$).

I

Vessel Modelling and Identification

2

DASh Scale Model

To validate the GNC algorithms discussed in this thesis, a 1:25 scale model of a Damen Azimuth Stern Drive (ASD) 3111 Tug was custom built. The scale model was named "DASh", short for "Damen Autonomous Ship". As the hardware aboard DASh will dictate the actuators constraints and available measurements to be used for navigation and control, an overview of the hardware and software architecture of DASh is given first.

The hull of DASh was made by a professional model maker. Actuators, sensors and communications were build and integrated by a mechanical engineering intern, an embedded systems intern and myself. Particular care was taken to streamline data flow between subsystems to enable future autonomy. In this chapter, an overview of the hardware and software architecture of DASh is given.

2.1. DASh Particulars

The hull shape of DASh enables the vessel to be highly maneuverable. DASh has a more pronounced “V shape” in the forward section compared to traditional tugs. The vessel is course stable. An overview of DASh is shown in figure 2.1. The ship’s particulars obtainable by direct measurements are summarized in table 2.1. A geometric overview is given in figure 2.2.

Figure 2.1: An overview of DASh. The colored boxes on deck contain indoor positioning sensors.

Table 2.1: The ship particulars of DASh.

Ship Particulars			
	Symbol	Value	Unit
Length	L	1.20	m
Length waterline	L_{wl}	1.13	m
Breadth	B	0.41	m
Draft	T	0.25	m
Length bow to COG	x_{COG}	0.57	m
Length bow to CF	x_{CF}	0.63	m
x-position azi.	$L_{i,x}$	0.49	m
y-position azi.	$L_{i,y}$	0.10	m
Displacement	Δ_m	39.28	kg
Propeller Diameter	D_p	0.080	m
Pitch/Diameter	P/D_p	0.8	-

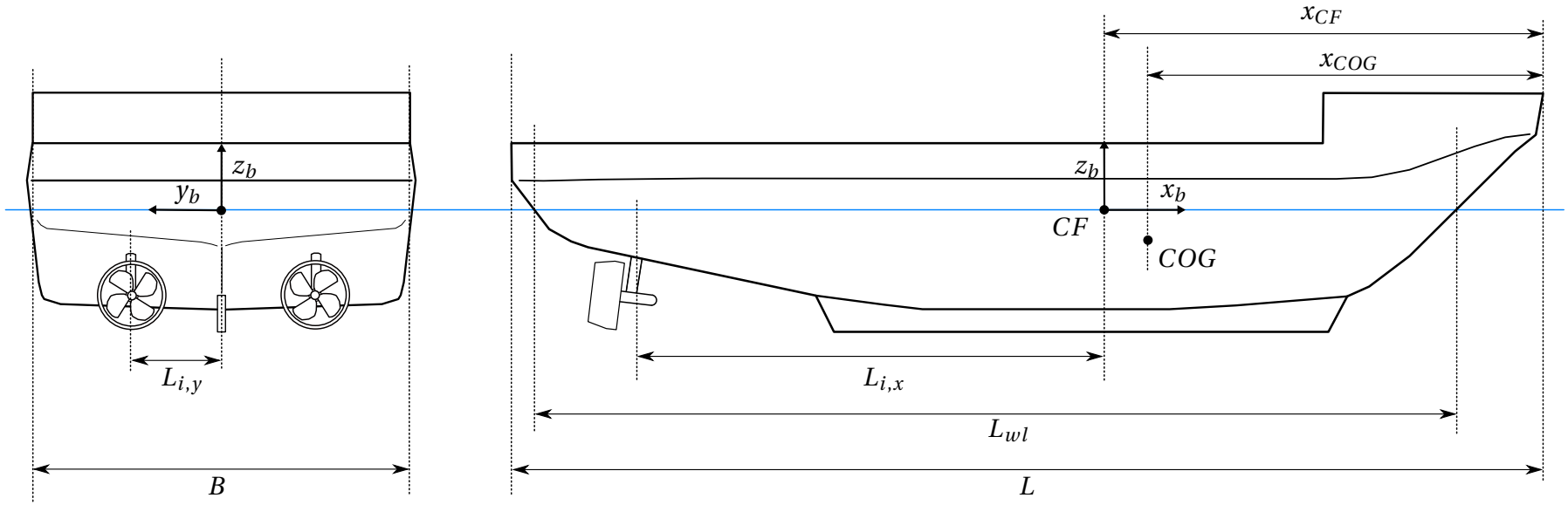

Figure 2.2: (left) Aft view of DASh. (right) A side view of DASh. The waterline is drawn in blue. The principle dimensions, location of the center of flotation CF, the center of gravity COG and azimuths are shown.

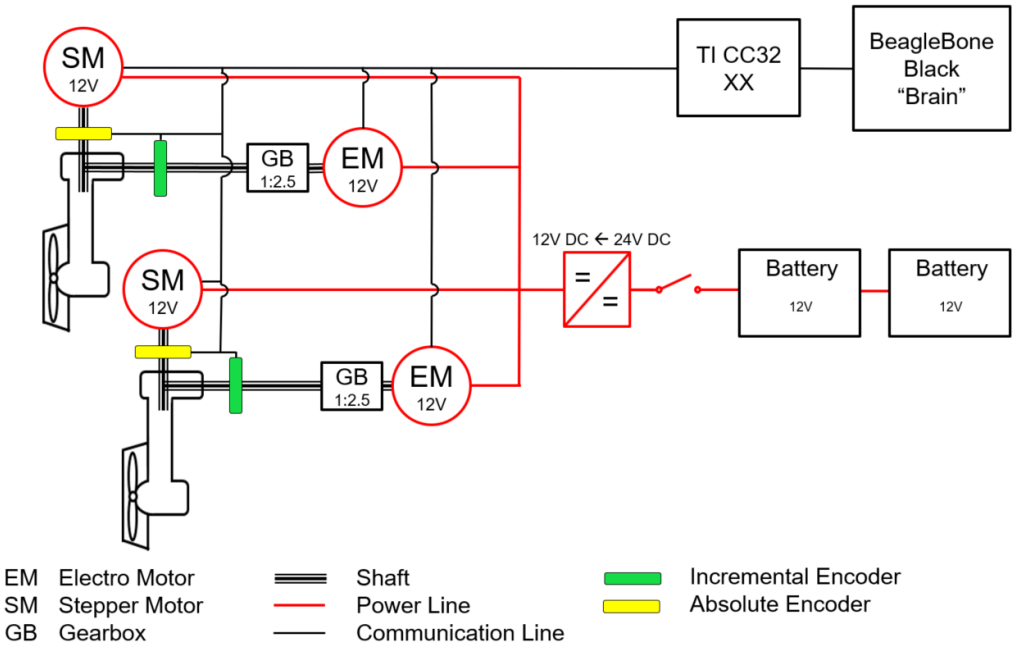

Figure 2.3: A schematic overview of the propulsion configuration.

2.2. Propulsion Train

DASh is propelled by two azimuth thrusters located at the stern. They are powered independently by two 12V DC-motors. A 1:2.5 reduction gearbox was fitted on the motor shaft, to set a realistic range of the propeller rotation speeds of 400 RPM to 1200 RPM. The azimuth angles are set by two 12V stepper motors. They operate at 0.71A and have a holding torque of 1.28Nm. The stepper motors are connected to the azimuth shafts by a belt drive. They can rotate the azimuths at a maximum speed of 1 rad/s and a step resolution of 0.45 degrees. A top view of the propulsion train is shown in figure 2.5. The propulsion configuration is shown in figure 2.3. The rotation speed of the propellers are controlled using a PID controller with a feed forward term for quick convergence to the reference rotation speed. The propeller rotation speed is measured using an CUI INC AMT102 incremental encoder. Due to significant friction in the drive lines, commanded rotational speeds below 400 rpm tend to oscillate around the set reference speed. The influence of bad tracking at low rpm's during operation is unknown.

The azimuth angles are controlled using a stepping scheme, that rotates towards the reference at constant speed until it is reached. The azimuth angles are measured using CUI INC AMT203 absolute encoders with a measurement accuracy of 0.2 degrees.

The four blade azimuth propellers have a diameter D_p of 80 millimeter. The propeller is fitted inside a nozzle with an inner diameter of 82 millimeter. The propeller has a pitch of 64 millimeter, resulting in a pitch/diameter ratio of 0.8.

DASh is powered by two generic 12V, 10Ah lead acid batteries connected in series. Two stable voltages are needed during operation: A 12V voltage circuit to power the propulsion train and a 5V voltage circuit to power the micro controllers and other remaining subsystems. Since battery voltages will drop during operation, two DC/DC converters in cascaded configuration stabilize the voltages from 24V to 12V and from 12V to 5V. An on/off switch can be used to power on or off the entire system.

2.3. System Architecture

The system architecture can roughly be divided in two parts: The Low-Level and the High-Level control system. The Low-level system directly controls actuators and handles communication with sensors. Sensor data is shared with the High-Level control system that performs all GNC tasks. Actuator references are sent back

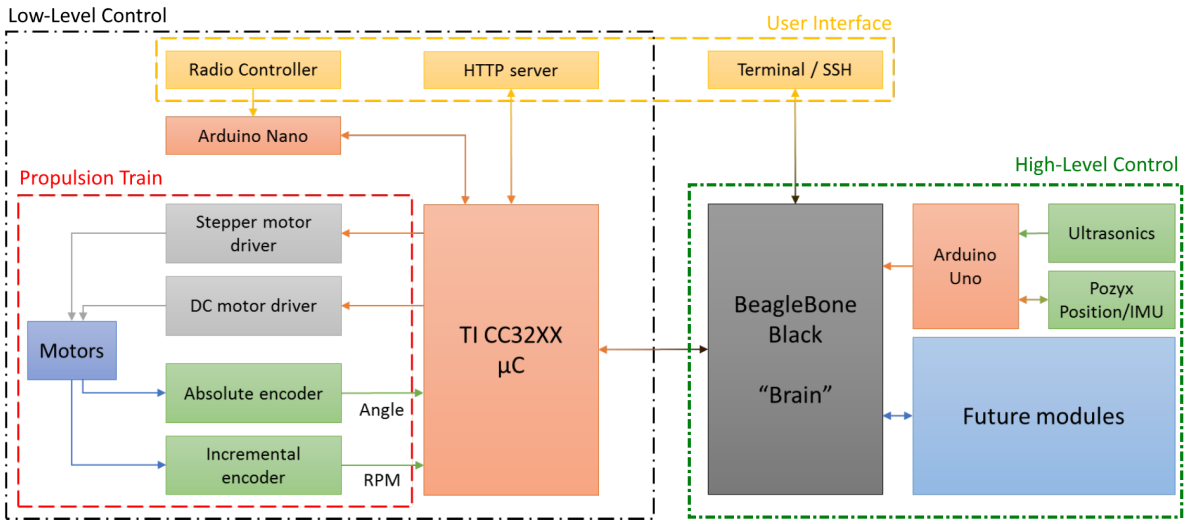

Figure 2.4: The complete software architecture of DASh, including the low-level control, high-level control and user interface.

to the low-level system.

2.3.1. Low-level Control System

The low-level control systems comprises a main controller, DC and Stepper motors, propulsion train sensors and an Arduino Nano that handles the communication with radio controller. A TI CC32XX μc micro controller from Texas Instruments is implemented as a main controller. It tracks the propeller rotation speeds and azimuth angle references sent by the high-level controller. The TI CC32XX host a webserver that displays the current sensor readings. The webserver can also be used as an user interface to control the actuators directly. Finally, the actuators can be controlled directly using a radio controller (hand held joystick). The radio controller overrides any task set by the high-level controller or webserver and thus also acts as a kill switch in case unwanted autonomous behavior is observed.

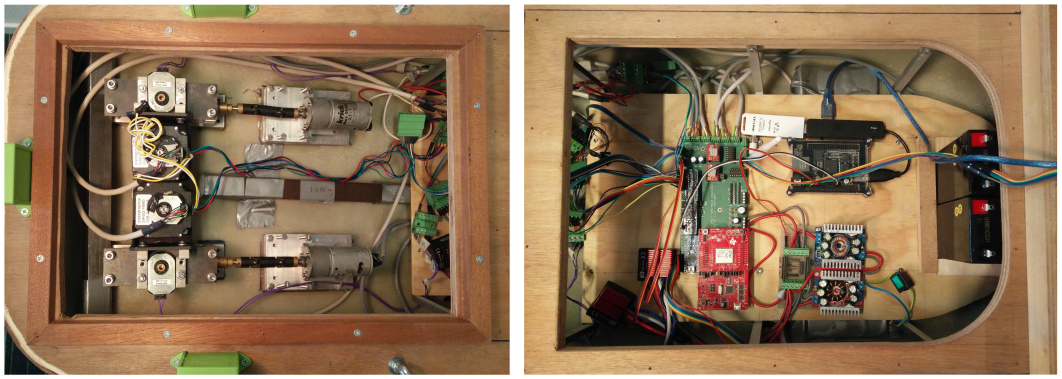

Figure 2.5: (left) The propulsion train in the aft of the vessel powering two azimuth propellers. (right) The following electrical components are named clockwise, starting from the right. Two 12V, 10Ah lead acid batteries, two DC/DC converters. The TI CC32XX μc board (red) placed on a custom cable management PCB (green). The BeagleBone Black with USB-hub (black) and USB WiFi adapter (white).

2.3.2. High-level Control System

The high-level control systems consists of a BeagleBone Black Revision C, an Arduino Uno, the indoor positioning system Pozyx [46] and ultra sonic distance sensors. The BeagleBone is a master to the lower level TI CC32XX μc controller. The planar x, y position and heading measurements are gathered by the Pozyx and sent to the BeagleBone via an Arduino Uno. The Arduino Uno also gathers data from 6 ultra sonic distance

sensors, mounted in green housings on deck. The beagleBone runs a Debian 9.2 Linux operating system and handles the computationally heavy GNC algorithms. Its terminal can be accessed over WiFi via SSH (Secure Shell communication protocol). Future modules can be easily implemented on the BeagleBone due to its high compatibility. An overview of the total architecture is given in figure 2.4.

2.4. Test Environment

DASh was tested in the water basin at the RDM innovations dock in Rotterdam. The rectangular 10 x 20 meter basin has a water depth of 0.7 meter and is outfitted with a wave making system. The tank is free from external disturbances such as wind. The water temperature is not controlled. Ambient air temperature is controlled using floor heating. No towing carriage is present on which captive tests could be performed. The water basin is shown in figure 2.6.

Figure 2.6: The water basin at the RDM Innovation dock.

3

Ship Modelling

In the previous chapter, the 1:25 scale model DASh was presented. In this chapter the equations of motions of DASh are determined based on rigid-body dynamics. The necessity of a detailed vessel model is threefold. Firstly, it is an be used in observer and control design to estimate unmeasured states, filter measurement, and apply model-based control methods. Secondly, a model-based approach in planning and control allows the user to understand the inner workings of the algorithms, in contrast to model-free learning methods where decisions are made by obscured layers of neural nets e.g. Thirdly, using a generic vessel model allows the algorithms to be easily transferred to other vessels using their respective dynamics.

Various types of vessels exist with fundamentally different dynamics such as planing vessels, sailboats and semi-submersibles. This work only covers the modelling of of displacement vessels such as DASh in which lift caused by buoyancy forces dominates relative to hydrodynamic forces. Most commercially operated vessels are displacement vessels.

3.1. Vessel Motions and Coordinate Frames

A displacement vessel is modelled as a single rigid body with six Degrees Of Freedom (DOF) which fully define the position, rotation and velocities of the vessel. Existing models used for speed prediction, time-domain simulation and motion controller design are based on rigid-body dynamics. To describe ship motions and their respective names, a body-fixed reference frame {b} is defined with origin O_b, as shown in figure 3.1. The orthogonal axes x_b, y_b and z_b in {b} are directed along the principle axis of the vessel:

- x_b is positive towards the bow.
- y_b is positive towards portside.
- z_b is positive pointing upwards.

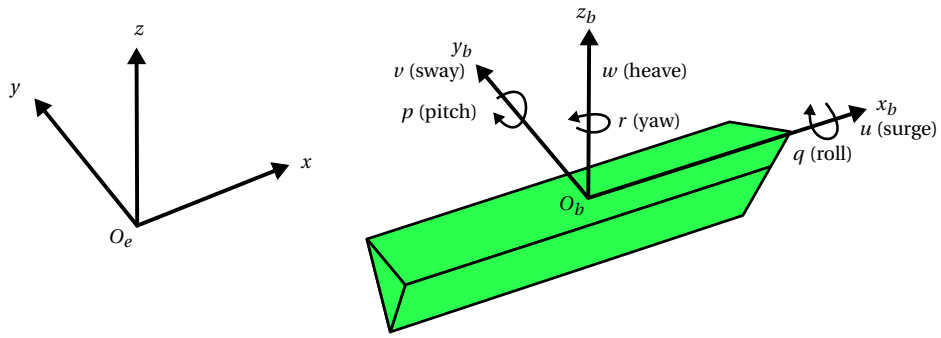

Figure 3.1: The earth-fixed coordinate system {e} on the left and body-fixed coordinate system {b} on the right.

velocities along x_b, y_b and z_b are referred to in shipping as *surge, sway* and *heave* velocity, denoted by u, v and w, respectively. Rotational velocities about the x_b, y_b and z_b axes are referred to in shipping as *pitch, roll* and *yaw*, denoted by p, q and r. The linear velocities (u, v, w) are defined positive along the body-fixed axes. The angular pitch, roll and yaw velocities (p, q, r) are positive according to the right hand rule applied to the axes of {b}. The vessel position and orientation in space are captured in an earth-fixed reference frame {e} with origin O_e. The x, y and z coordinates capture the position of O_b and the Euler angles (ϕ, θ, ψ) describe the orientation of the vessel.

Following the notation of SNAME [66], the forces acting on the rigid body at O_b in the direction of x_b, y_b and z_b are denoted as X, Y, and Z. The moments about x_b, y_b and z_b are denoted as K, M and Z.

3.2. Modeling Approaches

Two distinct modelling approaches can be found in literature that have their focus on different types of vessel motions. *Maneuvering* models aim at modelling vessels traveling at constant forward speed in calm waters. Hydrodynamic coefficients of the model are assumed to be frequency independent. As a result, maneuvering models neglect the influence of wave excitation on the in-plane speed of vessels.

Sea-keeping models have a particular focus on dynamics as a result of wave excitations. This is done by introducing frequency dependent hydrodynamic coefficients that more accurately model dissipation forces[1]. As a result, sea-keeping models better estimate pitch, roll and heave motions of the vessel in waves.

Guidance and navigation tasks are specified in the x, y-plane which implies control of position and heading and the surge, sway and yaw modes. In this situation, correct modelling of low frequency surge, sway and yaw modes is of more importance than modelling the high frequency roll, pitch and heave modes. For a longitudinally and metacentrically stable ship, the pitch, roll and heave in calm waters are small $\phi = \theta = p = q = w \approx 0$ and may be neglected. The resulting 3 DOF maneuvering model is used for guidance, navigation and control of the vessel [64, 81], as shown in figure 3.2.

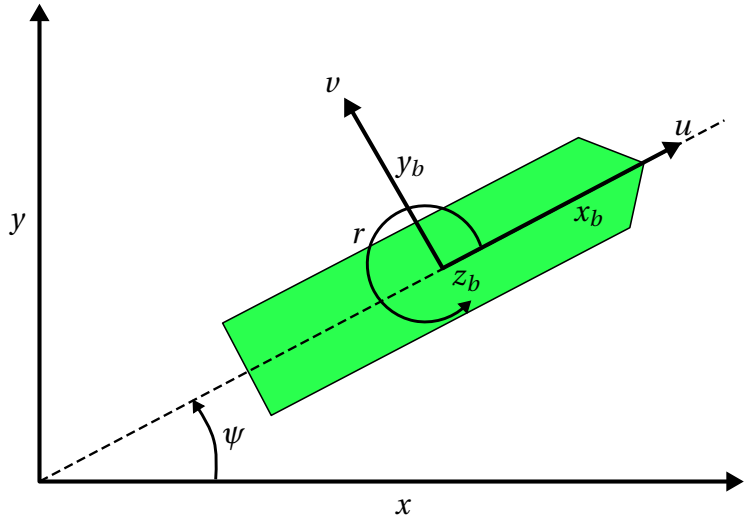

Figure 3.2: The 3 DOF earth-fixed coordinate system {e} and body fixed coordinate system {b}.

[1] also known as fluid memory effects

3.3. Rigid Body Dynamics

The equations of motion are determined using the Newton-Euler equation for a rigid body with respect to the body-fixed reference frame. The origin of this frame does not have to coincide with the Center of Gravity (*COG*) of the vessel. Let the origin O_b coincide with the Center of Flotation *CF*, placed at the centroid of the water plane area. Roll and pitch in calm water will be about this point. Assuming a 3 DOF representation, if the position of *COG* in {b} is denoted by $r_g = \begin{bmatrix} x_g & y_g \end{bmatrix}$ the rigid body dynamics are given by

$$\begin{aligned} m[\dot{u} - vr - x_g r^2 - y_g \dot{r}] &= X \\ m[\dot{v} + ur - y_g r^2 + x_g \dot{r}] &= Y \\ I_z \dot{r} + m[x_g(\dot{v} + ur) - y_g(\dot{u} - vr)] &= N \end{aligned} \tag{3.1}$$

Coriolis and Centrifugal forces arise as a result of the rotation of the body-fixed frame with respect to the inertial frame. A full derivation of (3.1) can be found in [24]. The forces X, Y and N are a summation of forces acting on the hull:

Hydrostatic forces. A ship floats as a result of the hydrostatic pressure acting on its hull, exerting a restoring force. If the hull shape and mass distribution of the ship is engineered properly, the ship is metacentrically stable. This means it floats upright and can right itself under roll or pitch angles.

Hydrodynamic forces. During free sailing, several hydrodynamic forces act on the hull of the ship. These forces are caused by the interaction of moving fluids along the hull of the vessel. An overview of hydrodynamic forces is given in section 3.4.

Environmental forces. External forces which are not related to hydrostatic or hydrodynamic forces are often stochastic in nature. Forces as a result of wind and waves are directionally dependent and vary in size.

Control forces. The control forces result from the propulsion, rudder lift forces, stabilizer fins or other controllable surfaces. These inputs are used to control the motion of the vessel.

The summation of these forces are the input to the rigid-body dynamic model

$$\boldsymbol{\tau} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \boldsymbol{\tau_{hs}} + \boldsymbol{\tau_{hyd}} + \boldsymbol{\tau_{wind}} + \boldsymbol{\tau_{wave}} + \boldsymbol{u} \tag{3.2}$$

Where $\boldsymbol{u}$ are the control forces, which are also referred to as control *inputs*. Assuming portside starboard symmetry, the *COG* is located at a distance x_g from *CF* and $y_g = 0$, simplifying the equation further. Using a matrix notation, equation (3.1) can be compactly written as

$$\boldsymbol{M_{RB}}\dot{\boldsymbol{\nu}} + \boldsymbol{C_{RB}}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{3.3}$$

$\boldsymbol{M_{RB}}$ is the rigid-body inertia matrix, $\boldsymbol{C_{RB}}(\boldsymbol{\nu})$ is a matrix containing rigid-body Coriolis and centripetal forces

$$\boldsymbol{M_{RB}} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix}, \qquad \boldsymbol{C_{RB}}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix} \tag{3.4}$$

where $\boldsymbol{I_z}$ is the rotational moment of inertia about the vertical axis. The vector $\boldsymbol{\nu}$ contains the vessel velocities in the body-fixed frame

$$\boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{3.5}$$

The vector $\boldsymbol{\eta}$ contains the vessel coordinates in the earth-fixed inertial frame

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \tag{3.6}$$

The total vessel state is denoted as $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\eta}^T & \boldsymbol{\nu}^T \end{bmatrix}^T$. The coordinate frames are related by the kinetic relation

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{3.7}$$

where $\boldsymbol{R}(\psi)$ is the rotation matrix around the vertical axis of the inertial frame. The rigid body dynamics in (3.1) and (3.7) lay at the foundation of various maneuvering models.

3.4. Hydrodynamic Forces

The interaction of moving fluids and the vessels hull and propellers give rise to various hydrodynamic flow phenomena.

Potential damping models the effect of fluid pressure forces on the vessel hull as if it travels through an ideal fluid. Damping refers to the fact the forces are generated that oppose the direction of movement of the vessel, thus reducing the energy in the system [40]. The magnitude of the force can be modelled linearly with speed, but is dependent on the (wave) frequency.

Inertial fluid forces arise if a body in fluid accelerates or decelerates. Fluid particles around the hull will have to accelerate in tandem with the vessel resulting in inertial fluid forces. This is phenomena is modelled by increasing the original mass of the vessel with an Added Mass term.

Viscous Damping results from the laminar or turbulent fluid boundary layer on the vessels hull. Its magnitude scales quadratically with speed due to turbulence. Damping forces as a result of a boundary layer is also referred to as skin friction.

Wave drift damping accounts for the added resistance for vessels in waves. Its scales quadratically with the wave height and will dominate resistance terms in the surge direction in high seas.

Damping due to vortex shedding occurs if bodies with sharp edges travel through viscous fluid. The shedding of an eddy introduces a drag force that scales quadratically with fluid speed.

Lifting forces are the result of pressure differences acting on the sides of the vessel due to sideways motion. The body of the vessel acts as a wing creating a lift force perpendicular to the direction in which the vessel is moving. Fluids that flow from side to side by going underneath the vessel give cause to non-linear damping forces. The influence of these flows are referred to as cross-flow drag, which is a non-linear function of sway and yaw velocities. The cross-flow drag becomes dominant in tight turns during which large sway velocities occur, relative to the surge velocities.

The modelling of all phenomena and their respective interactions results in models with a large amount of parameters. If in-depth knowledge of flow around the wetted surface is required, Computational Fluid Dynamics (CFD) methods can approximately predict fluid flow based on the Navier-Stokes equations and fluid boundary conditions. CFD is however computationally expensive and unsuited to be used as a maneuvering modelling method.

3.5. Control Forces

The control forces arise from the reaction forces of any controllable actuator on the vessel. In this work only the azimuth thruster will be discussed, as it the only type of actuator found on DASh. The force generated by a propeller is called thrust. The total thrust T per propeller depends on the flow of water entering and exiting through its blades. Factors such as the inflow speed, the shape of the hull and density of the water all influence the hydrodynamic forces acting on the propeller.

A popular empirical model discusses the properties of a propeller in uniform flow. This particular test situation is also referred to *free running* or *open water* condition. In this condition it is assumed the propeller travels at constant advance speed V_a through a stationary fluid with density ρ. The relation between the

thrust T and the propeller diameter D_p and its rotational speed n is captured in the non-dimensional thrust coefficient $K_T(J)$, given by

$$K_T(J) = \frac{T}{\rho D^4 n^2} \tag{3.8}$$

The thrust coefficient if a function of the advance ratio J, which defines the angle of attack of the propeller blade with respect to the fluid flowing into the propeller. It is defined as:

$$J = \frac{V_a}{nD_p} \tag{3.9}$$

The values of $K_t(J)$ can be found in an open water propeller diagram for the particular propeller, one of which is shown in figure 3.3.

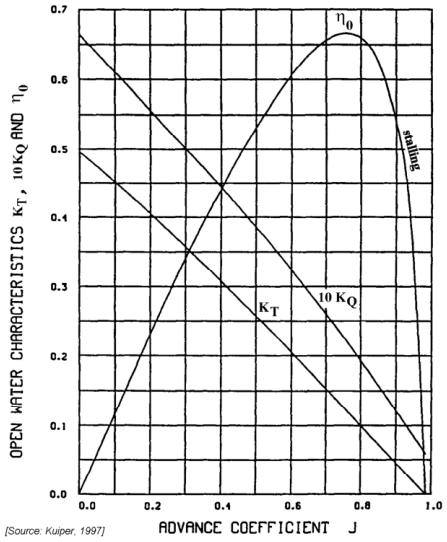

Figure 3.3: An open water propeller diagram.

According to Blanke [8] the thrust coefficient for a range of J may be linearly approximated as

$$K_t(J) = \gamma_0 - \gamma_1 J \tag{3.10}$$

where $\gamma_0 > 0$ and $\gamma_1 > 0$. Substituting (3.10) into (3.8) and grouping the constants, the approximate formula for thrust becomes

$$T = T_{nn} n^2 + T_{nu} nu \tag{3.11}$$

Where T_{nn} and T_{nu} are constants based on the propeller properties. For ease of notation, the thrust forces T_i of both azimuth propellers $i = 1,2$ can be projected on the body-fixed x_b and y_b axis. The resulting projected forces $T_{i,x}$ and $T_{i,y}$ are a function of the azimuth angle α_i and are calculated as

$$\boldsymbol{T} = \begin{bmatrix} T_{1,x} \\ T_{1,y} \\ T_{2,x} \\ T_{2,y} \end{bmatrix} = \begin{bmatrix} T_1 \cos(\alpha_1) \\ T_1 \sin(\alpha_1) \\ T_2 \cos(\alpha_2) \\ T_2 \sin(\alpha_2) \end{bmatrix} \tag{3.12}$$

Resultant control input $\boldsymbol{u}$ acting on CF can be calculated using

$$\boldsymbol{u} = \boldsymbol{B_T T} \tag{3.13}$$

where $\boldsymbol{B_T}$ is the constant thruster configuration matrix describing the position of all thrusters

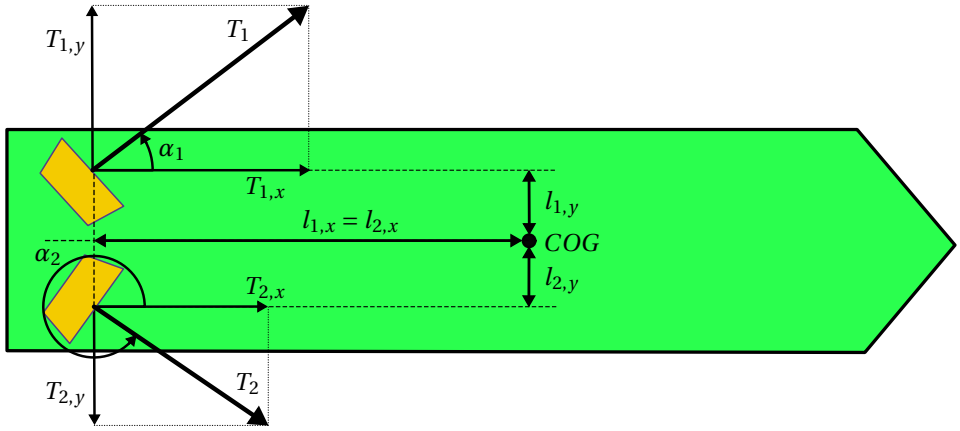

Figure 3.4: The thruster layout and projected thrust forces.

$$\boldsymbol{B_T} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -|l_{1,y}| & |l_{1,x}| & -|l_{2,y}| & -|l_{2,x}| \end{bmatrix} \tag{3.14}$$

where $l_{i,x}$ and $l_{i,y}$ are the absolute distances from COG to the axis of the azimuth in x and y directions of the body fixed frame. and $\boldsymbol{u}$ are the system inputs. A sketch of the actuator forces are shown in figure 3.4.
Since the input dimension of $\boldsymbol{T}$ is higher than the dimension of $\boldsymbol{u}$, equation (3.13) is underdetermined and has no unique solution. Optimization methods that minimize $\boldsymbol{T}$ are used to determine a optimal solution, given a desired $\boldsymbol{u}$. These optimizations are referred to as thrust allocation algorithms and can take maximum thrust constraints and azimuth rotation constraints into account. A complete approach for fixed and rotatable thrusters is presented in [76]. An overview of thrust allocation methods is surveyed in [38]. The thrust allocation used in DASh is based on the constrained minimization of a quadratic cost function using a Sequential Quadratic Programming (SQP) solution. Details of this implementation can be found in appendix B.

3.6. Modelling of Hydrodynamic Forces

In literature, two approaches to parametrize the hydrodynamic forces dominate. The first approach was pioneered by the Japanese Mathematical Modelling Group (MMG) that advocated to empirically calculate the forces acting on individual modules of the ship. Concrete methods to do so were first presented by Inueo [36]. Hydrodynamic forces are defined per module and by summation the total force can be found

$$\begin{aligned} X_{hyd} &= X_H + X_R + X_P \\ Y_{hyd} &= Y_H + Y_R + Y_P \\ N_{hyd} &= N_H + N_R + N_P \end{aligned} \tag{3.15}$$

where the subscript stand for the interaction with the hull, rudder and propeller forces. Using empirical methods, each force can be estimated separately. A generalized MMG approach is presented in [80].

The second approach is to parametrize all hydrodynamic and input forces using a Truncated Taylor series expansion, also referred to as the Abkowitz model [1]. The fit is based on experimental data obtained from captive or free-running model tests. The resulting model contains multiple non-linear terms. A Taylor series expansion to approximate the hydrodynamic force in the surge direction is given as:

$$\begin{aligned} X_{hyd} = &X_0 + X_u \Delta u + X_u \Delta u^2 + X_{uuu} \Delta u^3 + X_{vv} v^2 + (X_{rr} + m x_g) r^2 + \\ &X_{\delta\delta} \delta^2 + (X_{vr} + m) vr + X_{r\delta} r\delta + X_{v\delta} v\delta \end{aligned} \tag{3.16}$$

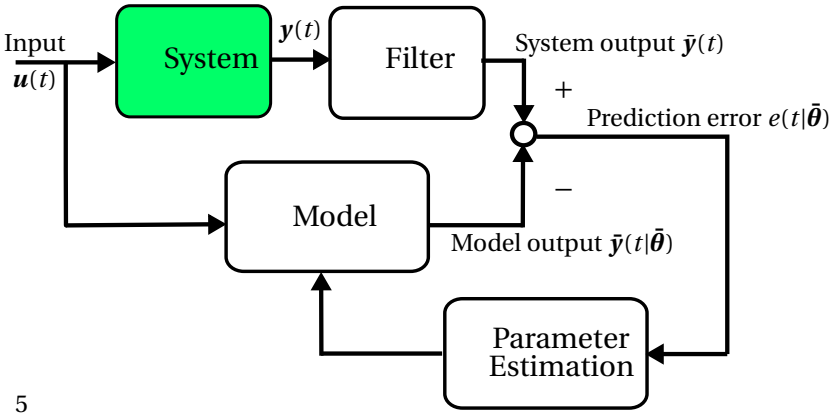

Figure 3.5: The structure of system identification methods. After performing model tests, the prediction error between the system output and the model output is minimized using a parameter estimation algorithm.

Here δ is the angle of the rudder. The model is parametrized around a design velocity u_0. To utilize the model at speeds other then the design speed the delta operator is used

$$\Delta u = u - u_0 \tag{3.17}$$

The Abkowitz model is sensitive to over parametrization due to the large amount of coupling parameters between states. In contrast to the MMG model, the truncated Taylor series regression models the total hydrodynamic forces as a result of ship velocities and control inputs. Modelling of forces as shown in equation (3.16) results in complex maneuvering models that consists of 40 to 150 hydrodynamic parameters [70]. Nevertheless, the Abkowitz model is widely used and does provide adequate results for the prediction of vessel maneuvering.

In this thesis a combination of the MMG and Abkowitz model will be used. The propulsion forces and hydrodynamic forces will be superimposed, similar to the MMG model. The hydrodynamic forces acting on the hull will be modelled using a truncated Taylor series, similar to the Abkowitz model.

3.7. Determination of Hydrodynamic Coefficients

Traditionally the numerical determination of hydrodynamic parameters is done by means of towing tank tests. A scale model is fixed to a towing carriage that imposes a surge and sway speed while measuring the resultant hydrodynamic and damping forces. Multiple test are carried out such as straight-line, rotating-arm or planar-motion-mechanisms tests. The goal is to isolate several parameters for identification per test, making their numerical identification straight forward. To determine a complete maneuvering model, a large series of towing tank tests is needed, making identification time consuming and expensive.

System Identification (SI) is a field of mathematics that use statistical methods to build models of dynamic systems based on system output measurements. Let $\boldsymbol{\theta}$ be the true parameter vector defining all numerical constants in the vessel model. SI techniques try to find the best approximation $\bar{\boldsymbol{\theta}}$ of the true parameter vector, by minimizing the prediction error between the (filtered) system output $\boldsymbol{y}(t)$ to the model output $\bar{\boldsymbol{y}}(t|\bar{\boldsymbol{\theta}})$ under identical inputs $\boldsymbol{u}(t)$, as shown in figure 3.5.

Using SI, it is possible to determine a multitude of parameters when provided with output measurements of free-running tests. In contrast to towing tank tests, SI methods are able to determine many parameters from a single dedicated run, greatly reducing the time and cost compared to towing tank tests. A big plus is that SI can directly determine the maneuvering model on any scale, including real-size vessels. The identified parameters avoid any bias that may normally result from scaling parameters obtained using towing tests[70].

Various parameter estimation algorithms have been used in literature to estimate vessel parameters from system output measurements. One of the earlier methods used the Extended Kalman Filter (EKF) by augmenting state variables to include the parameters. Their numerical value is obtained during the *a posteriori* update

of the EKF [25, 79]. A popular method is the Least Squares (LS) method as used in [65]. Global optimization methods such as simulated annealing were successfully used to identify a linearized maneuvering model in [70]. An adaptive parameter estimation was performed for a non-linear vessel model using a backstepping procedure in [11]. However, only six parameters were identified. In [14] an Opposition-Based Particle Swarm Optimization was performed to identify twelve parameters in a linear heave-pitch model. The methods consistently returned good fits on the measurement data. Bhattacharyya and Haddara [7] used artificial neural networks to identify hydrodynamic derivatives based on spectral analysis methods resulting in an adequate vessel model. Neural Networks methods do however have defects, as they require large training data sets, are susceptible to over fitting and easily settle in a sub-optimal solution.

3.7.1. Parameter Identifiability

The proper choice of model is important when using SI methods. Models with few parameters are easier to identify as the sensitivity of the output with respect to a change in a parameter, is higher. As a result, SI optimizations converge more quickly. Performing SI for models with a very large amount of parameters will result in over fitting. In this scenario the fitted parameters correspond to closely to the training measurements and will fail to predict general movements. In contrast, if too little parameters are present in the model, the system can become under-modelled and will fail to correctly capture the vessel dynamics. A trade-off must be made between model complexity and being under-modelled.

Not all parameters in the MMG and Abkowitz models can be identified using SI techniques. Take for example equation (3.3). Neglecting wind and wave forces, it can be rewritten to

$$\dot{\boldsymbol{\nu}} = -\boldsymbol{M}_{\boldsymbol{RB}}^{-1}\boldsymbol{C}_{\boldsymbol{RB}}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{M}_{\boldsymbol{RB}}^{-1}\boldsymbol{B}_{\boldsymbol{T}}\boldsymbol{u} + \boldsymbol{M}_{\boldsymbol{RB}}^{-1}\boldsymbol{\tau}_{hyd} \tag{3.18}$$

Based on measurements of $\boldsymbol{\nu}$, only the matrices $\boldsymbol{M}_{\boldsymbol{RB}}^{-1}\boldsymbol{C}_{\boldsymbol{RB}}$ and $\boldsymbol{M}_{\boldsymbol{RB}}^{-1}\boldsymbol{B}_{\boldsymbol{T}}$ can be determined, but the matrices $\boldsymbol{M}_{\boldsymbol{RB}}$, $\boldsymbol{C}_{\boldsymbol{RB}}(\boldsymbol{\nu})$ and $\boldsymbol{B}_{\boldsymbol{T}}$ cannot be obtained separately [50]. The parameters in these matrices are badly identifiable. Determining one of the matrices beforehand, using other methods, allows us to identify the remaining parameters. In literature, most of the time the inertia terms in matrix $\boldsymbol{M}_{\boldsymbol{RB}}$ are determined using slender-body theory [71], CFD calculations, empirical formula or captive model tests. The thruster configuration matrix $\boldsymbol{B}_{\boldsymbol{T}}$ contains only geometric information and can be determined by direct measurement.

Regardless of the identifiability example above, the estimated parameter vector $\tilde{\boldsymbol{\theta}}$ may still deviate from the true parameter vector $\boldsymbol{\theta}$ as a result from simultaneous drift of parameters. This happens when parameters can change their value in relation to each other, without impacting the output of the model. Identified numerical values for parameters can thus result in correct mathematical fits but have no physical meaning [35]. The drift of nonlinear parameters, also called multicollinearity, is commonplace in SI and can be shown statistically by checking the correlation between parameters when performing SI using difference methods [50].

Multicollinearity of parameters can be diminished by removing parameters with strong linearly dependence. Direct modification of the model in this way is a effective method to reduce the multicollinearity but comes at the cost of possible under-modelling. Parameter drift may also be reduced by rewriting the model using a difference method [51]. The input and output measurements are rewritten, but the parameters that must be identified remain the same. The new structure of the model may be less sensitive to multicollinearity. If possible, it is valuable to perform measurements where parts of the system are not excited, to identify the parameters for a subsystem first.

3.8. Maneuvering Model of DASh

Based on the considerations in previous sections, a maneuvering model is proposed that is complex enough to capture the dynamics of the vessel, but compact enough to be identifiable with SI methods. The model presented here is based on the vectorial maneuvering model [23] and uses the rigid-body dynamics from equation (3.1).

3.8.1. Linear and Non-Linear Hydrodyamic Damping

The hydrodynamic excitation terms are modelled as a resultant force when the hull moves through the water. Analogous to Fedyaevsky and Sobolev [21] and Norrbin [54], the following non-linear representation for the hydrodynamic reaction forces is used, based on a truncated Taylor series:

$$\begin{aligned} X_{hyd} &= X_{\dot{u}}\dot{u} + X_u u + X_{|u|u}|u|u + X_{uuu}u^3 \\ Y_{hyd} &= Y_{\dot{r}}\dot{r} + Y_{\dot{v}}\dot{v}Y_v v + Y_r r \\ &\quad + Y_{|v|v}|v|v + Y_{|v|r}|v|r + Y_{|r|v}|r|v + Y_{|r|r}|r|r \\ N_{hyd} &= N_{\dot{r}}\dot{r} + N_{\dot{v}}\dot{v} + N_v v + N_r r + N_{|v|v}|v|v \\ &\quad + N_{|v|r}|v|r + N_{|r|v}|r|v + N_{|r|r}|r|r \end{aligned} \tag{3.19}$$

All parameters $X_\bullet$, $Y_\bullet$ and $N_\bullet$ follow the SNAME notation. The absolute operator is introduced in the second order modulus terms e.g. $|u|u$, such that the sign of a quadratic term is preserved and thus damping terms always act in the opposite direction of motion. The non-linear representation is valid for all feasible vessel velocities.

The terms $X_{\dot{u}}$, $Y_{\dot{r}}$, $Y_{\dot{v}}$, $N_{\dot{r}}$ and $N_{\dot{v}}$ are referred to as added mass which model the fluid inertia forces by adding a directionally dependent virtual mass. The linear parameters X_u,Y_v,Y_r,N_v and N_r capture the linear effect of potential damping forces in calm waters.

The viscous damping forces, normally calculated in surge direction as

$$X_{visc} = -\frac{1}{2}\rho S C_f(u)|u|u \tag{3.20}$$

where ρ is the water density, X_{visc} the viscous damping force, S the wetted surface of the hull, C_f is the dimensionless flat plate friction coefficitent which depends on the Reynolds number[2]. Both the viscous damping forces and damping due to vortex shedding are captured in the regression by the quadratic terms $X_{|u|u}$, $Y_{|v|v}$, $Y_{|r|r}$, $N_{|v|v}$ and $N_{|r|r}$ parameters.

3.8.2. Cross-Flow Drag

The more complex phenomena of cross-flow drag induce nonlinear lift forces in sway direction and moments in yaw if the vessel is has a sway velocity or yaw rate $|v| > 0$, $|r| > 0$. The lift forces are determined by integration of the drag forces along the length of the vessel

$$\begin{aligned} Y_{cross} &= -\frac{1}{2}\rho \int_{-\frac{L_{WL}}{2}}^{\frac{L_{WL}}{2}} T(x) C_d^{2D}(x)|v + xr|(v + xr)dx \\ N_{cross} &= -\frac{1}{2}\rho \int_{-\frac{L_{WL}}{2}}^{\frac{L_{WL}}{2}} T(x) C_d^{2D}(x) x|v + xr|(v + xr)dx \end{aligned} \tag{3.21}$$

Where $C_d^{2D}(x)$ is two dimensional cross-flow drag coefficient for the cross section at position x. An approximate constant value for C_d^{2D} based on vessel breadth B and draft T, can be found in the Hoerner's curve [34] in figure 3.6. Solving the integrals of equation (3.21), the lift forces consists of quadratic terms and coupled terms in sway and yaw. The coupled terms depend on the product of $|v|r$ and $|r|v$, and therefore the cross-flow drag forces are captured by curve fitting to the coupled second order modulus terms

$$\begin{aligned} Y_{cross} &= Y_{|v|v}|v|v + Y_{|v|r}|v|r + Y_{|r|v}|r|v + Y_{|r|r}|r|r \\ N_{cross} &= N_{|v|v}|v|v + N_{|v|r}|v|r + N_{|r|v}|r|v + N_{|r|r}|r|r \end{aligned} \tag{3.22}$$

which are already included in equation (3.19).

The model of hydrodynamic forces presented here is not able to capture the hydrodynamic forces from flow phenomena separately. Instead terms in the regression simultaneously represent various forces resulting for different phenomena e.g. the quadratic terms model viscous damping and cross-flow drag. As such, the model parameters have no clear physical meaning in what forces they represent.

[2]The Reynolds number is a dimensionless quantity that predicts the occurrence of certain flow patterns, based on flow velocity and fluid viscosity.

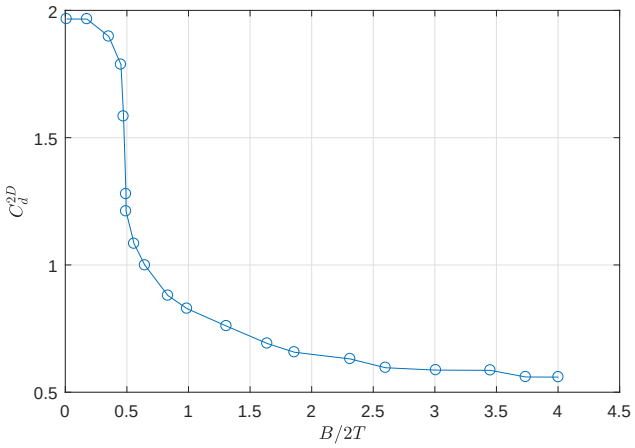

Figure 3.6: Hoerner's two dimensional cross-flow drag coefficient C_d^{2D}, based on vessel geometry.

3.9. Complete Maneuvering Model

With the addition of the hydrodynamic added mass, the rigid body inertia matrix and Coriolis and centripetal matrix are expanded to account for the extra inertia. The model given by (3.1) and (3.19) can elegantly be rewritten in a vectorial setting

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{3.23}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{B}\boldsymbol{u} + \boldsymbol{\omega} \tag{3.24}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{\kappa} \tag{3.25}$$

Modelling noise $\boldsymbol{\omega}$ and measurement noise $\boldsymbol{\kappa}$ is assumed to be white noise. The resulting inertia matrix is

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix} + \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -N_{\dot{v}} & -N_{\dot{r}} \end{bmatrix} \tag{3.26}$$

and Coriolis and centripetal matrix

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB} + \boldsymbol{C}_A = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & Y_{\dot{v}} v + Y_{\dot{r}} r \\ 0 & 0 & X_{\dot{u}} u \\ -Y_{\dot{v}} v - Y_{\dot{r}} r & -X_{\dot{u}} u & 0 \end{bmatrix} \tag{3.27}$$

The damping matrix $\boldsymbol{D}$ consists of a linear part $\boldsymbol{D}_L$

$$\boldsymbol{D}_L = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} \tag{3.28}$$

and a non-linear part $\boldsymbol{D}_{NL}$ that is dominant at high speeds

$$\boldsymbol{D}_{NL}(\boldsymbol{\nu}) = \begin{bmatrix} -X_{|u|u}|u| - X_{uuu}u^2 & 0 & 0 \\ 0 & -Y_{|v|v}|v| - Y_{|r|v}|r| & -Y_{|v|r}|v| - Y_{|r|r}|r| \\ 0 & -N_{|v|v}|v| - N_{|r|v}|r| & -N_{|v|r}|v| - N_{|r|r}|r| \end{bmatrix} \tag{3.29}$$

$$\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D}_L + \boldsymbol{D}_{NL}(\boldsymbol{\nu}) \tag{3.30}$$

The output matrix is defined as

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}^{3\times3} & \boldsymbol{0}^{3\times3} \end{bmatrix} \tag{3.31}$$

The hydrodynamic model presented in this section does not model frequency dependent wave drift, dissipation and potential damping. For the modelling of DASh this is a valid assumption considering there is no explicit wave excitation in the water basin in which it operates. Also, the surge dynamics are decoupled from the sway and yaw dynamics under the assumption that the hull is symmetric about the xy-plane. The influence of current is also neglected. It is however relatively simple to include the current velocity $\boldsymbol{u}_c$ in the model by rewriting equation (3.23) using the relative relative velocity vector $\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c$ instead of $\boldsymbol{\nu}$. This is valid assuming the ocean currents are constant and irrotational.

3.10. Observer Design for Navigation

The measurable system output $\boldsymbol{y}(t)$ consists of the x and y position and the heading angle ψ. To estimate unmeasured states, an observer is employed based on the Extended Kalman Filter (EFK). If the system is observable, the EKF can reconstruct unmeasured states and removing noise from state estimates. Observability and Controllability of the vessel model is proven in [52].

Let the continuous time model equations (3.23) be rewritten to the form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\omega} \tag{3.32}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{\kappa} \tag{3.33}$$

The implementation of a discrete time EKF consists of a prediction and update step. Let the state estimate $\hat{\boldsymbol{x}}(0) = x_0$ and error covariance matrix $\hat{\boldsymbol{P}}(0) = \boldsymbol{I}$ be the systems initial conditions. The sampling period is denoted by T_s. Let $\boldsymbol{J}(\hat{\boldsymbol{x}})$ be the Jacobian of the system. The prediction step in the discretized system using a simple Euler integration becomes

$$\bar{\boldsymbol{x}}(k) \approx \hat{\boldsymbol{x}}(k-1) + T_s \cdot f(\hat{\boldsymbol{x}}(k-1), \boldsymbol{u}(k-1)) \tag{3.34}$$

$$\boldsymbol{\Phi}(k) \approx \boldsymbol{I} + T_s \boldsymbol{J}(\hat{\boldsymbol{x}}) \tag{3.35}$$

$$\bar{\boldsymbol{P}}(k) = \boldsymbol{\Phi}(k)\hat{\boldsymbol{P}}(k-1)\boldsymbol{\Phi}^T(k) + T_s^2 \boldsymbol{Q} \tag{3.36}$$

the predicted state $\bar{\boldsymbol{x}}(k)$ and error covariance $\bar{\boldsymbol{P}}(k)$ are updated after receiving the most recent measurement $\boldsymbol{y}(k)$

$$\boldsymbol{K}(k) = \bar{\boldsymbol{P}}(k)\boldsymbol{H}^T[\boldsymbol{H}\bar{\boldsymbol{P}}(k)\boldsymbol{H}^T + \boldsymbol{R}]^{-1} \tag{3.37}$$

$$\hat{\boldsymbol{x}}(k) = \bar{\boldsymbol{x}}(k) + \boldsymbol{K}(k)[\boldsymbol{y}(k) - \boldsymbol{H}\bar{\boldsymbol{x}}(k)] \tag{3.38}$$

$$\hat{\boldsymbol{P}}(k) = [\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}]\bar{\boldsymbol{P}}^T(k)[\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}]^T + \boldsymbol{K}(k)\boldsymbol{R}\boldsymbol{K}^T(k) \tag{3.39}$$

Where the diagonal, positive definite, design matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ represent the variance of model and measurement noise. The state estimate $\hat{\boldsymbol{x}}(k)$ contains the filtered position $\hat{\boldsymbol{\eta}}$ and estimated body-fixed velocities $\hat{\boldsymbol{v}}$. This observer for the non linear vessel model will be used to gather velocity measurements for the system identification and model-based control in the second part of this report.

4

System Identification

In this chapter the unknown parameters of the maneuvering model of DASh are identified. The vessel model provided in chapter 3 was constructed to have maximum parameter identifiability while not under modelling its dynamics. Without the access to a towing tank, the system parameters matrices $\boldsymbol{M}$, $\boldsymbol{C}(\boldsymbol{\nu})$ and $\boldsymbol{D}(\boldsymbol{\nu})$ and the propeller thrust are identified by direct measurement or by using Least Squares and unconstrained non-linear parameter estimation algorithms that minimizing the prediction error between system output measurements and model output.

4.1. Decoupled SI approach

A effective method to reduce parameter drift and bad identifiability of parameters, is to decouple the planning problem into multiple subproblems. In order of appearance, the SI is decoupled into an identification of

1. inertia parameters
2. propeller thrust
3. surge parameters
4. sway and yaw parameters

The respective parameters to be identified are captured in parameter vectors $\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \boldsymbol{\theta}_3 & \boldsymbol{\theta}_4\end{bmatrix}^T$. Their contents are displayed in in table 4.1. Parameter vectors found in earlier parts are held constant when used in later parts. This allows is to identify the model step by step. This approach is only possible if parameters can be isolated when performing experiments. This is indeed the case for the maneuvering model, as will be shown in section 4.5. Parameters in the rigid-body Mass matrix will be determined

Table 4.1: Vessel parameters vectors to be identified.

Decoupled subsystem	Parameter Vector	Parameters
Inertia	$\boldsymbol{\theta}_1$	m, I_z, x_g
Propeller Thrust	$\boldsymbol{\theta}_2$	T_{nn}
Surge	$\boldsymbol{\theta}_3$	X_u, X_{uu}, X_{uuu}, $X_{\dot{u}}$
Sway and Yaw	$\boldsymbol{\theta}_4$	Y_v, $Y_{\|v\|v}$, $Y_{\|r\|v}$, Y_r, $Y_{\|v\|r}$, $Y_{\|r\|r}$, $Y_{\dot{v}}$, $Y_{\dot{r}}$ N_v, $N_{\|v\|v}$, $N_{\|r\|v}$, N_r, $N_{\|v\|r}$, $N_{\|r\|r}$, $N_{\dot{v}}$, $N_{\dot{r}}$

4.2. Estimation of Body-Fixed Velocities

The parameters that need to be identified are part of equation (3.24), which is a non-linear differential equation in $\boldsymbol{\nu}$. Using the observer, the measurements $\boldsymbol{y} = \begin{bmatrix}x & y & \psi\end{bmatrix}^T$ are filtering with using a simplified version of the EKF from section 3.10. The simplified model has the Coriolis and non-linear damping matrix set to zero. The inertia matrix and linear damping matrices reduced to diagonal matrices $\boldsymbol{M} = m\boldsymbol{I}$ and $\boldsymbol{D}_L = -10\boldsymbol{I}$,

Table 4.2: Numerical values for the inertia parameters $\boldsymbol{\theta}_1$

Parameter	Value
m	39.28
I_z	4.76
x_g	0.0585

respectively. The resulting EKF acts as a low-pass filter using the linearized vessel dynamics, which returns an estimate of the body-fixed velocities $\hat{\boldsymbol{\nu}} = \begin{bmatrix} \hat{u} & \hat{v} & \hat{r} \end{bmatrix}$. As equation (3.23) is just a non linear mapping of $\boldsymbol{\nu}$ to $\dot{\boldsymbol{\eta}}$, the identification is performed using just the body-fixed velocity estimate $\hat{\boldsymbol{\nu}}$ and equation (3.24).

4.3. Identification of Inertial Parameters

Vessel mass m was determined by direct measurement. The location of the COG was determined by measuring the mass at two support struts using two scales. Solving for the equilibrium of moments returns the position x_{COG}, on the longitudinal vessel axis. Movable weights in the vessel are used to position the COG midships. The location of the center of flotation x_{CF} at the centroid of the waterline is obtained by direct measurement. See figure 2.2 for the definition of the lengths. The relative location of the COG in the body-fixed coordinates equals $x_g = x_{CF} - x_{COG}$. The rotational inertia I_z about the vertical axis is approximated as if the vessel is a slender rod of homogeneous density

$$I_z = \frac{1}{12} m L^2 \tag{4.1}$$

The numerical values of the inertial parameter vector $\boldsymbol{\theta}_1$ is displayed in table 4.2.

4.4. Propeller Identification

For identification of the propeller, parameter vector $\boldsymbol{\theta}_2$ must be estimated. Propeller thrust is normally estimated using open water propeller diagrams of systematic propeller series like the Wageningen B series, Kaplan Series or (M)AU Series. These propeller series have been extensively tested in open water conditions and provide empiric solutions for thrust estimation in the form of the thrust coefficient $K_t(J)$, as shown in equation (3.8).

The thrust exerted by the propeller on the vessel depends on the flow speed of the fluid entering it. Due to the absence of a towing tank, the advance of DASh speed cannot be superimposed on the model. It is neither possible to impose a constant flow speed of the fluid in which the propeller is suspended. As such, the parameter T_{nu} that relates the advance speed, propeller rotation speed and the thrust, cannot be determined.

However, it is possible to measure propulsion forces if the vessel is stationary In shipping these measurements are called bollard pull tests. Using a scale, rope and pulley system detailed in appendix A, the tension in the line as a result of propulsion forces can be measured if the vessel is stationary. The approximate formula for propeller thrust at low speeds is fitted

$$T = T_{nn} n^2 \tag{4.2}$$

Note that the decrease in thrust due to non zero inflow speed is neglected by omitting the unidentifiable term $T_{nu} nu$. The input is thus only correctly modelled at low velocities where its influence is small. As a result, the thrust decrease due to positive surge speed will instead be partly captured by higher order damping terms e.g. X_u, $X_{|u|u}$, X_{uuu} obtained using System Identification.

4.4.1. Propeller Identification Results

The measured thrust at constant rotation speeds is shown in figure 4.1. Using Least Squares, T_{nn} was identified to be $T_{nn} = 6.2941e-06$, assuming both propellers give the same amount of thrust at the same rotation speed. The normalized root mean squared error (NRMSE) of measurements y is defined as

$$NRMSE = \frac{100\%}{y_{max} - y_{min}} \sqrt{\frac{1}{k} \sum_{i=1}^{k} (y - \hat{y})^2} \tag{4.3}$$

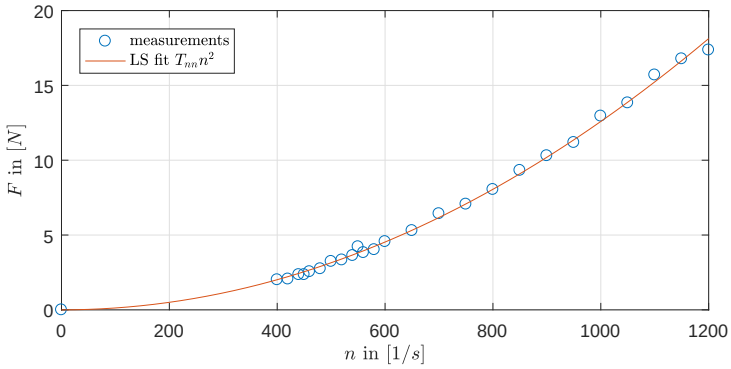

Figure 4.1: The bollard pull force measurements.

where y is the measured value and $\hat{y}$ the estimated value at time or experiment k. A normalization term subtracts the highest measured output y_{max} from the smallest measured output y_{min} in its denominator. The LS fit for the thrust measurements has a low NRMSE of 0.26%. The behavior of the thrust at zero advance speed is well modelled using the quadratic equation (4.2) based on visual inspection of figure 4.1 and the low NRMSE.

4.5. Identification of Hydrodynamic Surge Parameters

The identification methods presented here first decouples the surge state from the sway and yaw states. The vessel is straight-line stable, ensuring it sails in surge motion given positive thrust. The sway and yaw remain small $v \approx r \approx 0$ and the vessel dynamics of equation (3.23) reduce to the a 1 DOF model

$$(m - X_{\dot{u}})\dot{u} - X_u u - X_{|u|u}|u|u - X_{uuu}u^3 = 2T_{nn}n^2 \tag{4.4}$$

As model test can be performed in pure surge direction, this procedure has successfully decoupled the surge dynamics and its parameters related to surge. Note there is no linear relation between any parameter, avoiding parameter drift.

For a constant propeller rotation speed n, the surge speed of the vessel will stabilize at its terminal velocity. A series of terminal velocity test have been performed. Since $\dot{u} = 0$ at terminal velocity, the parameters X_u, $X_{|u|u}$ and X_{uuu} are identified solving the linear system of the form $\boldsymbol{Ax} = \boldsymbol{b}$ for a set of k terminal velocity measurements $u_{i=1\ldots k}$ at constant propeller rotation speed $n_{i=1\ldots k}$

$$\begin{bmatrix} u_1 & u_1|u_1| & u_1^3 \\ \vdots & \vdots & \vdots \\ u_k & u_k|u_k| & u_k^3 \end{bmatrix} \begin{bmatrix} X_u \\ X_{|u|u} \\ X_{uuu} \end{bmatrix} = \begin{bmatrix} -2T_{nn}n_1^2 \\ \vdots \\ -2T_{nn}n_k^2 \end{bmatrix} \tag{4.5}$$

Since $\boldsymbol{x}$ is the only unknown, a Least Squares solution is given as $\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. These values are then used as an initial guess in a non-linear optimization problem that identifies the parameter vector $\boldsymbol{\theta}_3 = [X_u \;\; X_{|u|u} \;\; X_{uuu} \;\; X_{\dot{u}}]$. The System Identification Toolbox of Matlab is used to estimate $\boldsymbol{\theta}_3$ by solving an unconstrained nonlinear optimization problem with cost function

$$V = \sum_{k=1}^{K}\left(\frac{1}{N}\sum_{t=1}^{N}\boldsymbol{e}^T(t,\bar{\boldsymbol{\theta}},k)\boldsymbol{W}e(t,\bar{\boldsymbol{\theta}},k)\right) \tag{4.6}$$

where $\boldsymbol{W}$ is a weighing scalar and $\boldsymbol{e}(t,\bar{\boldsymbol{\theta}},k)$ is the prediction error for the k^{th} experiment defined as

$$\boldsymbol{e}(t,\bar{\boldsymbol{\theta}},k) = \boldsymbol{u}(t,k) - \hat{y}(t,\bar{\boldsymbol{\theta}},k) \tag{4.7}$$

and the model output $\hat{y}(t,k)$ is calculated using

$$\bar{\boldsymbol{y}}(t,\bar{\boldsymbol{\theta}},k) = \int_{t_0}^{t}\left(\frac{1}{m - X_{\dot{u}}}(X_u u + X_{|u|u}|u|u + X_{uuu}U^3 + 2T_{nn}n_k^2)\right)dt + u_{0,k} \tag{4.8}$$

Table 4.3: Surge parameters fitted from terminal velocity data using LS.

Parameter	Value
X_u	-1.66
$X_{\|u\|u}$	-9.49
X_{uuu}	1.77

for the same inputs n and initial surge state $u_{0,k}$ for experiment k. Each run started at zero surge speed $u_{0,k} = 0$. The trust region reflective algorithm of the Matlab Optimization Toolbox is used to solve the non-linear optimization problem. The search is terminated after 50 iterations of the optimization algorithm. Termination also occurs at the termination tolerance on the loss function of $10e^{-5}$ or at the termination tolerance on the estimated parameter values reaches $10e^{-6}$. Convergence of this optimization is not trivial. Bad initializations of $\boldsymbol{\theta}_3$ resulted in physically meaningless fits with very large added mass terms. A good initial guess for X_u, $X_{|u|u}$, X_{uuu} has shown to increase convergence of the fit to rational values of $X_{\dot{u}}$.

4.5.1. Surge Parameters Identification Results

A series of constant propeller rotation speed test were performed for various n. In figure 4.2 the forward terminal velocities are plotted against the amount of forward thrust.

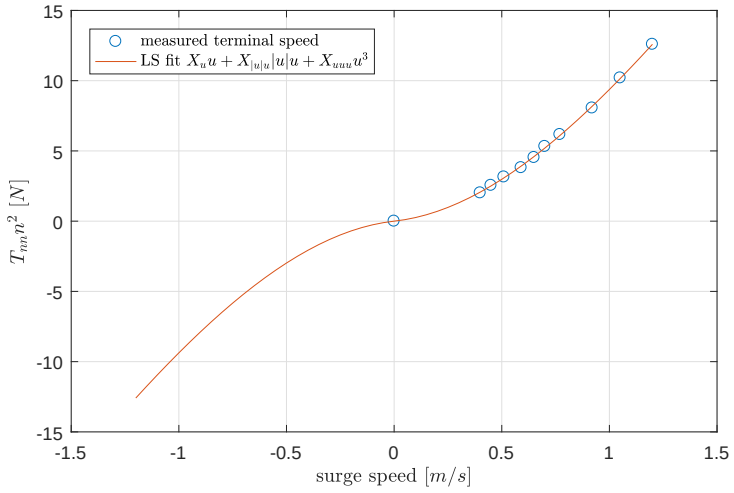

Figure 4.2: The terminal surge velocity for various amounts of forward thrust.

Using the Least Squares fitting approach the surge parameters for X_u, $X_{|u|u}$ and X_{uuu} were determined. Their numerical values are displayed in table 4.3.

The non-linear optimization was ran on a set of five experiments $K = 5$. The first three consists of pure surge motion in which the terminal velocity is reached under constant propeller rotation speed. Experiment 4 and 5 had their propeller rotation speed stepwise varied, resulting in acceleration and deceleration of the vessel. These experiments are more rich, in the sense that it contains more information on the surge dynamics in which $\dot{u} \neq 0$. Experiment 1 to 4 were used as a training set on which the parameters were identified. Experiment 5 is the test data set, on which the quality of the fit can be accessed. To asses the quality of the estimated model the Variance Accounted For (VAF) for each state of each individual experiment k is calculated as

$$VAF(\boldsymbol{y}(t), \bar{\boldsymbol{y}}(t, \bar{\boldsymbol{\theta}}, k)) = \left(1 - \frac{\frac{1}{N}\sum_{t=1}^{N} ||\boldsymbol{y}(t,k) - \bar{\boldsymbol{y}}(t, \bar{\boldsymbol{\theta}}, k)||_2^2}{\frac{1}{N}\sum_{t=1}^{N} ||\boldsymbol{y}(t,k)||_2^2}\right) \cdot 100\% \tag{4.9}$$

The measured system output $\boldsymbol{y}(t,k)$ and model output $\hat{y}(t, \bar{\boldsymbol{\theta}}, k)$ with the optimized parameters and the VAF for each experiment, are shown in figure 4.3. The prediction error is lower for a higher VAF, which has a maximum value of 100%. The optimized parameter values are given in table 4.4.

Table 4.4: Optimized parameter vector $\boldsymbol{\theta}_3$ using Non-Linear Least Squares optimization

Parameter	Value
X_u	-0.46
$X_{\|u\|u}$	-13.04
X_{uuu}	3.46
$X_{\dot{u}}$	-5.10

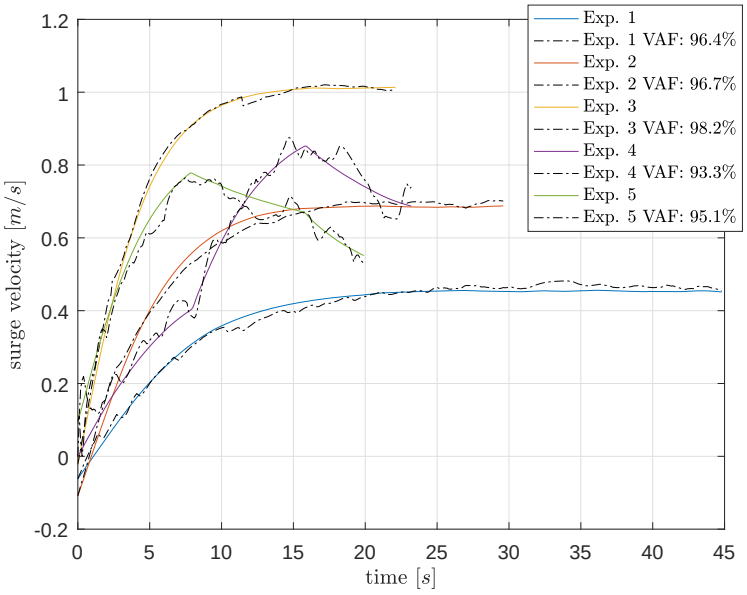

Figure 4.3: The surge experiments with measured system output $y(t,k)$ (black dotted) and model output $\bar{\boldsymbol{y}}(t,\tilde{\boldsymbol{\theta}},k)$ (colored) with estimated surge parameters.

Visual inspection of figure 4.3 shows a good fit between the dynamic surge behavior of the vessel and the estimated model. The VAF of all experiments, including the test data of experiment 5, exceeds values of 94%, indicating good alignment between the identified model and real surge dynamics. The surge speed under constant propeller thrust shown no large steady state error, confirming that the truncated Taylor series of the third order does not under-model damping in surge direction.

4.6. Identification of Hydrodynamic Sway and Yaw Parameters

In contrast to the surge dynamics, the sway and yaw dynamics cannot be decoupled from the vessel dynamics due to coupling terms in $\boldsymbol{C}(\boldsymbol{\nu})$. Instead a System Identification run is performed on the complete system dynamics with previously found parameters in $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_3$ held constant.

The experiments performed to excite the sway and surge dynamics are the so called zig-zag maneuvers, recommended by the ITTC. In the zigzag test, a vessel is brought to a constant surge velocity. A predetermined steering action, such as a rudder angle or azimuth angle, is executed until a certain change in the heading is achieved. From that moment the opposite steering angle is executed until the opposite heading angle is reached, and so on. Following this protocol, a vessel will 'zigzag' along the initial surge direction. A figure of the commanded steering angle and heading during a zigzag test is shown in figure 4.4. The propeller rotation speed is held constant during the test.

Although the zigzag test is not proven to be the an optimally designed experiment, it is chosen due to its widespread industry usage and obvious excitation of the sway and yaw dynamics. In [55] an example is provided wherein free running zigzag test were used to identify a vessel model, that was shown to also correctly

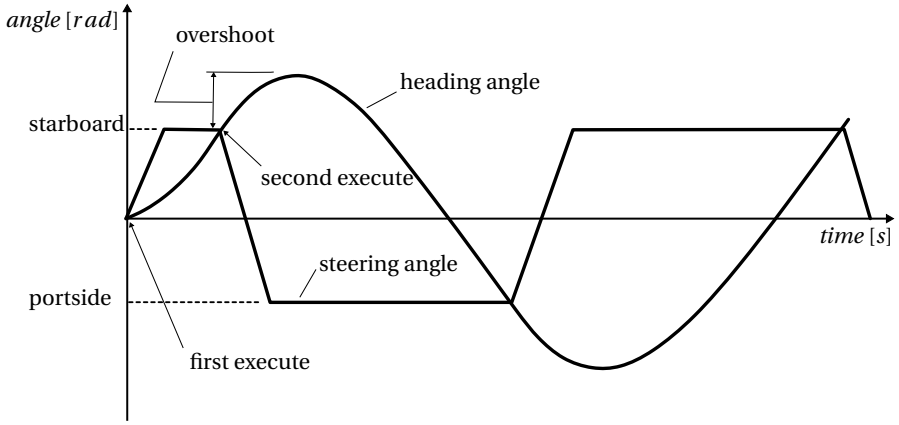

Figure 4.4: The commanded steering angle and heading during a zigzag test.

Table 4.5: Identified sway and yaw parameters of $\boldsymbol{\theta}_4$ using the Non-Linear Least Squares optimization for the zigzag experiments.

Parameter	Value	Parameter	Value
Y_v	-7.30	N_v	-2.75
$Y_{\|v\|v}$	-18.93	$N_{\|v\|v}$	16.58
$Y_{\|r\|v}$	-97.25	$N_{\|r\|v}$	-57.07
Y_r	-1.71	N_r	-4.80
$Y_{\|v\|r}$	-46.68	$N_{\|v\|r}$	-32.65
$Y_{\|r\|r}$	-61.91	$N_{\|r\|r}$	-81.04
$Y_{\dot{v}}$	-0.20	$N_{\dot{v}}$	-18.41
$Y_{\dot{r}}$	-2.31	$N_{\dot{r}}$	-3.59

predict steady-state turning of vessels. This increases the belief that zigzag test adequately excite sway and yaw dynamics at positive surge speeds, as identified parameters are valid for multiple turning maneuvers. However, in section 4.6.1 it is shown this is not the case for DASh.

Four zigzag experiments were performed at the constant propeller speeds in rpm $n = \begin{bmatrix}450 & 500 & 550 & 600\end{bmatrix}$. The set steering angle of both azimuth propellers were $10°/-10°$ and the heading angle to reach before changing the steering angle was $10°/-10°$. The zigzag experiment 2 at $n = 500$ is used as test data, experiment 1, 3 and 4 are used as the training data set ($K = 3$). The same cost function is minimized as defined in equation (4.6). However, now the model output is extended to $\boldsymbol{y}(t,k) = \begin{bmatrix}u & v & r\end{bmatrix}^T$ and the diagonal weighing matrix becomes

$$\boldsymbol{W} = \begin{bmatrix}0.1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{4.10}$$

The model output is now calculated as

$$\tilde{\boldsymbol{y}}(t,\tilde{\boldsymbol{\theta}},k) = \int_{t_0}^{t}\left(-\boldsymbol{M}^{-1}\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{M}^{-1}\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{M}^{-1}B\boldsymbol{u}(t,k)\right)dt + \boldsymbol{\nu}_{0,k} \tag{4.11}$$

The optimization is performed using the same optimization algorithm and termination conditions as used in section 4.5.

4.6.1. Sway and Yaw Parameter Identification Results

The identified parameters of $\boldsymbol{\theta}_4$ are shown in table 4.5. Various parameter initialization were used, but best convergence was achieved at $\boldsymbol{\theta}_4 = \boldsymbol{0}$. The model output of the identified model and the VAF per state are shown in figure 4.5, for each experiment.
Visual inspection shows a good fit between the surge, yaw and sway dynamic of the vessel and the estimated model. The VAF values for surge and yaw remain above 69% for all experiments. the VAF for sway is sig-

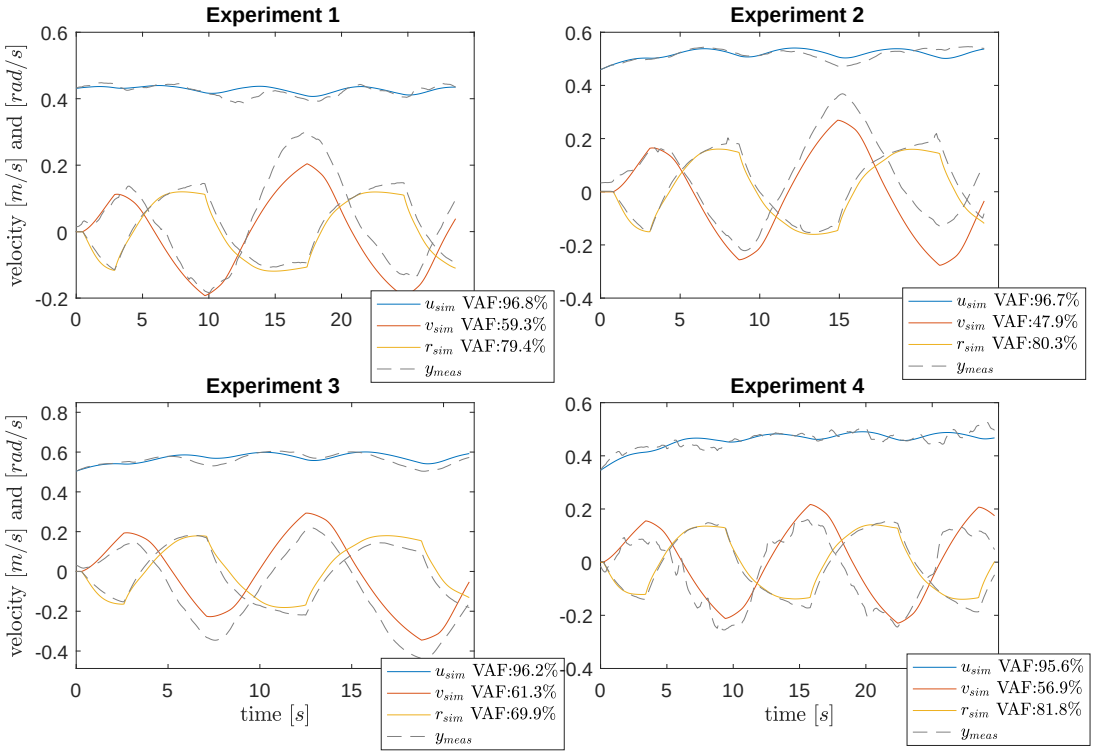

Figure 4.5: The zigzag experiments with measured system output $\boldsymbol{y}(t,k)$ and estimated model output $\bar{\boldsymbol{y}}(t,\hat{\boldsymbol{\theta}},k)$. The VAF is displayed per state of each experiment.

nificantly lower with values between 47.9% and 61.3%. The sway tends to diverge more quickly compared to surge or sway predictions. Increasing the weight on the sway in W results in only marginally increases of the VAF for sway, at the cost of yaw. The test data of experiment 2 shows increased divergence of the sway prediction after 14 seconds.

The fully identified model is used to predict a maneuver of which a trajectory is shown in figure 4.6. The model is unable to correctly predict future vessel positions over the long time horizon of the maneuver. The prediction of states $\bar{x}$, $\bar{y}$ and $\bar{\psi}$ diverge quickly after entering the first right turn at $t \approx 23$. Model output on the position and heading remain valid for a period of only a few seconds. The prediction of the velocity states $\bar{u}$, $\bar{v}$ and $\bar{r}$ are of similar quality compared to the zigzag tests.

Most likely, the bad prediction of $\bar{x}$, $\bar{y}$ and $\bar{\psi}$ is a result of the choice to perform identification on just the estimated body-fixed velocities $\hat{\boldsymbol{v}}$. The velocity states were obtained with a low-pass filter that uses a simplified vessel model, described in section 4.2. Filtering with this model, incorporated information of the simplified model into the velocity estimates, resulting in a poor estimation of the true system dynamics. Nevertheless their were reasons to do so. Firstly, a direct measurement of body-fixed velocities was not available. Secondly, the parameter estimation algorithm was not able to converge when identifying the 6 DOF model based on the noisy measurements of x, y and ψ.

Although imperfect, the identified model can still be used in model-based controllers such as Model Predictive Control (MPC), as long as the MPC prediction horizon is small. For prediction horizons up to ± 4 seconds the model returns and adequate prediction of the state, in longer prediction horizons the model will provide bad long term predictions, resulting in worse control inputs. Keeping the prediction horizon below 4 seconds allows NMPC to stabilize and control the vessel, as will be shown in chapter 7.

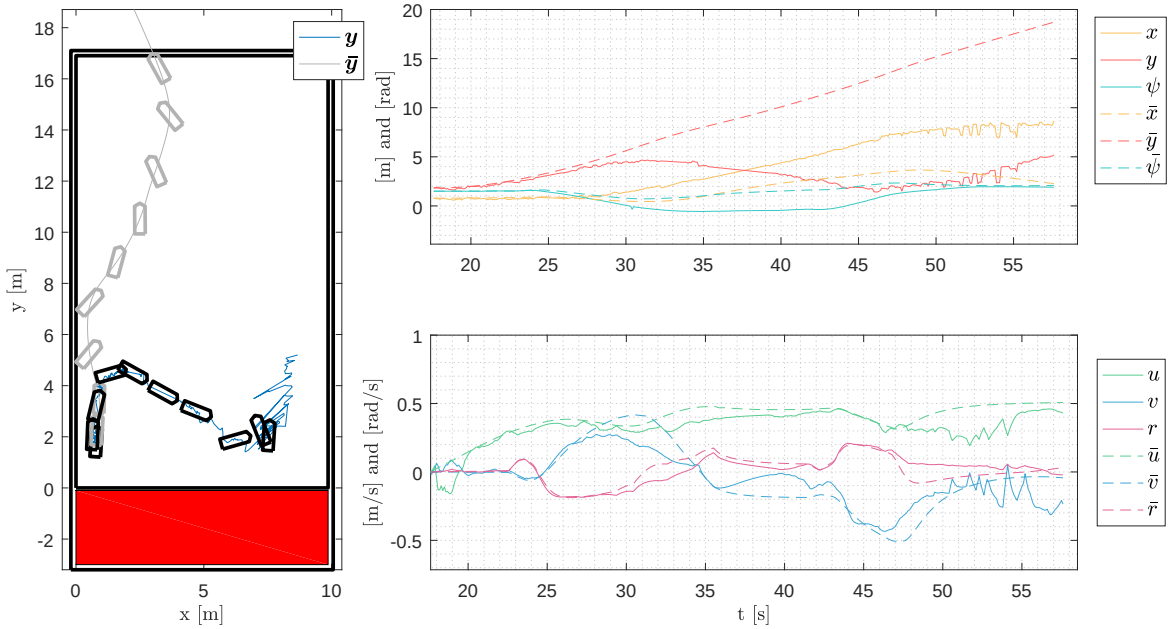

Figure 4.6: Measured system output of a turning maneuver and the estimated model output. (left) The sailed and predicted trajectory. (top right) Measured and predicted position. (bottom right) Measured and predicted body-fixed velocities.

4.7. Conclusion

In this chapter all propeller and hydrodynamic parameters of the nonlinear vessel model of equation (3.23) have been identified. The identification consisted of four steps. First, the inertia matrix was obtained. Secondly, bollard pull tests were performed to obtain the thrust characteristics at zero advance speed. It is assumed the estimated propeller thrust remains valid at low advance speeds. Thirdly, the surge dynamics were decoupled from the vessel model, allowing surge parameters to be identified individually, decreasing the chance of parameter drift. An initial guess for hydrodynamic surge damping terms was obtained by applying a Least Squares fit on terminal velocities for a set measurements performed at constant forward thrust. Using non-linear optimization, the prediction error between a training set of measured system outputs and model output. Surge parameters were identified using a training data set of 4 surge experiments. The test data set consisted of one surge experiment that scored a Variance Accounted For (VAF) value of 95.1%. Visual comparison of measured and modelled output in figure 4.3 and the high value for the VAF imply good identification of surge dynamics. Lastly sway and yaw parameters were determined using non-linear optimization by performing free running zigzag tests. A training data set consisted of 3 zigzag experiments at different surge velocities. A test data set consisted of 1 zigzag test that scored a VAF of 47.9% for sway and 80.3% for yaw. Although the relatively low VAF, figure 4.5 shows the model is able to capture the sway and yaw behavior.

The output prediction of the turning maneuver shows the identified model is unable to correctly predict states x, y and ψ over a long time horizon. However, the identified model can be used for model-based control methods such as MPC, as long as its prediction horizon is shorter than ± 4 seconds. Otherwise, the state estimates will diverge quickly from the true system.

The identified vessel model will be used for the guidance and control systems in the following chapters.

II

Model-Based Guidance & Control

5

Kinodynamic Sampling-Based Planning

In previous chapters the hardware of DASh was presented and a mathematical model of its dynamics was identified. In this chapter, a kinodynamic sampling-based path planner is proposed to be used as a guidance system. In combination with a motion controller, the planner enables the vessel to navigate its environment autonomously. To start, the planning problem for DASh is defined after which relevant sampling-based path planning principles are introduced on which the framework in chapter 6 will expand. At last, the choice of planner to be implemented in DASh is detailed.

5.1. DASh Planning Problem

The movement of DASh is constrained by obstacles in its environment and the capabilities of its azimuth thrusters and its dynamics, of which the vectorial representation is repeated here from chapter 3

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{5.1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{u} + \boldsymbol{\omega} \tag{5.2}$$

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\nu} \end{bmatrix} \tag{5.3}$$

The location and heading of the vessel is captured in $\boldsymbol{\eta} = [x \quad y \quad \psi]^T$ and the body-fixed surge, sway and yaw velocity in $\boldsymbol{\nu} = [u \quad v \quad r]^T$. Modelling noise $\boldsymbol{\omega}$ is assumed to be a zero mean white noise process. A derivation of the complete model can be found in chapter 3. The equation of motions locally constrain the movement of DASh. Also, the propeller forces are finite and impose input constraints on the model.

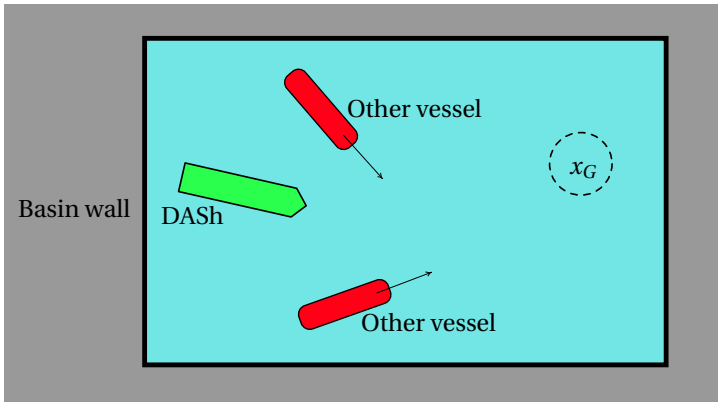

Figure 5.1: The planning environment of DASh planning problem. DASh is colored in green, static obstacles in gray and the dynamic obstacles in red.

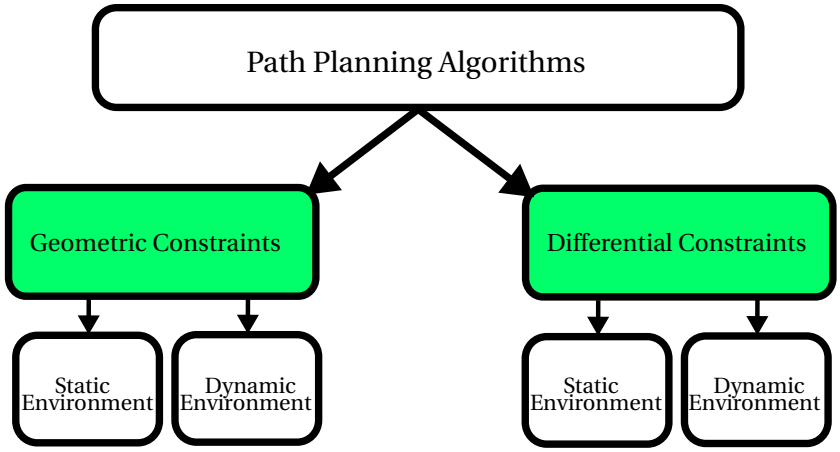

Figure 5.2: A classification of path planning algorithms.

A guidance system is responsible for the creation of a path between an initial state $\boldsymbol{x}_I$ and the goal state $\boldsymbol{x}_G$. Algorithms that achieve this goal are called path planners. The goal of DASh is to navigate the 10 by 20 meter basin at the Rotterdamse Droogdok Maatschappij (RDM) Campus. Inside the tank other stationary and moving vessels will be present. It is assumed the other vessels do not react to DASh and will not try to avoid collisions. A top view of the planning environment is given in figure 5.1.

5.2. Introduction to Path Planning

Important planning principles and terminology are introduced in this section. A direct focus is placed on sampling-based planners, as it is the most promising approach to solving kinodynamic planning problems.

5.2.1. Constraints in Planning

The complexity of planning problems is governed by the constrains a path has to satisfy. Let the object or robot for which a path is planned be referred to as the *agent*. If the movements of an agent are constrained by obstacles in the environment, a path must be planned under *geometric constraints*. If the allowable actions or transitions of the agent are strongly governed by its dynamics, a path must be planned under *differential constraints*. Planning under differential constraints is also referred to as *kinodynamic planning*. In case moving obstacles are present one must plan within a dynamic environment, else the environment is said to be static. An overview of planning algorithms based on these classifications is shown in figure 5.2.

The interested reader is encouraged to lookup Geometrically constrained planners such as the Reduced Visibility Graph planner in static environments [53], or generalized velocity obstacle planning for safely navigating amongst dynamic obstacles [74]. Differentially constrained planners are often based on the kinodynamic Rapidly-exploring Random Tree [48].

The complex spatial and dynamical constraints on the agents state pose the largest challenge in planning and distinguishes path planning from ordinary control problems. DASh and displacement vessels in general are strongly locally constrained by their dynamics. To illustrate; displacement vessels are slow to accelerate, decelerate and suffer from drift in corners due to their large inertia.

Surprisingly, kinodynamic planners are rarely implemented in autonomous vessels. The majority of literature on guidance and path planning for ships use a geometrically constrained planner [23, 31, 45, 69]. The paths are defined by way-points connected by collision free straight line segments. A path-following controller in combination with collision avoidance algorithm [67] is used to account for moving obstacles and the vessel dynamics during execution. These methods have no guarantee that the planned path is dynamically feasible or is optimal with respect to any metric other than path length.

5.2.2. Representation of Space

Broadly speaking, one wishes to minimize the dimension of the space in which to perform path planning to reduce the computational load. For a dynamic system like DASh, the minimal set of independent parameters describing the position and velocities of the agent is the state space X. The physical space in which the agent and obstacles reside is called the *workspace*, which is often a 2D space $W \in \mathbb{R}^2$ or 3D space $W \in \mathbb{R}^3$. The space the agent occupies at state $\boldsymbol{x}$ is denoted by $A(\boldsymbol{x})$. The space occupied by static and moving obstacles is called the obstacle region $O(t)$. The set of states that lead to collision is state space obstacle region X_{obs}, defined as

$$X_{obs} = \{ \boldsymbol{x} \in X \mid A(\boldsymbol{x}) \cap O(t) \neq \emptyset \} \tag{5.4}$$

The algebraic complement of X_{obs} is the free state space

$$X_{free} = X \setminus X_{obs} \tag{5.5}$$

Let a kinodynamic planner create a path for a differentially constrained agent with dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$. A *feasible* path is defined as a sequence of states $\tilde{\boldsymbol{x}} \in X_{free}$ accompanied by a sequence of control inputs $\tilde{\boldsymbol{u}} \in U$, where the action space U is the set of allowed control inputs. An example of a geometrically and differentially feasible path is shown in figure 5.3. Note that planning under differential constraints result in smooth paths, compared to the jagged geometrically constrained path, that would be harder to execute without tracking error.

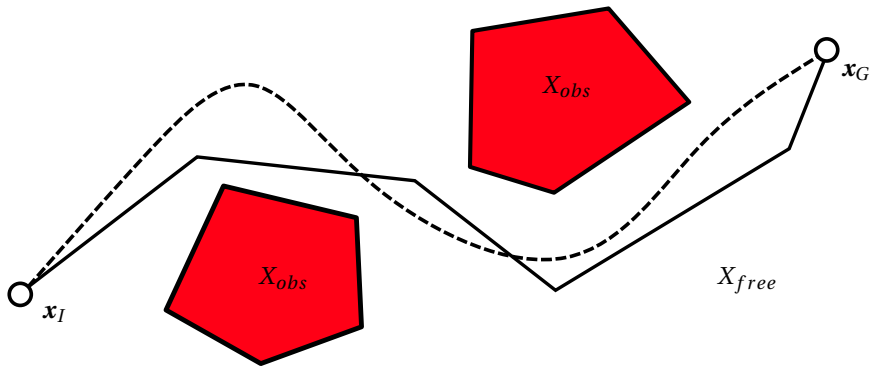

Figure 5.3: A geometrically constrained (solid) and differentially constrained (dotted) feasible path from x_I to x_G in the state space X.

5.3. Optimal Kinodynamic Planning

In the majority of planning problems, it is not only important to reach the goal state with a feasible path, but also minimize a performance measure such as travel time, path length or energy consumption. Given an initial state $\boldsymbol{x}(0) = \boldsymbol{x}_I$ and goal state $\boldsymbol{x}_G$, the optimal state and input trajectory, denoted by $\tilde{\boldsymbol{x}} : t \to X$ and $\tilde{\boldsymbol{u}} : t \to U$ connect $\boldsymbol{x}_I$ to $\boldsymbol{x}_G$ while minimizing a cost function $\Phi(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$. The planning problem can be rewritten as an optimization:

$$\begin{aligned} \min_{\tilde{\boldsymbol{x}} \in X_{free},\, \tilde{\boldsymbol{u}} \in U,\, T>0} \quad & \Phi(\tilde{x}, \tilde{u}, t) \\ \text{subject to} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)). \qquad \forall\, t \in [0, T] \\ & \boldsymbol{x}(0) = \boldsymbol{x}_I \\ & \boldsymbol{x}(T) = \boldsymbol{x}_G \end{aligned} \tag{5.6}$$

The trajectory duration is set by the decision variable T. The value of $\Phi(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, t)$ at the optimal solution is called the cost-to-go. The globally optimal solution to this optimization is extremely difficult to find. Without including the differential constraints, the optimization in (5.6) has been shown to quickly become *intractable* for state dimensions of 3 or higher due to the complex spatial constraint $\tilde{\boldsymbol{x}} \in X_{free}$. Planning for a polyhedral

agent with polyhedral obstacles in three dimensional space has been shown to be PSPACE-hard[1] in [59]. Due to the proven complexity, it is unreasonable to search for a solution (5.6) directly.

5.4. Kinodynamic Sampling-Based Planning

Sampling-based planning is the most popular approach for differentially constrained planning in high dimensions. Sampling-based planners explore the free state space by discretization. The connectivity of X_{free} is captured by (pseudo)randomly sampling states called vertices, and connecting them with edges using a steering method. A collision detection module checks if vertices or edges are collision free and if so, they are added to a graph (or tree). The inherent bias towards unexplored regions allow quick identification of the free state space. At the base of kinodynamic sampling-based planners stand the Rapidly-exploring Random Tree (RRT) [48] and the Probabilistic Roadmap (PRM) [42]. Over the years, many improved sampling-based planners have emerged, but they still share the same core routines that will be explained in the next section.

The counterpart to sampling-based planners are combinatorial path planning methods. Combinatorial approaches try to map the complete free state space, preserving the continuity of the space. In geometrically constrained planning problems of low dimension, combinatorial methods outperform sampling-based planners as they are able to reliably find minimum distance paths. However, the approach struggles in high dimensional spaces or when planning under differential constraints. For more information on combinatorial planning the reader is referred to [47].

5.4.1. Completeness

A path planner is called *complete* if it guarantees to find a path if one exist or correctly report that it does not. The guaranteed solution is a very desirable characteristic in path planners, but is not always present. A planning approach is said to be incomplete is it does not guarantee to find a solution, even though one exist. Forms of completeness in decreasing strength are:

- complete
- resolution complete
- probabilistic complete
- incomplete

Combinatorial methods are *complete*. This is due to their ability to capture X_{free} in a continuous and complete manner. Sampling-based methods capture the connectivity of the state space by sampling and building a graph G. If the vertices of G are *dense* in X_{free}, the connectivity is captured appropriately and a solution can be found. To obtain a dense graph, often many samples are needed and path planning algorithms require an increasingly longer runtime. If random sampling methods are used, sampling-based algorithms are *probabilistic complete*. This means that if a solution exist, the probability of find a feasible path converges to one as the amount of samples goes to infinity.

Planning methods that in some way discretize a space in the planning problem are *resolution complete*. This level of completeness guarantees to find a solution if that discretization has reached a certain level of resolution. Discretization of any space leads to a loss information. This may result in planners becoming incomplete, an example of which will be shown in section 6.6.

5.4.2. Kinodynamic Rapidly-exploring Random Tree Planner

The Kinodynamic RRT algorithm iteratively solves a string of routines to grows a graph G. Let a vertex be denoted by V and an edge by E. Inputs to the kinodynamic RRT algorithm are the initial state $\boldsymbol{x}_I$, free state space X_{free} and the steering time Δt. A circular goal region $X_G \in X_{free}$ is defined around $\boldsymbol{x}_G$ such that $X_G = \{ \boldsymbol{x} \in X_{free} \mid \|\boldsymbol{x} - \boldsymbol{x}_G\|_2 < r_G \}$ for a small scalar value r_G. Increasing the size of the goal state is a necessary condition to find a path as the chances of randomly sampling the goal state during planning are zero.

A graph G is initiated with a single vertex at $\boldsymbol{x}_I$. The following routines are then performed:

[1] PSPACE-hard is an expression to characterize the computational complexity of a problem

1. **State Sampling:** A state $\boldsymbol{x}_{rand}$ is randomly sampled in X_{free}.

2. **Nearest Neighbor:** Given $\boldsymbol{x}_{rand}$, this routine searches and returns the nearest vertex V_{near} in the graph G based on the used distance metric.

3. **Steering Method:** A steering method connects V_{near} to $\boldsymbol{x}_{rand}$ by a trajectory pair $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ over a time period of Δt seconds. If $\boldsymbol{x}_{rand}$ is not reached, a new node $\boldsymbol{x}_{new}$ is returned which has made maximal progress towards $\boldsymbol{x}_{rand}$.

4. **Collision Checking:** The trajectory pair is checked to be in X_{free}, using a collision checker. Only if the path is collision free, $\boldsymbol{x}_{new}$ and the trajectory pair are added to G as a vertex and edge, respectively. If the path is in collision, the algorithm returns to the state sampling routine.

5. **Goal Reached:** If $\boldsymbol{x}_{new}$ lies inside the goal region X_G a feasible path is contained in G and the algorithm is terminated. Graph search methods such as A* [32] or Dijkstra's algorithm [17] can be used to extract the path from G. If $\boldsymbol{x}_{new} \notin X_G$ the algorithm returns to the state sampling routine.

The extension of a graph towards $\boldsymbol{x}_{rand}$ using the kinodynamic RRT algorithm is shown in figure 5.4. Pseudo code for the kinodynamic RRT algorithm is provided in algorithm 1.

Algorithm 1 Kinodynamic Rapidly-exploring Random Tree

Input: Initial state $\boldsymbol{x}_I$, Goal region X_G, free state space X_{free}, action space U, integration time Δt
Output: Graph G

```
function BUILDRRT(x_I, x_G)
    G.init(x_I)
    for i ← 1 to K do
        x_rand ← SAMPLERANDOMSTATE()
        V_near ← NEARESTNEIGHBOR(x_rand, G)
        (x̃, ũ, x_new) ← STEERINGMETHOD(V_near, x_rand, Δt)
        if x̃ ∈ X_free and ũ ∈ U then
            G.addVertex(x_new)
            G.addEdge(V_near, x_new, x̃, ũ)
        end if
        if any V ∈ X_G then
            return G
        end if
    end for
end function
```

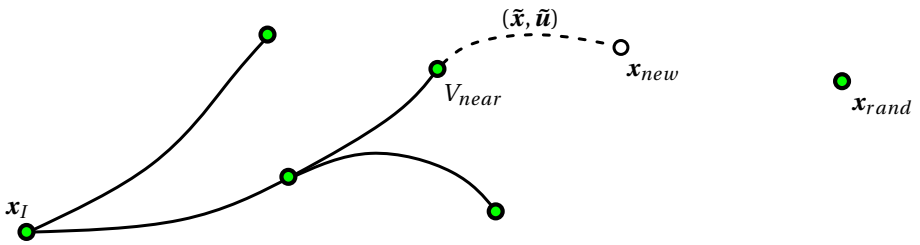

Figure 5.4: Extension of a graph G towards a randomly sampled node $\boldsymbol{x}_{rand}$ using a steering method resulting in the trajectory $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ ending at $\boldsymbol{x}_{new}$.

The kinodynamic RRT subroutines may be improved upon, depending on the agent's dynamics, state or workspace. Several improvements are discussed in the following sections.

5.4.3. State Sampling Methods

Sampling-based planners originally sampled states uniformly over the search space. The probability of drawing samples from a wide unexplored region is thus higher than in narrow regions, a phenomena called Voronoi

bias. This bias causes sampling-based methods to quickly explore the free state space.

Information about the environment can be used to sample states at a certain location. This is called informed sampling and is employed in several planners [3, 10, 60, 73]. The main purpose of biased sampling is to accelerate computation times. One of the commonly used biased sampling method is to replace a randomly sampled node by the desired goal state. The probability of occurrence is often chosen equal to 5%. As a result, information of the goal state allows the RRT to operate in more than just an exploratory manner, but be slightly guided to the goal state. As a result, less state samples are needed to reach te goal region. This method will also be used in the kinodyanmic RRT for DASh.

5.4.4. Nearest Neighbors

A steering method should connect the sampled state to the closest vertex in G. This vertex is called the nearest neighbor. To define the proximity of one state to another in absence of obstacles, a distance metric must be formulated. To have maximum expansion of the RRT, it is paramount to have a good distance metric and correctly connect nearest neighboring states. Bad distance metrics result in larger than necessary graphs, resulting in less optimal paths overall. For example, a good distance metric for a geometrically constrained planning problem in a two dimensional planar space would be the Euclidean distance. In this case, the straight line distance between two points in the state space is the shortest connection possible.

In the state space of dynamic agents no straight forward distance measures exist. An agent's state often consists of a set of translations, angles, velocities and accelerations. Combining these unrelated metrics into a single scalar distance requires arbitrary weights to be applied to all variables. The resulting scalar encodes no information about the constrained relationship between positions and velocity [30]. A correct distance metric is especially important in sampling-based planners, as it significantly improves the planners performance, as shown in [12]. An intuitive example of problems with distance in the state space is depicted in figure 5.5.

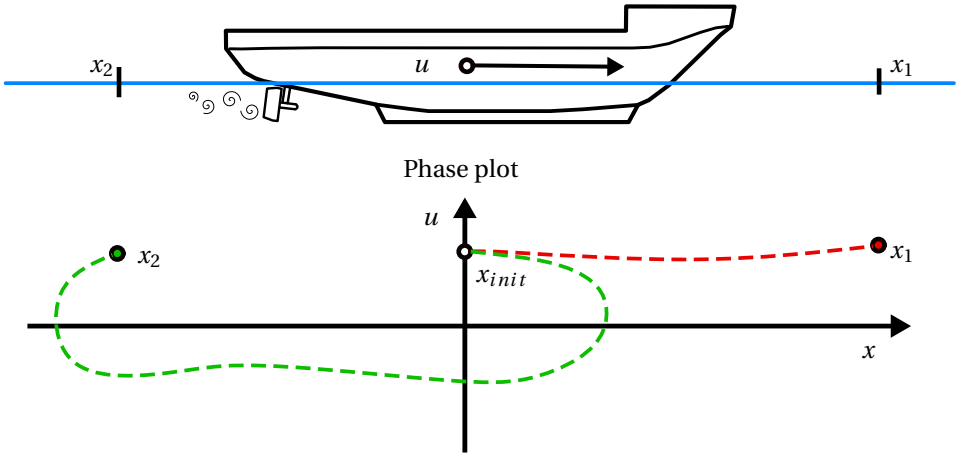

Figure 5.5: An example of distance in state space for a two dimensional vessel model $\dot{x} = u$. A vessel at x_{init} with positive surge speed u is connected to goal state x_1 (red) and x_2 (green). The straight-line (Euclidean) distance between x_{init} and both goal states is equal. However, x_1 is easier to reach, as it is located in front of the vessel.

Instead, the ideal distance metric for dynamic agents should be expressed in a cost necessary to travel between points in the state space. This distance metric is the cost-to-go, earlier introduced in section 5.3 [48]. Sadly, the computation of the true cost-to-go is as complicated as solving the original planning problem. As a best alternative, the optimal cost-to-go of the linearized system is used [30, 49]. Linearizing the dynamics allows for quick optimal cost computations. The improvement of the linearized cost-to-go above the Euclidean distance as a distance metric drops off with increasing non-linearities in the system.

A workaround where the computationally expensive cost-to-go were moved to an a priori phase were proposed in [6, 78] in which a learning algorithm estimates the optimal cost-to-go. The learned model is able to quickly return approximations of the optimal cost-to-go during planning, reducing the overall planning time. A similar approach using machine learning was used in [5] to quickly determine cost-limited reachability sets

in [5]. A planner using the cost-to-go approximations was implemented in real-time fashion for a quad rotor in a environment with static obstacles [4]. In chapter 6 the performance of the learned distance metric is compared to the Euclidean distance metric and a minimal curve length metric, when applied to the DASh planning problem.

5.4.5. Steering Methods

The steering methods is responsible for connecting two states. In geometric planning, this task is as easy as connecting the states with a straight line to get the shortest path. If differential constraints are present, this creates a two-point boundary value problem (BVP) with the initial and goal states as constraints. If an efficient two-point BVP solver is available that can quickly connect states $\boldsymbol{x}_{near}$ to $\boldsymbol{x}_{rand}$ in the absence of obstacles, this should be used as steering method. Using an efficient BVP, all issues regarding kinodynamic planning are removed for sampling-based planner. However, solving the two-point BVP is often very costly, or a solution may not exist for complex dynamics. Instead an approximate state input pair can be found using optimization methods as will be discussed further discussed in chapter 6.

For nonholonomic systems that are bound by a minimum turning radius, Dubins's paths [18] or Reeds and Shepp's curves [58] can be used to quickly connect states with a minimal distance path. This steering method is often used for wheeled robots as it is optimal under the set constraints and computationally inexpensive.

In absence of a good steering method, a viable approach is to sample the action space to obtain a sequence of inputs. If done multiple times, one can integrate the system forwards for all inputs [19]. The most promising control input is kept, and the path added to the tree. This method is rather slow as it requires multiple forward integration steps per pulled node. Instead, the action space U of the system can be discretized to a set of a priori calculated input sequences called motion primitives. A set that densely covers the time-limited reachable set[2] maximizes the expansion towards sampled nodes, leading to quick exploration of the state space.

5.4.6. Asymptotically Optimal Planning

An asymptotically optimal sampling-based planner named RRT* was introduced in [41]. It features a rewire routine. After sampling a random state, it is connected to multiple nearest neighbors. The path to the new state with the lowest cost is then added to the tree. A rewire step then checks if any of the nearest neighbors can be reached via the new node, with a lower cost path. If so, the nodes are rewired. The RRT* planner will therefore reach the optimal solution if ran for an infinite amount of time.

If a efficient two-point BVP solver or state to state steering method is available, state-of-the-art asymptotically optimal planners such as Informed RRT* [27], Fast marching trees (FMT*) [37] or Batch informed Trees (BIT*) [28] can be used. For more information and a comparison of these planners the reader is referred to [28].

5.5. Decoupled Path Planning

The decoupled planning approach was discussed in the chapter 1. Decoupled trajectory planning is a hierarchical method that can plan under differential constraints by first planning a global path in a lower dimension. Secondly the path is transformed to handle non-holonomic [3] constraints. Thirdly, the path is time parametrized such that it satisfies the differential constraints[57]. The weakness of this scheme is the unidirectional passing down of fixed intermediate solutions between steps. The receiver might not succeed as a result of an unfortunate choice made by the step before. In the example of chapter 1 this resulted in an inefficient path with the possibility of collisions, resulting in an incomplete planner. Hence completeness is lost and optimality is sacrificed. This motivates us to directly take the differential constraints into account during planning.

[2] The time-limited reachable set contains all states that can be reached from the current state within a certain time period while satisfying input constraints.

[3] Non-holonomic constraints are constraints on the state velocities e.g. cars cannot have a velocity perpendicular to the direction of travel of the wheels.

5.6. Path Planner for DASh

The sampling-based kinodynamic approach to path planning mitigates the drawbacks of the decoupled planning approaches found in literature. The resulting paths are guaranteed to be feasible and executable under the set input constraints.

In the literature report [68], the kinodynamic RRT has been chosen to be implemented for two reasons. First, it finds feasible and executable paths. Secondly, in the absence of an efficient BVP solver for the vessel model, asymptotically optimal planners would introduce immense computational loads.

5.7. Conclusion

In this chapter an introduction to path planning approaches and their difficulties has been given. Based on the requirements and properties of planners, the kinodynamic RRT planner was proposed, which will use the vessel model identified in chapter 4. The core routines of the planner were presented and possible improvement on the subroutines were discussed. In the following chapter a framework is presented to implement a computationally cheap distance measure and steering method to minimize the time and path cost of the kinodynamic RRT.

6

A Control-Based Framework for Kinodynamic RRT

In this chapter, a control-based framework is presented that improves upon the standard kinodynamic RRT algorithm. The framework uses a learned approximation of the cost-to-go as a distance metric for the state space of the vessel. The cost-to-go is calculated using forward integration of the vessel under non-linear model predictive control (NMPC). Since the NMPC is too computationally expensive to be used in the kinodynamic RRT algorithm directly, a maneuver automaton is defined to constrain vessel movements to a set of trim states and motion primitives. To test the effectiveness of the proposed distance metrics, the performance of paths generated using different distance metrics are compared.

6.1. The Control-Based Framework

The framework aims to improve kinodynamic sampling-based planners for any observable and controllable agent with linear or non-linear dynamics $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$ with input constraints $\boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max}$. Compared to geometric sampling-based planning, kinodynamic planners run into several extra challenges related to the distance metric and the steering method. Both challenges are addressed by the proposed framework.

The motivation for this framework has two sources. Firstly, the bulk of cost-to-go metrics for kinodynamic planners are based on the linearization of the agent's dynamics, as discussed in section 5.4.4. The cost-to-go approximation becomes inaccurate as the system becomes more non-linear. Secondly, the optimal control method of this framework that is used to formulate the cost-to-go distance metric, can also be used as a time-varying reference tracking controller during path execution. As a result, the cost-to-go objectives during path planning and path execution are the same, underpinning the correct use of the created distance metric.

6.1.1. Cost-To-Go Distance Metric

No natural distance metric is available to select nearest neighbors in the state space, particularly if input constraints are present. A widely accepted notion, is that the ideal distance metric of a dynamic agent is equal to the true cost-to-go between states [47, 48]. In this case the cost is specified by an objective function that captures the desired behavior of the agent. Finding the true cost-to-go requires the solution to a NP-hard Two Point Boundary Value Problem (BVP). Methods to solve BVP's are non trivial and do not exist for most non-linear systems. An ideal cost-to-go metric is not crucial for sampling-based methods to function. However, non optimal nearest neighbors will be selected for expansion, embedding suboptimal edges in the graph, while this could be avoided.

A good metric can improve the results of sampling-based planners as shown in [12]. In kinodynamic planning literature, various methods to retrieve a cost-to-go are employed. In [2] a genetic algorithm was used to approximate a cost-to-go between states. In [43] the lie group structure of the non-linear system was exploited to formulate a metric based on the shortest distance path in the configuration space. The cost-to-go used in this work consists solely on the sequence of control inputs $\tilde{\boldsymbol{u}}$ needed to traverse between states. The cost-to-go V_{ctg} is defined as

$$V_{ctg} = \int_0^{T_{end}} \boldsymbol{u}(t)^T \boldsymbol{R}\boldsymbol{u}(t) \tag{6.1}$$

Where $\boldsymbol{R} = \boldsymbol{I}^{\boldsymbol{3x3}}$ is the constant, positive definite input weighing matrix and T_{end} the time of arrival at the desired state. The control-based cost is based on similar efforts in [62, 78]. In an NMPC objective function, a quadratic penalty on the state error would also be present. This penalty is omitted in the cost-to-go as this error on the difference in state is a bad distance metric for the state space, a conclusion that will be made on the results presented in this chapter. Also note that a trajectory is always captured by a state and input pair $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ where one will uniquely define the other based on the initial states and vessel dynamics.

To quantify the cost-to-go most literature tend to use direct optimal control methods. Li and Todorov determined a locally-optimal cost-to-go based on linearized system dynamics [49]. The approach consists of the iterative improvements of the cost under a Linear Quadratic Regulator(iLQR). The linearization enables quick computations and as a result the cost converges quickly, even in high dimensional systems. The major downside to this method is the inability to account for constraints in the action space. A similar method is employed by Glassman et al [30] that used a cost-to-go of an Affine Quadratic Regulator (AQR). Improved exploration of the state space was demonstrated, but its performance decreased quickly with increasingly non-linear dynamics. The cost-to-go of equation (6.1) will be calculated using non-linear optimal control methods, eliminating the approximation error made by linearization. In more recent efforts in [4, 78] resorted to indirect optimal control to determine the cost-to-go by numerically solving two-point Boundary Value Problems (BVP).

6.2. Optimal Control Methods

In this section optimal control of the vessel model 3.24 is explored to calculate the cost-to-go and double as steering methods. Direct optimal methods are preferred for steering, as they find globally optimal solutions when using multi-start, or locally optimal solutions without multi-start. It is shown that direct optimal control has poor convergence performance for the vessel model. A non-linear model predictive control (NMPC) approach is proposed instead based on good convergence properties.

6.2.1. Direct Optimal Control

The cost-to-go V_{ctg} between initial state x_0 and terminal state x_T can be found by solving the following optimal control problem

$$\begin{aligned} \min_{\boldsymbol{u}(t)\in U,\ T>0} \quad & V_{ctg} + T_{end} \\ \text{subject to} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)). \qquad \forall\, t \in [0, T_{end}] \\ & \boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max} \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ & \boldsymbol{x}(T) = \boldsymbol{x}_T \end{aligned} \tag{6.2}$$

In general, this optimization problem can be solved without discretization using dynamic programming, or indirect optimization methods [16]. Dynamic programming recursively computes an optimal feedback control law for all $\boldsymbol{x}_0$ and t, ultimately resulting in a Hamilton-Jacobi-Bellman (HJB) partial differential equation. Numerical methods to solve the HJB exist but suffer tremendously of the "curse of dimensionality" restricting solutions to low dimensional systems only. Indirect optimization methods rewrite the optimality condition to be contained inside the systems equations of motion, only to arrive at a BVP. The approach is also referred to as "first optimize, then discretize". Numerical solutions to the BVP are difficult to compute due to numerical instability, especially if input constraints are present.

Direct optimal control methods first discretize the input trajectory and obtain a nonlinear programming problem (NLP) of finite dimension. Numerical optimization methods can solve these finite problems and elegantly handle inequality constraints. This approach is often referred to as 'first discretize, then optimize'. The approach has its drawbacks: direct optimizations may not converge and fail to find a solution depending

on the initialization of the NLP. The direct optimization methods are however preferred in real world applications [16].

To test the convergence properties of direct optimal control for the vessel model, a optimization algorithm was implemented for the identified vessel dynamics of equation (5.2) in the Automatic Control and Dynamic Optimization (ACADO) Toolbox, written in C++. ACADO uses a multiple shooting sequential quadratic programming (SQP) algorithm for solving constraint NLPs. A more in-depth discussion on solving NLPs using ACADO can be found in [16].

The convergence of the NLP using SQP for an inital state of $\boldsymbol{x}_0$ and a terminal state $\boldsymbol{x}_T$ is troublesome for the vessel model. Convergence of the NLP using a Mayer term objective T_{end} and integral of the input in the objective function, is only achieved for one in three optimizations. The SQP used to solve the NLP can fail to converge for several reasons. The Hessian of Lagrangian functions are approximated in each SQP iteration, they may be ill-conditioned leading to a lack of convergence. Secondly, the input constraints may be too strong, and no solution exists. Relaxing the NLP by omitting the the input constraints yielded no improvements on the convergence. A more robust optimization approach is needed.

6.2.2. Non-Linear Model Predictive Control

Non-linear model predictive control (NMPC) recursively finds an optimal input that minimizes a cost over a smaller finite prediction horizon. The terminal state constraint $\boldsymbol{x}(T) = \boldsymbol{x}_T$ is omitted and replaced by a penalty on the distance to the goal state in the objective function. For a prediction horizon with N samples of Δt seconds, NMPC solves the following optimization at each time step

$$\begin{aligned} \min_{\tilde{\boldsymbol{u}} \in U} \quad & V(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}) = \sum_{i=1}^{N} (\boldsymbol{x}(i\Delta t) - \boldsymbol{x}_T)^T \boldsymbol{Q} (\boldsymbol{x}(i\Delta t) - \boldsymbol{x}_T) + \sum_{i=1}^{N} u(i\Delta t)^T \boldsymbol{R} u(i\Delta t) \\ \text{subject to} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)). \qquad \forall t \in [0, N\Delta t] \\ & \boldsymbol{x}(0) = \boldsymbol{x}_I \end{aligned} \tag{6.3}$$

where the cost consists of a penalty on the state error and control inputs with positive definite weighing matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, respectively. Depending on the cost function, NMPC is mainly used for reference tracking and has been shown to be a robust control method. During execution, only the first entry of the solution $\tilde{\boldsymbol{u}}(\boldsymbol{0})$ is applied to the vessel. At the next time step, the optimization is performed all over again. The vessel is propagated under NMPC control until it reaches the region $X_T = \{ \boldsymbol{x} \in X_{free} \mid \|\boldsymbol{x} - \boldsymbol{x}_T\|_2 < \epsilon \}$ for a small scalar value ϵ. Based on the hypothesis that using NMPC, a locally optimal trajectory is obtained between $\boldsymbol{x}_0$ and $\boldsymbol{x}_T$, provided the states are close to each other. The cost-to-go of the NMPC trajectory in terms of input is denoted as $V_{ctg}(\boldsymbol{x}_0, \boldsymbol{x}_T)$ and can be found by recursively solving (6.3) and integrate for a predetermined sampling period Δt.

The NMPC optimization was implemented using ACADO. For the vessel dynamics of DASh, propagation under NMPC always converges to the terminal state. Figure 6.1 illustrates the convergence performance of NMPC for a static reference trajectory. Although direct optimal control is favorable as it finds lower cost-to-go paths to $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$, the reliability of the NMPC allows us to create a dense training data set for the cost-to-go in section 6.4. An interesting feature of using the NMPC is that it allows to tune the ratio of the penalty on the state in and input using weighing matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. This ratio determines the desirable behavior for which in turn we find the cost-to-go.

6.3. Discretization of the Action Space

The NMPC itself, is an excellent steering function. It intrinsically handles input and dynamic constraints in a locally optimal manner. It can however not be used directly in the kinodynamic RRT, as its averaged computation time of ±2.10 seconds to connect two states is too long. Instead the action space is discretized to a finite set of N discrete action trajectories denoted by $\mathcal{U}_d = \{U_1(\boldsymbol{x}) \;\; U_2(\boldsymbol{x}) \; ... U_N(\boldsymbol{x})\}$. The action trajectory is defined as discrete piecewise constant series of inputs over a finite time period Δt.

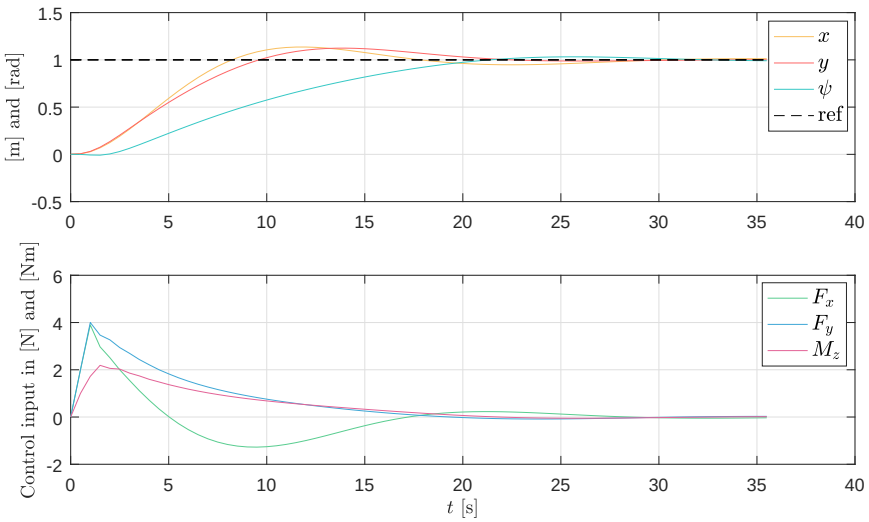

Figure 6.1: Convergence of NMPC towards static reference state.

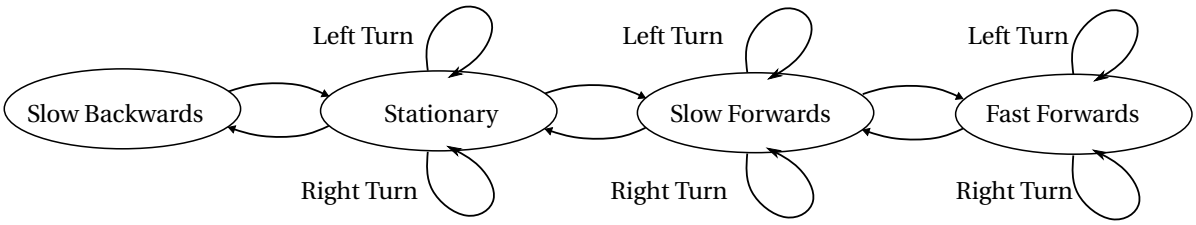

Figure 6.2: an automaton with trim states (ovals) and motion primitives (arrows).

A discrete trajectory $U_d(\boldsymbol{x}) \in \mathcal{U}_d$ is called a motion primitive. The system dynamics using motion primitives can then be modelled using a discrete-time state transition model

$$x_{k+1} = f_d(x_k, U_d) \tag{6.4}$$

where $x_k = x((k-1)\Delta t)$ and the action trajectory $U_d(\boldsymbol{x}) \in \mathcal{U}_d$ applied to the system over the time period from $(k-1)\Delta t$ to $(k)\Delta t$. In theory, a motion primitive must be present for any state $(\boldsymbol{x})$ the vessel is currently in. To limit the set of primitives, a maneuver automaton is used [26].

A maneuver automaton (or automaton) consists of states of constant velocity called *trim* states and motion primitives that steer the agent between the trim states. An elaborate automaton for a vessel that can turn in place, sail backward, sail forward at low and high speeds and perform predetermined turns, is shown in figure 6.2. The current trim state of the vessel determines the available motion primitives. The automaton can be formulated for any autonomous agent that has stabilizable trim states. The automaton used for DASh is shown in figure 6.4. It consists of only one constant forward surge trim state and a set of left and right turn motion primitives. If deemed necessary, the automaton can be extended with more trim states and primitives. However, the shown automaton is detailed enough to solve the planning problems in section 6.5. Note that the discretization of the input space weakens the completeness of a planner, if its is not sampled densely [47]. The influence of the discretization for path planning for DASh will be discussed in section 6.6.

Since we wish to find paths with a low overall cost-to-go, the motion primitive set $\mathcal{U}_d$ of right and left turns is generated using the NMPC controller. As a result, the motion primitives have a minimal cost-to-go when used in the steering method routine. The primitives can be determined a priori, dramatically decreasing the computation time of the local planning routine within kinodynamic RRT. The discretized steering function concatenates one primitive on the state of the nearest vertex, using the predetermined discrete set of motion

primitives $\mathcal{U}_d$, valid at the current trim state. The primitive that minimizes the distance between the end node of the motion primitive and $\boldsymbol{x}_{rand}$ is returned. Pseudo-code of the steering function is given in algorithm 2.

Let the constant surge trim state be $\boldsymbol{x}_{trim} = [\bullet \quad \bullet \quad \bullet \quad u_c \quad 0 \quad 0]^T$, where u_c is the forward cruise speed and $\bullet$ can be any value. The set if minimal cost-to-go turning primitives for $u_c = 0.10\ m/s$ and a desired turn angle ψ_d, is shown in figure 6.3. The primitives are created using a slightly modified model predictive controller that enforces a change in heading, while minimizing the control inputs, sway and deviation of the surge speed with respect to the forward cruise speed. The resulting series of control inputs satisfy the input and differential constraints of the identified vessel model.

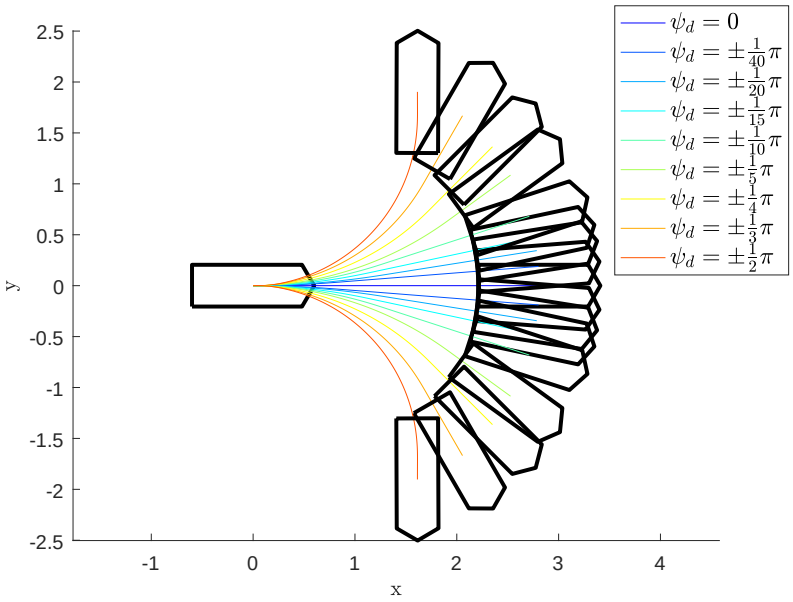

Figure 6.3: Motion primitives valid at trim state $\boldsymbol{x}_{trim} = [\cdot \quad \cdot \quad \cdot \quad u_c \quad 0 \quad 0]^T$.

Algorithm 2 Steering method using motion primitives

Input: Randomly sampled state x_{rand}, nearest neighbor V_{near}, motion primitive set $\mathcal{U}_d$
Output: Trajectory pair $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$, Trajectory end node $\boldsymbol{x}_{new}$

function STEERINGMETHOD(V_{near}, $\boldsymbol{x}_{rand}$, G)
  $Dist \leftarrow 0$
  $minDist \leftarrow$ inf
  for all primitives U_d in $\mathcal{U}_d$ **do**
    $(\tilde{\boldsymbol{x}}', \tilde{\boldsymbol{u}}', x'_{new}) \leftarrow f_d(V_{near}, U_d)$
    $Dist \leftarrow$ GETDIST(x'_{new}, x_{rand})
    if $Dist < minDist$ **then**
      $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, x_{new}) \leftarrow (\tilde{\boldsymbol{x}}', \tilde{\boldsymbol{u}}', x'_{new})$
      $minDist \leftarrow Dist$
    end if
  end for
  return $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, x_{new})$
end function

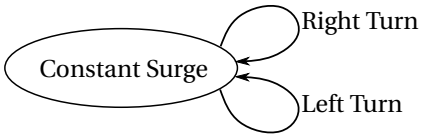

Figure 6.4: The simple maneuver automaton consisting of one trim state and turning motion primitives.

6.4. Fast Cost-To-Go Approximation using Learning Methods

The cost-to-go is expensive to compute. The computational time spend on computing the distance between nodes per RRT iteration, grows linearly with the amount of vertices in G. To use the cost-to-go by calculating a connecting trajectory using NMPC quickly becomes computationally excruciating. Using cost-to-go as a distance measure, its computation time must be decreased significantly.

6.4.1. Learned Cost-To-Go using LWPR

Instead of using a simplified heuristic as a distance metric, machine learning methods can be trained to find approximate values of cost-to-go $\tilde{V}_{ctg}(\boldsymbol{x}_0, \boldsymbol{x}_T)$, as proposed in [6, 78]. Learned models are able to more quickly return learned values by avoiding the online use of NMPC entirely. In this work the supervised learning algorithm Locally Weighted Projection Regression (LWPR) [77] is used for this purpose, as it is well-suited to approximate non-linear functions in high dimensions. For a piecewise constant input sequence $\tilde{\boldsymbol{u}} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_k]$ that connects $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$, the learned algorithm approximates the cost-to-go

$$\hat{V}_{ctg}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}) \approx V_{ctg}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}) = \Delta t \sum_{i=1}^{K} \tilde{u}_i^T \boldsymbol{R} \tilde{u}_i \tag{6.5}$$

Let $\boldsymbol{s}$ denote the input to the LWPR model and $\hat{V}_{ctg}(\boldsymbol{s})$ be the approximated cost. The output is determined by weighing K locally linear models $\Psi_k(\boldsymbol{s})$, called receptive fields, in the form

$$\hat{V}_{ctg}(\boldsymbol{s}) = \frac{1}{W(\boldsymbol{s})} \sum_{k=1}^{K} w_k(\boldsymbol{s}) \Psi_k(\boldsymbol{s}), \quad W(\boldsymbol{s}) = \sum_{k=1}^{k} w_k(\boldsymbol{s}) \tag{6.6}$$

Here $w_k(\boldsymbol{s})$ is a local receptive field that weights the validity of the k^{th} linear model. $w_k(\boldsymbol{s})$ is modelled as a Gaussian

$$w_k(\boldsymbol{s}) = \exp\left(1\frac{1}{2}(\boldsymbol{s} - \boldsymbol{c}_k)^T \boldsymbol{D}_k (\boldsymbol{s} - \boldsymbol{c}_k)\right) \tag{6.7}$$

where $\boldsymbol{c}_k$ is the center of the k^{th} linear model and $\boldsymbol{D}_k$ is its distance metric, setting the size of the locally linear model.

6.4.2. Transformation of Input Data

To simplify the computation of the distance metric between a state $\boldsymbol{x}_I$ and $\boldsymbol{x}_T$, we exploit the structure of the automaton of DASh. The automaton constrains the reachable state spaces of DASh to the defined trim states. In case of DASh, the automaton contains one trim state $\boldsymbol{x}_{trim} = [\cdot \quad \cdot \quad \cdot \quad u_c \quad 0 \quad 0]^T$, which has three free variables: The x and y position and the heading ψ. As a result, any distance computation performed between two trim states, depends on 6 free variables defined in the earth-fixed coordinate state denoted by {n}. To reduce the input dimension of the LWPR to 3 free variables, the terminal state $\boldsymbol{x}_T$ is expressed in the body-fixed coordinate frame of $\boldsymbol{x}_I$, denoted by {b}. The definition of the earth-fixed and body-fixed frame was provided in section 3.1. The transformation is shown in figure 6.5. The transformation does not influence the trajectory between the states and doesn't change the a cost-to-go of a path. The change between states is now captured by the position of $\boldsymbol{x}_T$ expressed in {b} denoted by $\boldsymbol{x}_{b,T} = [x_{b,T} \quad y_{b,T} \quad \psi_{b,T}]$. Details on this transformation are given in appendix C.

6.4.3. Creation of Training Data

The cost-to-go is obtained by controlling the vessel under NMPC control until the state of the vessel enters the small goal region around the goal state. It is not possible to cover the full domain of $x_b \in (-\infty, \infty)$ and $y_b \in (-\infty, \infty)$ as this would require an infinitely large training data set. The terminal states of the training

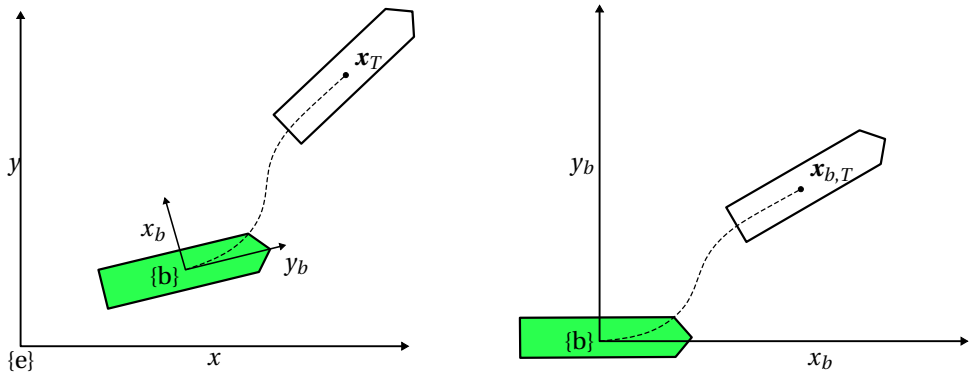

Figure 6.5: Transformation of the terminal state in the earth-fixed {e} to the body-fixed coordinate system {b}. The initial state $\boldsymbol{x}_0$ is displayed in green and terminal state $\boldsymbol{x}_T$ in white.

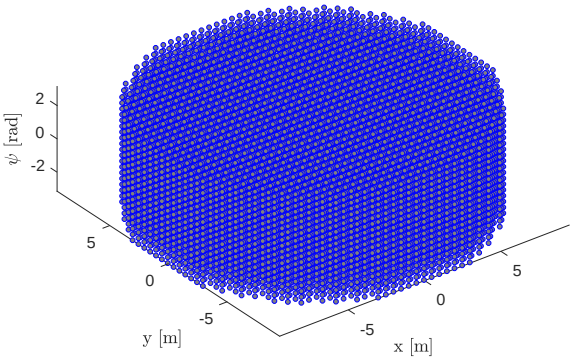

Figure 6.6: The uniform set of terminal states used for training of the LWPR model.

data are restricted to the set $S_{train} = \{ x_b, y_b \mid \sqrt{x_b^2 + y_b^2} < d \}$. The scalar d describes the maximal Euclidean distance from the initial state at the origin. Terminal states in S_{train} are generated on an uniformly spaced grid with a resolution of 0.5 meters for x_b and y_b, and $\frac{2\pi}{18}$ radians for ψ_b, resulting in a set of 12198 data points. Due to portside-starboard symmetry, all samples are mirrored along the x_b-axis while maintaining the same cost. 80% of these data points are selected at random and used as a training set. The remaining 20% of points are used as the test data set. The set of terminal states S_{train} is shown in figure 6.6. All terminal states have been reached using NMPC control. Note that the trained LWPR model will in turn only be valid for inputs $s \in S_{train}$. It is expected that during planning, the cost for points outside S_{train} must be determined. In this case an infinite cost will be returned. The implementation in pseudo-code of the learned cost-to-go in the nearest neighbor search function is shown in algorithm 3.

6.4.4. Training the LWPR Model

The LWPR is trained by normalizing the inputs $\boldsymbol{x}_b$ to their respective ranges. The most important tuning matrix is the positive definite distance metric $\boldsymbol{D}_k$, that sets the size of the receptive fields. Initializing the diagonals of $\boldsymbol{D}_k = c \cdot \boldsymbol{I}$ with too small values of c results in large receptive fields that over generalizes the cost-to-go over a large region. Initializing the diagonals of $\boldsymbol{D}_k$ with too large values of c results in small receptive fields which can lead to allocation of too many receptive fields and over fitting [77]. Both extremes lead to an increase in the prediction error of the LWPR.

The Normalized mean squared error of the predictions for the training data and test data for different initialization of receptor field sizes is shown in figure 6.7. Over fitting starts occurring at initial values for c of

Algorithm 3 Nearest neighbor search

Input: Randomly sampled state x_{rand}, Graph G, trained LWPR model on the domain S_{train}
Output: Nearest Neighbor V_{near}

```
1:  function NEARESTNEIGHBOR(x_rand, G)
2:      cost ← ∞
3:      minCost ← ∞
4:      V_near ← ∅
5:      for all vertices V in G do
6:          if x_rand ∈ S_train then
7:              cost ← GETLWPRCOSTTOGO(V, x_rand)
8:          else
9:              cost ← ∞
10:         end if
11:         if cost < minCost then
12:             minCost ← cost
13:             V_near ← V
14:         end if
15:     end for
16:     return V_near
17: end function
```

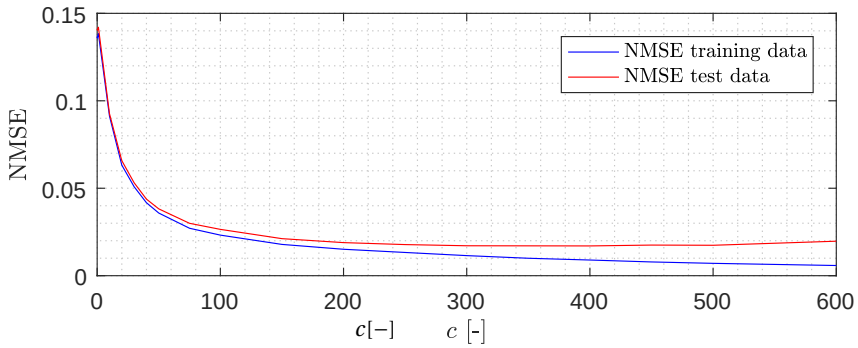

Figure 6.7: The NMSE on the model predictions on the training data set and the test data for various initialization values of $\boldsymbol{D}_k = c \cdot \boldsymbol{I}$.

500. The final model is initialized with $c = 400$, minimizing the NMSE at 0.0170. With the proper initialization of $\boldsymbol{D}_k$, LWPR runs an update rule that adapts the size of receptive fields automatically to further decrease prediction errors.

In figure 6.8 the trend of error between the predictions and true cost values are shown. The prediction error is minimal if all data points are located on the black dotted line. A deviation of this line shows the magnitude of the prediction error. Closer inspection shows good prediction performance for data points with low true cost-to-go. Predictions of high cost-to-go data tend to be less precise and slightly under estimate the true cost-to-go. Inevitably outliers are present when using learning methods. In figure 6.9, a histogram shows the distribution of the squared errors (SE) of predicted costs. The amount of outliers is limited.

$$SE = \left(\hat{V}_{ctg}(\tilde{x}, \tilde{u}) - V_{ctg}(\tilde{x}, \tilde{u})\right)^2 \tag{6.8}$$

Using the learned approximation of the cost-to-go increases the speed of the nearest neighbor search significantly. Using the learned LWPR representation, algorithm 3 returns an approximate cost $\hat{V}_{ctg}(\tilde{x}, \tilde{u})$ within 0.0204 seconds, on average. An speedup factor of approximately 100. However, the model is only able to return a valid cost approximation if the terminal state is within 10 meters of the initial state. This is a direct result of the created training data set that only explores terminal states close to the origin.

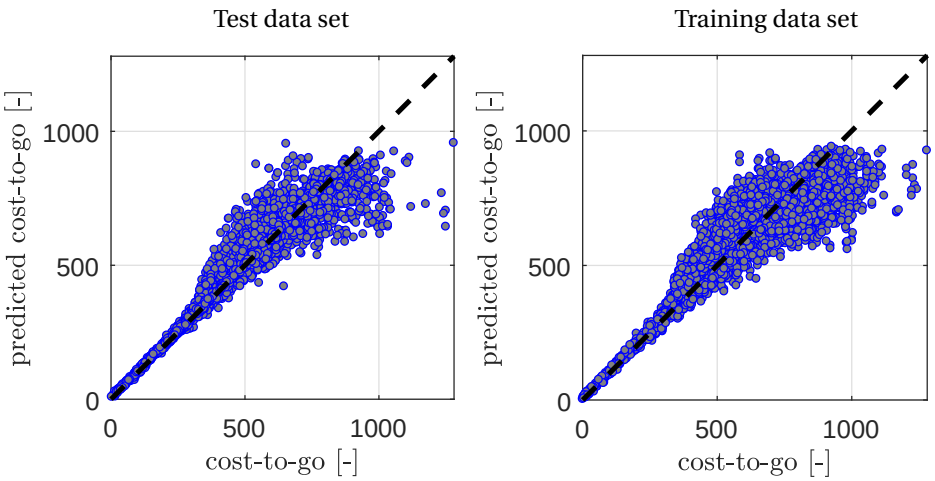

Figure 6.8: Comparison of true and predicted cost-to-go of the training and test data sets.

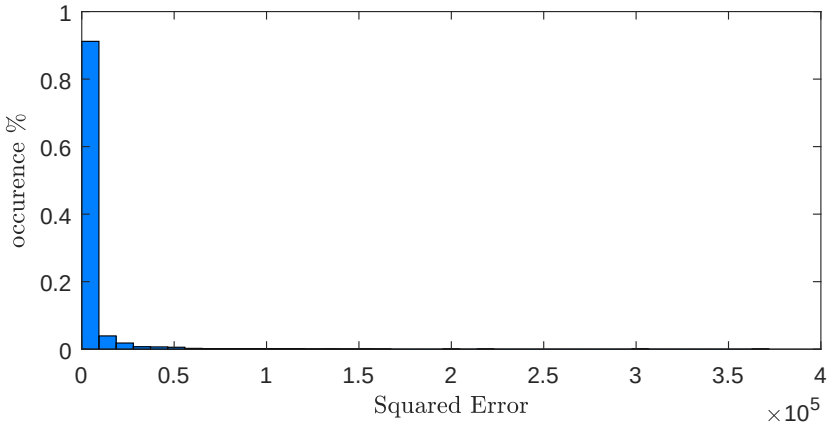

Figure 6.9: Histogram of the squared errors on the test data set.

6.5. Kinodynamic RRT using the Optimal Control-Based Framework

The proposed cost-to-go distance metric influences the cost of paths found using the kinodynamic RRT using the automaton in the steering function. First, nearest neighbors are chosen based on minimal cost-to-go, resulting in the extension of vertices that can make maximum progress towards sampled nodes. Secondly, nearest neighbors are extended towards randomly pulled states using minimum cost-to-go primitives, reducing the overall cost of path within G. As a result, X_{free} is explored with minimal cost-to-go edges. The direct influence on performance is measured by comparing the cost-to-go distance metric to the popular Euclidean (or straight-line) distance metric and a distance metric based on Dubins curves.

6.5.1. Performance Criteria of Paths

The influence of the distance metric in kinodynamic RRT is studied by comparing three performance criteria for three different distance metrics; cost-to-go, Euclidean, minimal path length Dubins curves. The hypothesis of their influence on paths created using kinodynamic RRT is given in table 6.1.
The Euclidean distance metric is defined as

$$V_{\text{Eucl}} = \|\boldsymbol{x}_I - \boldsymbol{x}_G\|_2 \tag{6.9}$$

The Euclidean distance metric computes the straight-line distance between states. Although often a valid

Table 6.1: Hypotheses on the influence of distance metrics in the kinodynamic RRT planning algorithm.

Distance Metric	Properties	Expected influence during planning	Distance metric computation time [s]
cost-to-go	Minimum control inputs required to traverse between states	Reduce overall cost-to-go of paths	0.0204
Euclidean	Minimal straight-line distance between states	Decrease performance	0.0008
Dubins	Minimal length path under turning radius constraint	Reduce the path length	0.0011

metric in geometric planners, the different units (e.g. m/s and rad) of states result in a meaningless, dimensionless distance if applied to the state of a dynamic agent. Planning under the Euclidean distance metric will still result in feasible paths as the planning problems are mostly governed by the in-plane coordinates x and y. Therefore is expected to be a bad distance metric to be used in kinodynamic planning. On the upside, the average computation time between two states of the Euclidean distance is only 0.0008 seconds.

A Dubins curve is the shortest curve that connects two states on a Euclidean plane denoted by $\boldsymbol{x} = \begin{bmatrix} x & y & \psi \end{bmatrix}^T$, given a minimum turning radius of the agent. Let S be a straight path and R and L a right and left turn with maximum curvature, respectively. It is proven that the set $\{S, LRL, RLR, LSL, LSR, RSL, RSR\}$ of concatenated turns and straight lines can connect two states on the Euclidean plane with a minimum distance curve [18]. Three examples of minimal distance Dubins curves are given in figure 6.10.

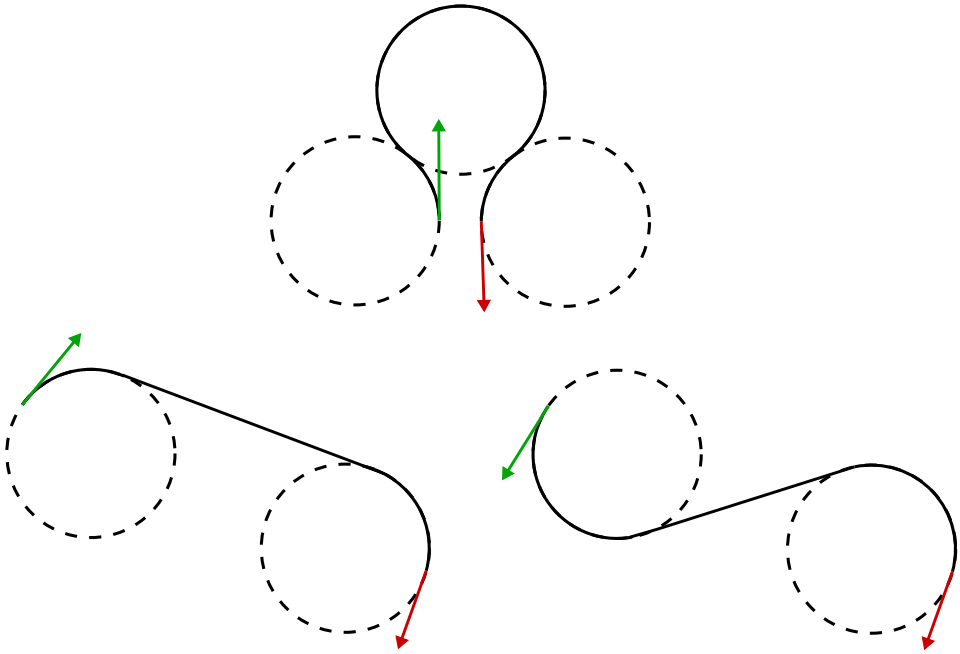

Figure 6.10: Dubins curves of the combination LRL (top), RSR (left) and LSR (right). Initial states are in green, terminal states in red. The dotted circles indicate the minimal turning radius.

Typically the Dubins curves are used in non-holonomic systems such as cars with a well defined minimal turning radius. Although the vessel model of equation (3.23)-(3.24) is a holonomic system, the motion primitives in figure 6.3 bear a resemblance to the Dubins curves as turns have minimal sway under constant surge speed. The minimal turning radius found among the motion primitives is ± 1.8 meters. Using a subspace $\begin{bmatrix} x & y & \psi \end{bmatrix}^T$ of the state space, the length of a Dubins curve is easily computable and returned within 0.0011 seconds. It can be used as a distance measure in order to decrease the length of path found using the kinodynamic RRT.

6.5.2. Implementation Details

The nearest neighbor function, the steering function and the kinodynamic RRT of algorithm were implemented in matlab, as an extension to the geometric RRT and graph functions in the *Robotics Toolbox* of Peter Corke [13]. The kinodynamic RRT creates a graph of the free state space until a vertex reaches the goal region or if a predetermined number of 500 states are sampled.

A goal sampling bias of 5% was used. The goal region is defined in the x-y plane and has a radius of 2 meter to avoid problems with resolution completeness near the goal state. To determine the performance of the various distance measures, additional improvements on state sampling, steering or collision checking as surveyed in [19], were not implemented. Although such modifications could benefit the planners performance, it would cloud the influence of the distance metric which would in turn hinder their comparison.

6.6. Planning Results

The performance measures of the kinodynamic RRT were determined by testing the algorithm on three different maps; the Harbor map, the Islands map and the RDM map. Due to the randomness in the state sampling routine, paths are created in a non-deterministic way. This means that every run of the kinodynamic RRT returns a different path. To determine the influence of the distance metrics, the planning problem was initialized twenty times with a different random state sampling sequence.

The planning success rate is the percentage of initializations that succeeded to find a feasible path. The overall cost-to-go is the integral of input as defined in equation (6.1) along the complete path. The path length on the x,y-plane is noted in other to confirm or reject the hypothesis that the Dubins metric results in shorter paths. The average runtime per path can be used to compare the on-line implementability of the planners. Finally, the amount of sampled nodes needed to find a feasible path are provided to determine the expansiveness of graph G for the various distance metrics. Note that the performance measures are an average of feasible paths. Paths that did not reach the goal region were excluded in the calculation of the averaged performance measures.

In the following pages, a visual comparison of the of the kinodynamic RRT graph under the learned cost-to-go, Euclidean and Dubins metric is shown per map, for an identical sampling scheme of 500 states. The resulting graphs and their progress towards the goal state are shown in figure 6.11, 6.12 and 6.13. The static obstacles are shown in gray, the goal region in green and moving obstacles in red with their respective oscillating paths in black. The RDM map is the only map that contains moving obstacles.

The remained of this page is intentionally left blank. Planning results per map are displayed on the next pages.

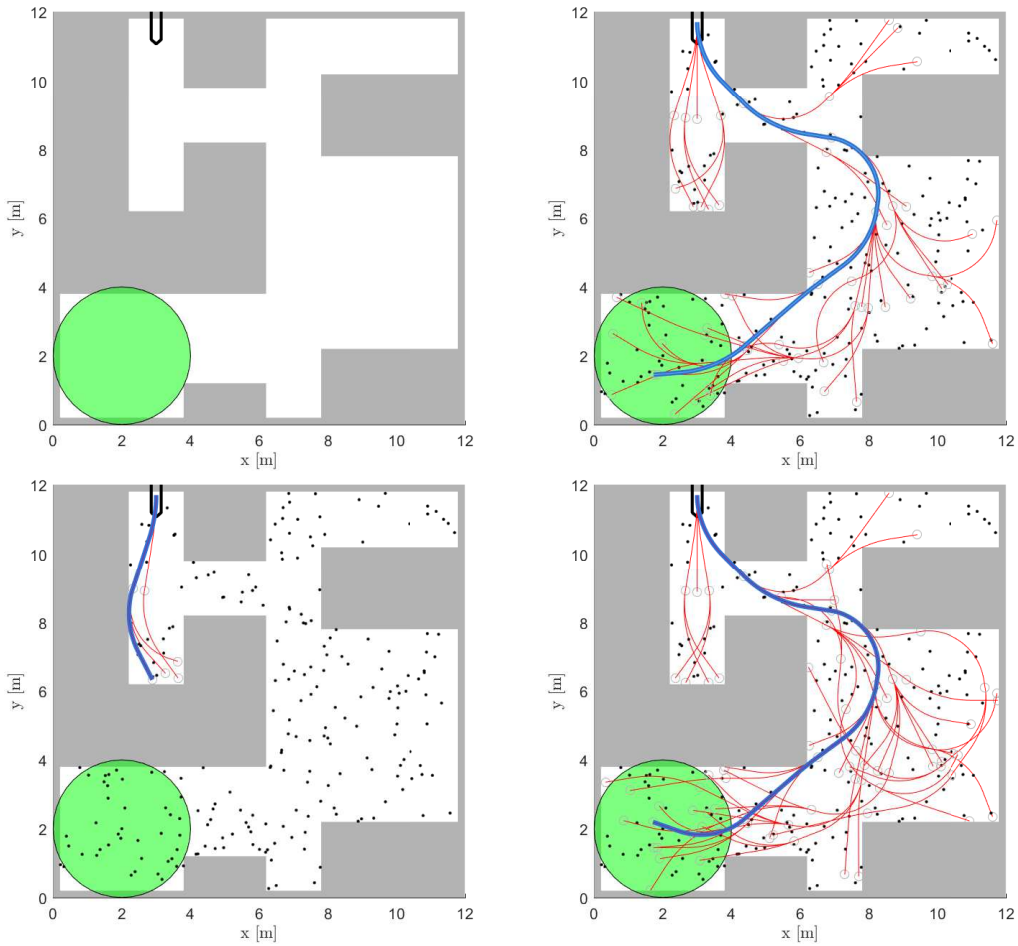

Figure 6.11: (top left) The harbor map. The graph G in red after sampling 500 states using the cost-to-go (top right), the Euclidean (bottom left) and the Dubins distance metric (bottom right). Static obstacles are shown in gray and the goal region in green. The path in G with maximal progress towards the goal state is highlighted in blue.

Table 6.2: Average performance of 20 paths for the Harbor map

Map: Harbor

Distance Metric	Succes Rate %	Overall cost-to-go	Path Length $[m]$	Runtime $[s]$	Sampled States
Cost-to-go	40%	209.7	17.9	28.16	71
Euclidean	0%	-	-	-	-
Dubins	70 %	217.9	17.3	1.32	80

The Harbor map contains small and narrow passages that have to be navigated in order to reach the goal. As the vessel dynamic locally constrains the state space it is important to used a good distance metric, to avoid incompleteness. Inspection of graphs G in figure 6.11 shows the superior exploration using the cost-to-go and Dubins metric, compared to the Euclidean metric. The Euclidean metric is unable to explore the free space on the map as it gets stuck at the first corner on the map. This can be the result of two factors: Too little samples (500) have been pulled, but actually the planner is probabilistically complete, or the discretization of the action space into motion primitives is too sparse and the planner is not resolution complete.

The averaged path performance measures are displayed in table 6.2. The cost-to-go and Dubins metric have similar performance in terms of the overall cost-to-go, path length and the amount of sampled nodes before finding a feasible path. A higher success rate of of 70% is observed for the Dubins metric while also sporting a significantly shorter computation time.

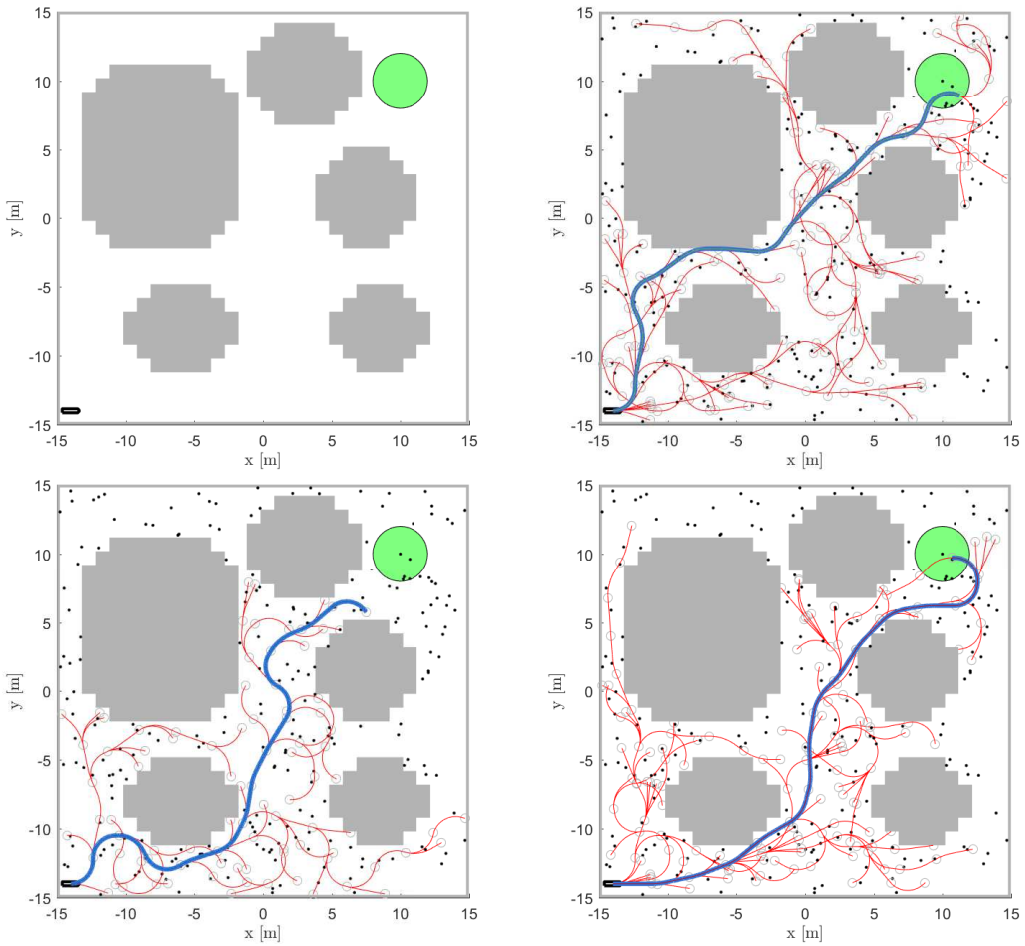

Figure 6.12: (top left) The Islands map. The graph G in red after sampling 500 states using the cost-to-go (top right), the Euclidean (bottom left) and the Dubins distance metric (bottom right). Static obstacles are shown in gray and the goal region in green. The path in G with maximal progress towards the goal state is highlighted in blue.

Table 6.3: Average performance of 20 paths for the islands map

Map: Islands

Distance Metric	Succes Rate %	Overall cost-to-go	Path Length $[m]$	Runtime $[s]$	Sampled States
cost-to-go	100%	452.0	42.1	32.5	135
Euclidean	90%	645.9	46.5	3.6	210
Dubins	100%	439.0	42.3	2.8	136

The Islands map of figure 6.12 is a spacious, open map. A visual comparison of graphs G shows good exploration of the free space using the cost-to-go and Dubins metric. The Euclidean metric shows improved exploration compared to the Harbor map, but still fails to reach the goal region after sampling 500 states.

The averaged path performance is displayed in table 6.3. The success rate has significantly improved, due to the openness of the map. The Euclidean metric results in an high overall cost-to-go paths, a factor 1.5 higher compared to the cost-to-go and Dubins distance metric. On average, more state samples samples are needed to find a feasible path, confirming its decreased expansiveness. Again, the performance of the cost-to-go and Dubins metric is very similar. The overall cost-to-go is marginally lower using the Dubins metric, while the shortest path length is obtained by using the cost-to-go metric. The average runtime of the Dubins metric low at 2.8 seconds, compared to the impractically long runtime of the cost-to-go metric at 32.5 seconds.

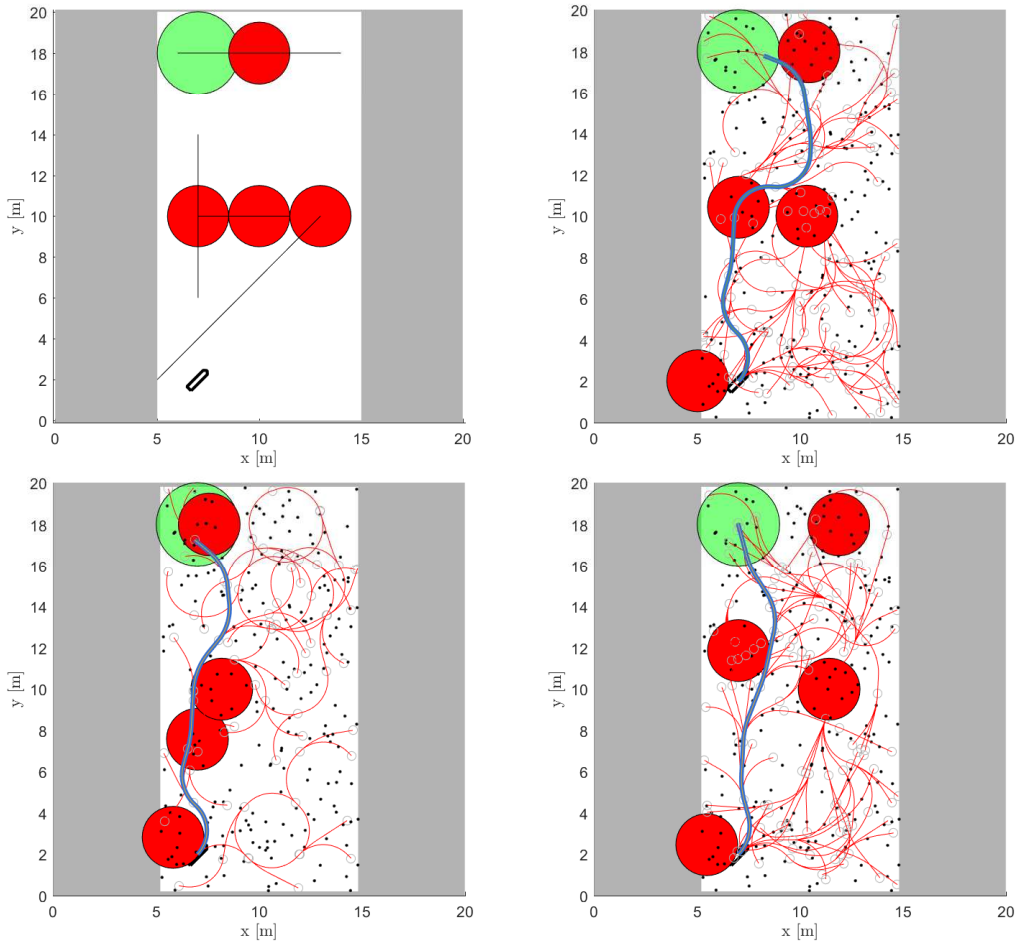

Figure 6.13: (top left) The Islands map. The graph G in red after sampling 500 states using the cost-to-go (top right), the Euclidean (bottom left) and the Dubins distance metric (bottom right). Static obstacles are shown in gray, the goal region in green. Moving obstacles in red move along the straight-line paths depicted in black. The path in G with maximal progress towards the goal state is highlighted in blue.

Table 6.4: Average performance of 20 paths for the RDM map

Map: RDM					
Distance Metric	Succes Rate %	Overall cost-to-go	Path Length $[m]$	Runtime $[s]$	Sampled States
cost-to-go	100%	223.3	19.0	15.2	42
Euclidean	100%	284.6	20.1	1.6	51
Dubins	100%	222.4	18.5	1.4	38

The RDM map is of the same size as the real water basin. Four moving obstacles are present which move back and forth along the black lines shown in the top left image of figure 6.13.

The small size and openness of the maps resulted in a 100% success rate of all planners. Visual inspection show the increased expansiveness of the graphs created using the cost-to-go and Dubins metric. The Euclidean metric under-performs resulting in the longest paths with a highest overall cost-to-go. Paths created using the cost-to-go and Dubins metric perform similarly, resulting in low overall cost-to-go paths of similar lengths, requiring about 40 sampled states before finding a feasible path. Again, the Dubins metric has a significantly lower computation time compared to the cost-to-go metric.

To gain futher insight into the runtime, a breakdown of the computation times per RRT routine is given in

Table 6.5: Computation time breakdown for the "Islands" map using the learned cost-to-go metric.

Sampled States	Runtime [s]	% in Nearest Neighbors	% in Steering method	% in Collision Checking	% in Other
50	5.9	68	13	4	15
100	24.3	85	6	2	7
150	52.6	90	4	1	5
200	94.5	93	3	1	3
250	147.5	94	2	1	3

Table 6.6: Computation time breakdown for the "Island" map using the Dubins curve distance metric.

Sampled States	Runtime [s]	% in Nearest Neighbors	% in Steering method	% in Collision Checking	% in Other
50	1.1	10	65	20	5
100	2.2	13	67	19	1
150	3.2	17	65	18	0
200	4.5	20	63	17	0
250	5.8	25	60	15	0

table 6.5 and 6.6 when using the cost-to-go metric and the Dubins metric, respectively. The percentage of computation times is given per RRT routine: Nearest neighbors search, steering method and collision checking. The time spent in other routines included state sampling, graph operations, communication between routines and goal checking is grouped into the "Other" category. All percentages are rounded to integers.

Using the learned cost-to-go metric, the majority of the time is spend in the nearest neighbor search. For increasingly large graphs the nearest neighbor search will compare the distance between more and more vertices, increasing the computation time. The cost-to-go metric seams to slow for planning, but note that a single distance computation using the LWPR model takes 0.0204 seconds on average, while the NMPC takes 2.10 seconds. This is a speedup of a factor 100 or two orders of magnitude. The learning step is the critically enabling feature to find paths in a reasonable time. As an indication, the amount of distance computations quickly increases with the amount of vertices in graph G, as shown in figure 6.14. Any graph that requires more then a hundred vertices to find a feasible path (e.g. the Islands map) would require $\pm 1e4$ metric computations resulting in a runtime of several hours.

Using the Dubins metric during planning, most time is spend in the steering method. The length of a minimal distance Dubins curve can be quickly computable, resulting in very fast planning.

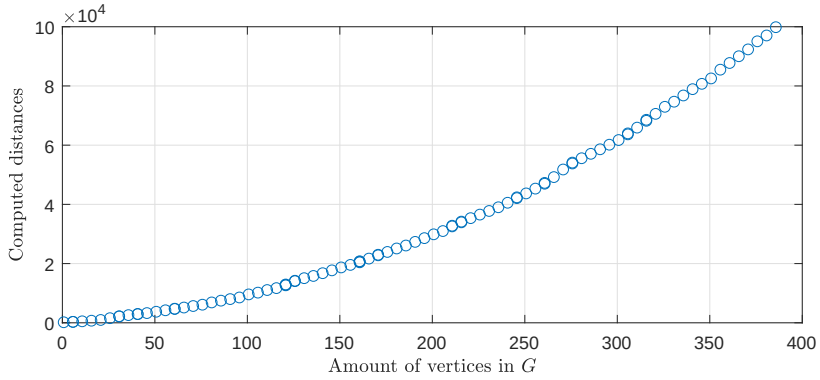

Figure 6.14: The amount of computed distances with increasing size of graph G for the "Islands" map using the learned cost-to-go metric.

6.7. Conclusion

A framework was proposed to implement a cost-to-go distance metric and a maneuver automaton into the kinodynamic RRT planner, in order to decrease the computation times and overall cost-to-go of paths. The proposed automaton of DASh constrained the obtainable velocities to a single trim state. Turning maneuvers were calculated a priori using NMPC to find minimal cost-to-go motion primitives. The simplified steering method concatenated motion primitives on the state of the nearest neighboring vertex, resulting in a computationally inexpensive expansion of G.

The computationally expensive cost-to-go distance metric based on the NMPC cost, was approximated using the LWPR learning method algorithm. By generating a training data set and performing learing a priori to planning, the learned approximation of the cost-to-go can be calculated a factor of 100 times faster. The resulting kinodynamic RRT planner is able to find paths for the presented planning problems in about half a minute.

The hypothesis of the influence of a distance metric on path found using kinodynamic RRT was tested by comparing three distance metrics, each with a different objective. The cost-to-go tries to minimize the cost-to-go between states, which should minimize the overall cost-to-go of the complete path. The Dubins curves distance metric minimizes the path length between states under minimal turning radius constraints, which is expected to result in shorter paths. The Euclidean metric minimizes the straight-line distance between states, which is a bad measure of distance in the state space. Therefore it is expected that path found using this metric are longer and less efficient compared to the cost-to-go and Dubins metric.

The Euclidean metric was shown to be unsuited for use in kinodynamic RRT, under performing compared to the other metrics. The hypothesis that the Euclidean metric has a negative effect on the performance of the kinodynamic RRT is therefore confirmed.

The results indicate that the framework allows for relatively fast path planning using the cost-to-go metric, in which paths were found within approximately half a minute. However, the hypothesis of the influence of the cost-to-go metric has on found paths, does not hold. The Dubins distance metric is equal to the cost-to-go metric in terms of overall cost-to-go, path length and expansiveness. Based on the results in section 6.6, we can state that the hypothesis can neither be confirmed or rejected due to the lack of a performance baseline. For implementation purposes, the Dubins metric is preferred based on it low computational cost, returning paths in a fraction of the time compared to using the cost-to-go metric.

One possible explanation for the similar planning performance of the Dubins and cost-to-go metric is the close relation between the path length and cost-to-go for our vessel. The vessel is only locally constrained by its dynamics. To determine the quality of the optimal control-based framework using the cost-to-go metric, it should be applied to an agent with stronger dynamical constraints e.g. an under actuated vessel or oil tanker with stricter input constraints and larger system inertia terms. The cost-to-go for these systems should capture distance in the state space more accurately compared to a geometric approximation, such as the Dubins metric.

Overall, the planning performances measures in tables 6.2, 6.3 and 6.4 have shown that the performance of kinodynamic RRT vary considerably, depending on the used distance measure. The Dubins metric and cost-to-go metric of the optimal control-based framework compare favorably to the widely used Euclidean metric. However, the Dubins metric requires significantly less runtime to return feasible paths.

7

Verification of the Vessel Model and Control

The guidance and control algorithms presented in earlier chapters are verified using a 1:25 ASD 3111 scale model called "Damen Autonomous Ship", *aka* DASh, at the water basin of the Rotterdamse Droogdok Maatschappij (RDM) in Rotterdam. First, it is verified that the identified vessel model can be used in a non-linear model predictive controller (NMPC) to stabilize DASh at a static reference position. The performance of NMPC in static reference tracking is compared to the more traditional proportional derivative (PD) controller. Secondly, it is verified that time-varying references created with the kinodynamic RRT planner can be executed using NMPC. The real-time capabilities of the kinodynamic RRT planner are not tested, as the planner is not directly embedded into the DASh board computer.

7.1. Real-World Disturbances

No noticeable wind, wave or other environmental disturbances were present at the indoor RDM water basin. However, it is certain that several unknown and unmodeled forces act on DASh during testing. First and foremost is the effect of waves generated by the vessel itself. The concrete basin walls reflect the waves, causing wave disturbances throughout the basin. Furthermore it was observed that DASh drifted around the basin when its actuators were stationary. It is expected this was caused by low velocity water currents and air draft. Unmodeled dynamics will also influence the movement of DASh which was proven using the turning maneuver in section 4.6.1. Results showed that long term forward model predictions diverged from the observed output under the same control inputs.

In combination with the unknown environmental forces, it is unrealistic to assume that open-loop control can be used to execute the planned paths without tracking error. A feedback control loop must be implemented to minimize any deviations from the planned path. Using the identified vessel model, NMPC is used as a reference tracking controller. The unmeasured states are estimated using the Extended Kalman Filter (EKF), which was presented in section 3.10.

7.2. Sensor Measurements

As discussed in chapter 2, DASh is outfitted with a Pozyx indoor positioning sensor. The basin was equipped with four stationary ultra wide-band (UWB) anchors that communicate with a Pozyx UWB receiver installed on DASh. The Pozyx provides the measurement $\boldsymbol{y}_{meas} = [x_{receiver} \quad y_{receiver}]$, at every sampling period. The $x_{receiver}$ and $y_{receiver}$ positions are determined by triangulation of distances to the anchors, based on the communiction latencies of each anchor. The receiver is placed midships, 0.40 meter in front to the *COG*. The heading ψ is measured using a BNO055 digital compass. The BNO055 runs an embedded sensor fusion algorithm that corrects the heading based on and inertial measurement unit (IMU) and a gyrocompass. The location of the *COG* and system output is determined as

$$\boldsymbol{y}_{meas} = \begin{bmatrix} x_{receiver} + 0.4\cos(\psi) \\ y_{receiver} - 0.4\sin(\psi) \\ \psi \end{bmatrix} \tag{7.1}$$

To determine the measurement noise, a measurement test series at static positions throughout the basin was performed. Results are displayed in figure 7.1. Large noise on the state measurements is observed with measurement errors of up to 5 meters(!). These measurements are regarded as outliers and are mainly present in the upper part of basin for $y \geq 8$.

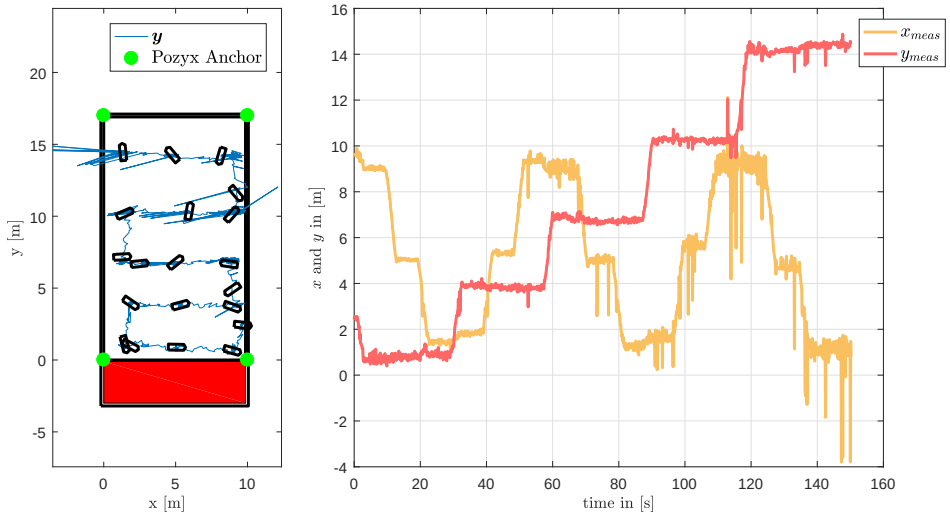

Figure 7.1: Measurements at static positions through out the basin. Large measurement noise and outliers are present, especially in the upper part of the tank. Pozyx anchor positions are given in green.

The source of the outliers is unknown. It is assumed that one of the anchors at the lower part of the basin periodically loses its connection to the receiver located on DASh. Triangulation is then performed using three anchors instead of four, leading to a vastly different position. The positional change is of such magnitude it cannot be regarded at measurement noise and therefore an outlier filter is implemented. The filter rejects any measurements that fall outside the tank or a distance greater then 1 meter from the moving average of the last 10 measurements. Let $\text{dist}(\boldsymbol{y}(i), \boldsymbol{y}(j))$ be the Euclidean distance between position $\boldsymbol{y}(i)$ and $\boldsymbol{y}(j)$, the outlier filter first classifies the latest measurement

- if $\boldsymbol{y}_{meas}(t)$ lays outside the tank, then $\boldsymbol{y}_{meas}(t)$ is an outlier.
- if $\text{dist}\left(\boldsymbol{y}_{meas}(t),\ \text{mean}([\boldsymbol{y}_{meas}(t-\Delta t), ..., \boldsymbol{y}_{meas}(t-10\Delta t)]\right) \geq 1$ is true, $\boldsymbol{y}_{meas}(t)$ is an outlier.

Based on the classification, the following actions are performed:

- if $\boldsymbol{y}_{meas}(t)$ is *not* an outlier, $\boldsymbol{y}(t) = \boldsymbol{y}_{meas}(t)$.
- if $\boldsymbol{y}_{meas}(t)$ is an outlier, $\boldsymbol{y}(t) = \boldsymbol{y}(t-\Delta t)$.
- if the last 10 measurements are classified as outliers, assume the next received measurement is *not* an outlier such that $\boldsymbol{y}(t) = \boldsymbol{y}_{meas}(t)$.

The last step ensures the outlier filter is able to recover in case of a sequence of wrongfully classified outliers. Using this filter the amount of outliers is significantly decreased, resulting in a more stable EKF and less jitter in the control action.

7.3. Feedback Control Loop

The NMPC, EKF, outlier filter and the thrust allocation algorithm were embedded on the BeagleBone Black Revision C board computer. Due to the real-time requirement for feedback control, all algorithms were coded in the efficient C++ to minimize computation times. The kinodynamic RRT path planner was not directly embedded, precomputed trajectory pairs $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ were stored on the board computer that acted as time-varying references. At the start of every test the digital compass and inertial measurement unit (IMU) were calibrated. The control loop displayed in figure 7.2 runs at 10 Hz, the highest frequency at which the position measurements can be obtained. The system dynamics are relatively slow and can easily be captured at this frequency.

At system startup, the initial state of the vessel is determined by taking the mean of 10 position measurements. The time-varying reference $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ generated by the planner is passed to the NMPC. At every sampling period, the following steps are performed: Upon receiving a measurement $\boldsymbol{y}_{meas}$ from the sensor drivers, the EKF estimates the vessel state $\boldsymbol{x}$ and feeds it to the NMPC. The NMPC computes the optimal control inputs over the control horizon. Only the first input $\boldsymbol{u}$ of this sequence is sent the the thrust allocation algorithm, which find the optimal azimuth angles α_i and propeller rotation speed n_i for each azimuth thruster, while accounting for the non-linear actuator constraints. The actuator drivers execute the given commands.

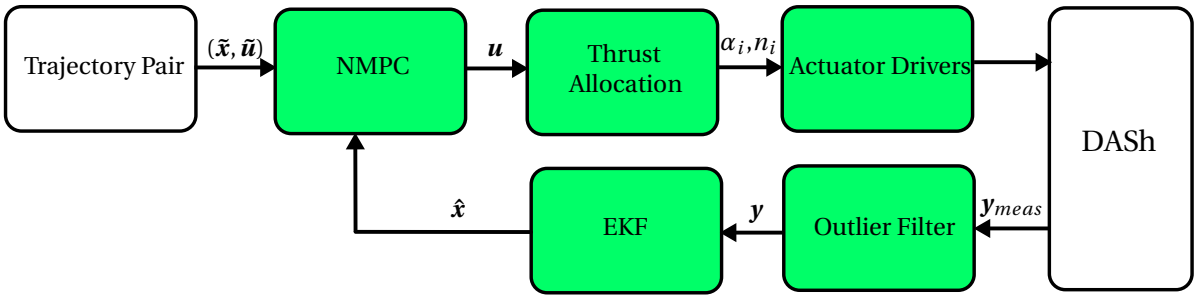

Figure 7.2: The control loop of DASh. Algorithms in green boxes are implemented on-board and run in real time.

7.3.1. Non-Linear Model Predictive Control

The precomputed discrete trajectory pairs $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ are treated as a time dependent state reference $\tilde{\boldsymbol{x}}(t)$ and input reference $\tilde{\boldsymbol{u}}(t)$. The NMPC performs the following optimization at each time step

$$\min_{\Delta \boldsymbol{u}} \quad \sum_{i=1}^{N} (\hat{\boldsymbol{x}}(i\Delta t) - \tilde{\boldsymbol{x}}(i\Delta t))^T \boldsymbol{Q} (\hat{\boldsymbol{x}}(i\Delta t) - \tilde{\boldsymbol{x}}(i\Delta t)) + \sum_{i=1}^{N} (\boldsymbol{u}(i\Delta t) - \tilde{\boldsymbol{u}}(i\Delta t))^T \boldsymbol{R_1} (\boldsymbol{u}(i\Delta t) - \tilde{\boldsymbol{u}}(i\Delta t)) + \ldots$$
$$\ldots \sum_{i=1}^{N-1} \Delta \boldsymbol{u}(i\Delta t)^T \boldsymbol{R_2} \Delta \boldsymbol{u}(i\Delta t) \tag{7.2}$$

$$\text{subject to} \quad \dot{\hat{\boldsymbol{x}}}(t) = f(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) \qquad \forall\, t \in [0, N\Delta t] \tag{7.3}$$
$$\boldsymbol{u}(k\Delta t) = \boldsymbol{u}((k-1)\Delta t) + \Delta \boldsymbol{u}(k\Delta t) \qquad k = 1, \ldots, N \tag{7.4}$$
$$\hat{\boldsymbol{x}}(0) = \boldsymbol{x}_I \tag{7.5}$$

with the estimated state $\hat{\boldsymbol{x}}$, prediction horizon N and positive definite diagonal weighing matrices $\boldsymbol{Q}$, $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$. The vessel model is rewritten in the incremental-input-output (IIO) form, by addition of equation (7.4). This allows large input changes $\Delta \boldsymbol{u}$ to be penalized with weighing matrix $\boldsymbol{R_2}$, reducing nervous actuator behavior. Normally, the IIO formulation would omit the penalty term on the input offset $(\boldsymbol{u}(i\Delta t) - \tilde{\boldsymbol{u}}(i\Delta t))$, resulting in a controller with a pure integrator which is able to handle drift terms in the noise [75]. This behavior is especially desirable for steady state tracking, as the cost function is able to go to zero.

In this implementation the penalty on the input offset $(\boldsymbol{u}(i\Delta t) - \tilde{\boldsymbol{u}}(i\Delta t))$ is maintained as it helps the thrust allocation algorithm to correctly predict future azimuth angles along the trajectory. As a result, large azimuth rotations during execution are reduced when the cost on the state penalty $(\boldsymbol{Q})$ is small. The thrust allocation

algorithm is able to better track the proposed inputs resulting in improved tracking of the path overall. Note that to avoid large objective costs when the cost on the state penalty is small, we choose the weights of $\boldsymbol{R_1}$ low compared to $\boldsymbol{R_2}$.

The trajectory pair $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ is a unique pair of discrete sequences, in which the one defines the other. It can be argued that the penalty on the input offset $(\boldsymbol{u}(i\Delta t) - \tilde{\boldsymbol{u}}(i\Delta t))$ should be removed, as the state reference $\tilde{\boldsymbol{x}}$ contains all information present in $\tilde{\boldsymbol{u}}$ via the dynamic relation $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, which in our case might, and probably will, deviate from the real system dynamics. Nevertheless, implementation of the NMPC without the penalty on the input offset introduced large delays of the input, due to (often) unnecessary large thruster rotations. As the thrust allocation algorithm only allocates for the next time step, it was not able to cope with the large variations in the inputs.

7.4. Static Reference Tracking Results

The NMPC for static reference tracking is tuned as follows

$$\begin{aligned} \boldsymbol{Q} &= \text{diag}(\begin{bmatrix} 10 & 10 & 10 & 1 & 1 & 1 \end{bmatrix}) \\ \boldsymbol{R_1} &= \boldsymbol{I^{3\times 3}} \\ \boldsymbol{R_2} &= 2.5\boldsymbol{I^{3\times 3}} \\ N &= 20 \\ \Delta t &= 0.1 \end{aligned} \tag{7.6}$$

The convergence of the NMPC for a static reference tracking test with initial state $\boldsymbol{x}_I = \begin{bmatrix} 1 & 9 & \frac{1}{2}\pi & 0 & 0 & 0 \end{bmatrix}$, state reference $\tilde{\boldsymbol{x}} = \begin{bmatrix} 3 & 9 & \frac{1}{2}\pi & 0 & 0 & 0 \end{bmatrix}$ and a zero input reference $\tilde{\boldsymbol{u}} = \boldsymbol{0^{3\times 1}}$ is shown in figure 7.3. DASh successfully converges to the static reference with a settling time of ±48 seconds for the x and y states. The controller prioritized to rotate the vessel and sail backwards towards the reference position. Once arrived there is corrected it's heading, overshooting the reference by 36 degrees. Significant measurement noise on the y position causes large oscillation in the control inputs. Regardless, the static reference position is kept without steady-state offset. Increasing the length of the prediction horizon N significantly impacts the performance of the NMPC resulting in high control inputs and overshoot. This is caused by the decreasing quality of long term predictions using the identified model, as discussed in section 4.6.1. The system becomes unstable for $N \leq 50$.

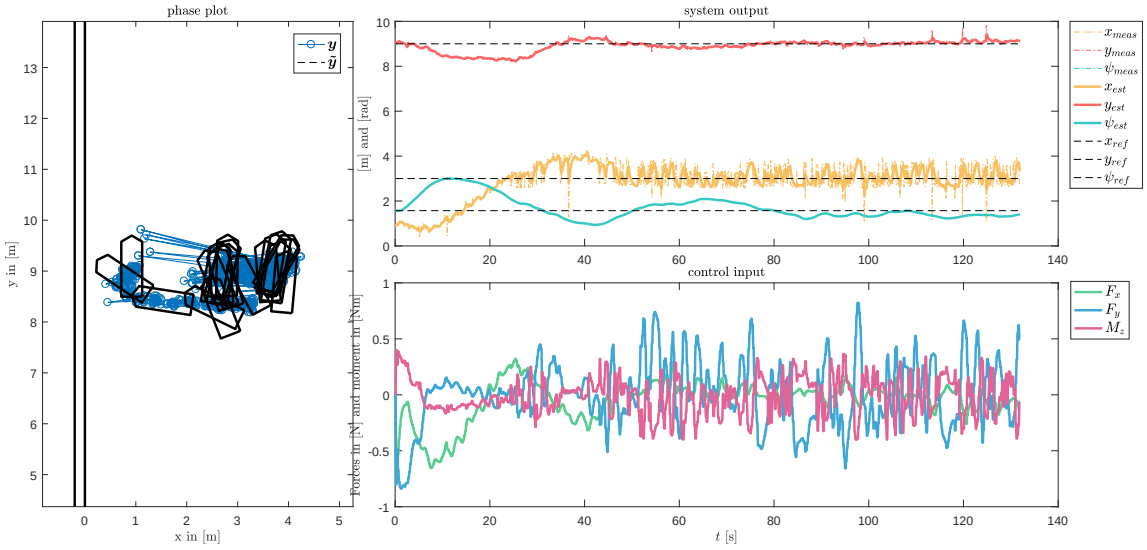

Figure 7.3: States and control inputs for the static reference tracking test under NMPC with initial state $\boldsymbol{x}_I = [\,1 \quad 9 \quad \frac{1}{2}\pi \quad 0 \quad 0 \quad 0\,]$, state reference $\tilde{\boldsymbol{x}} = [\,3 \quad 9 \quad \frac{1}{2}\pi \quad 0 \quad 0 \quad 0\,]$ and a zero input reference $\tilde{\boldsymbol{u}} = \boldsymbol{0}^{3\times 1}$.

To compare the performance of the NMPC, the same static reference tracking test is performed using a non-linear PD controller. The control law and a proof of its stability is described in appendix E. It is shortly repeated here

$$\boldsymbol{u} = -\boldsymbol{H_m}\dot{\boldsymbol{\nu}} + \boldsymbol{R}^T(\psi)(-\boldsymbol{K_p}\tilde{\boldsymbol{\eta}} - \boldsymbol{K_d}\dot{\boldsymbol{\eta}}) \tag{7.7}$$

where $\boldsymbol{x} = [\boldsymbol{\eta}^T \quad \boldsymbol{\nu}^T]^T$ and $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta} - \boldsymbol{\eta}_{reference}$ is the reference tracking error. $\boldsymbol{K_p}$ is the proportional feedback term, $\boldsymbol{K_d}$ the damping term and $\boldsymbol{H_m}$ is a damping term on accelerations. The following tuning of the parameters are used

$$\boldsymbol{H_m} = \boldsymbol{0^{3\times 3}}, \quad \boldsymbol{K_p} = \text{diag}(\begin{bmatrix} 0.7 & 0.7 & 0.7 \end{bmatrix}), \quad \boldsymbol{K_d} = \text{diag}(\begin{bmatrix} 1.8 & 1.8 & 4 \end{bmatrix}) \tag{7.8}$$

The tuning matrix $\boldsymbol{H_m}$ is set to zero to avoid nervous control actions as a result of large measurement noise. Test results using the non-linear PD control for static reference tracking are are shown in figure 7.4. The PD controller successfully steers DASh to the static reference with a settling time of ±63 seconds. No overshoot is present on x, y or ψ. The behavior of the planner is different from the NMPC as the vessel sails towards the static reference with a pure sway velocity. The added resistance of traveling in sway requires larger control inputs, over a prolonged period of time. Compared to the NMPC, the non-linear PD has a longer settling time and uses higher control inputs. Note the the noise on the position measurements introduce jitter in the control signals. As a result, the vessel is never truly stationary, but always correcting its position using small bursts of thrust. Improved position sensors could mitigate this problem.

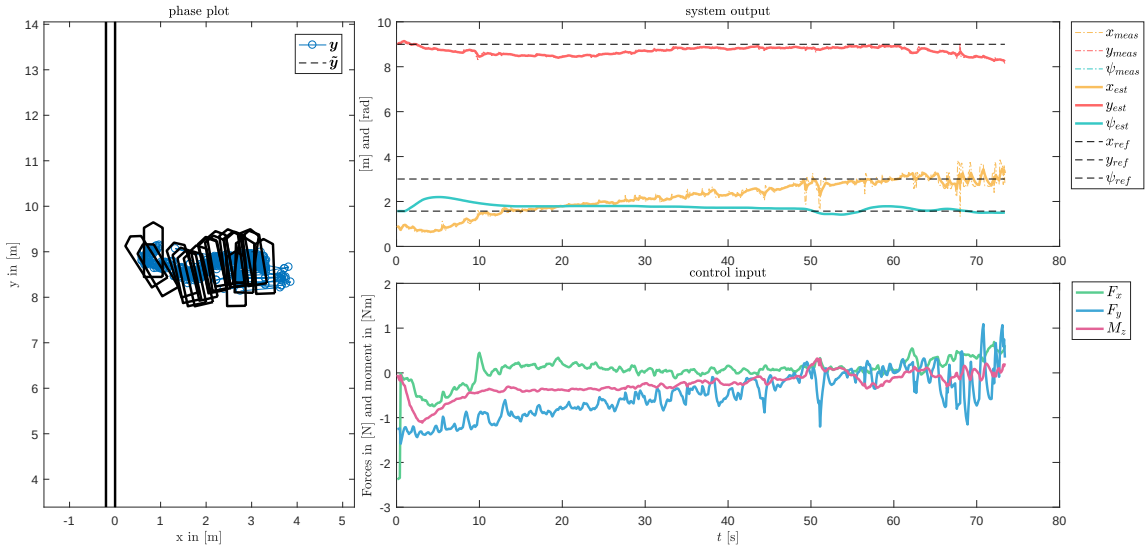

Figure 7.4: States and control inputs for the static reference tracking test under non-linear PD control with initial state $\boldsymbol{x}_I = [\,1 \quad 9 \quad \frac{1}{2}\pi \quad 0 \quad 0 \quad 0\,]$, state reference $\tilde{\boldsymbol{x}} = [\,3 \quad 9 \quad \frac{1}{2}\pi \quad 0 \quad 0 \quad 0\,]$ and a zero input reference $\tilde{\boldsymbol{u}} = \boldsymbol{0}^{3\times 1}$.

7.5. Time-Varying Reference Tracking Results

The NMPC for tracking time-varying references is tuned with larger penalties on the positional tracking error

$$\begin{aligned} \boldsymbol{Q} &= \text{diag}(\begin{bmatrix}30 & 30 & 50 & 1 & 1 & 1\end{bmatrix}) \\ \boldsymbol{R_1} &= \boldsymbol{I^{3\times 3}} \\ \boldsymbol{R_2} &= 2.5\boldsymbol{I^{3\times 3}} \\ N &= 20 \\ \Delta t &= 0.1 \end{aligned} \tag{7.9}$$

The performance of the NMPC will not be compared to the non-linear PD controller, as that type of controller only reacts to a reference error at the current time step. Since no information about the future reference is used to arrive at a control input, it is unable to converge to the time-varying references by default.

To determine the performance of the NMPC controller for time-varying reference, three tracking tests are performed. Each trajectory consists of a series concatenated constant surge speed motion primitives determined in section 6.3, at a cruise speed of $u_c = 0.1 \;\; m/s$. The resulting references are an equal representation of paths created by the kinodynamic RRT algorithm. Test results are displayed in figure 7.5, 7.6 and 7.7. In each test, the initial state of DASh is located at the start of the trajectory with a zero velocity. Note that the initial reference of the trajectory has a nonzero surge velocity equal to u_c. Therefore, DASh will have to catch up to the time-varying reference.

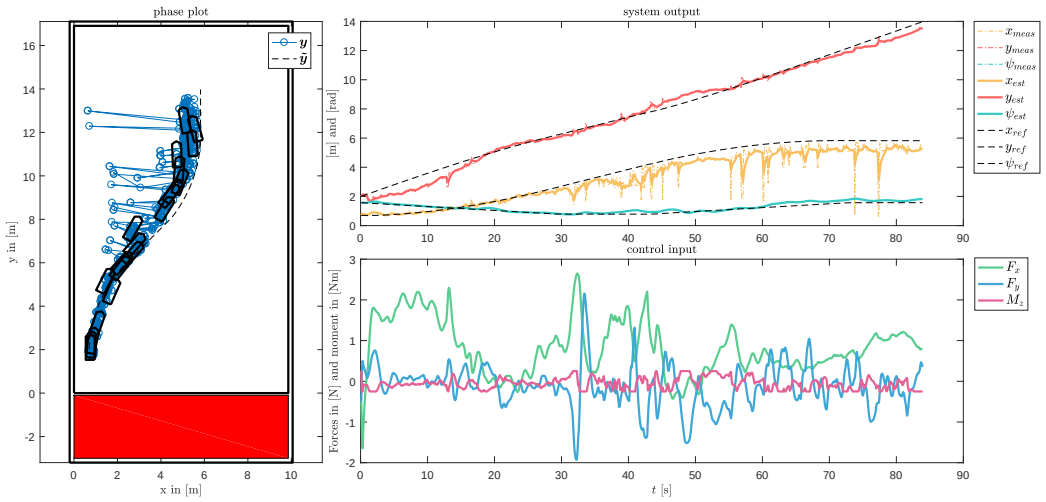

Figure 7.5: States and control inputs for the time-varying reference tracking test 1.

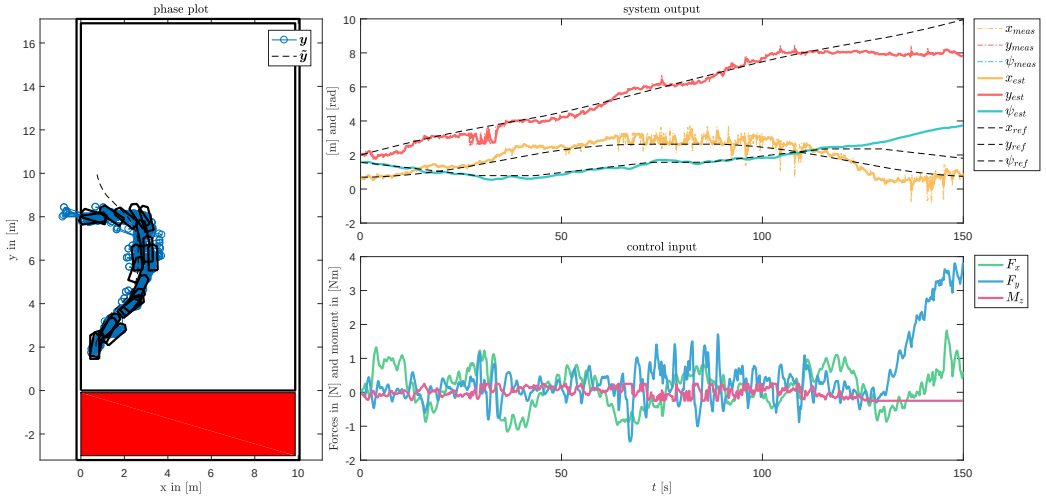

Figure 7.6: States and control inputs for the time-varying reference tracking test 2.

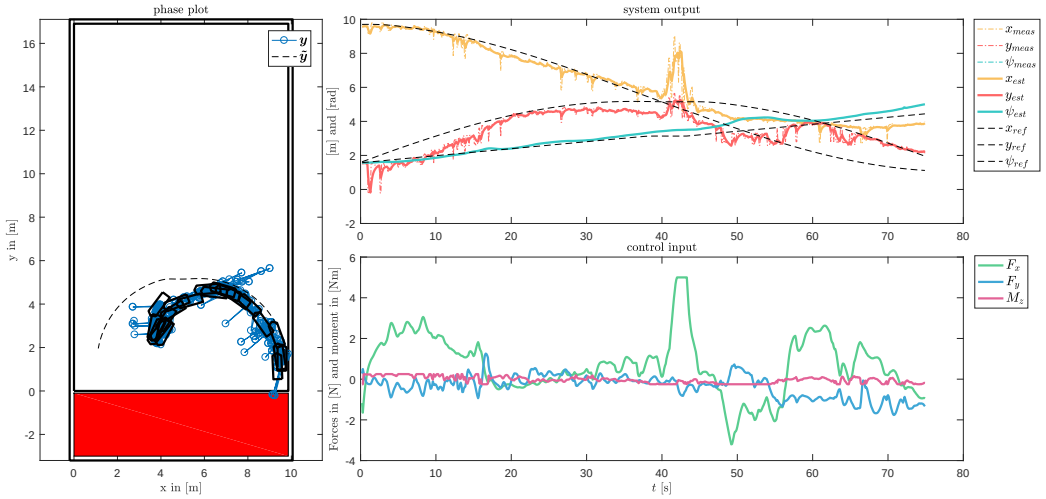

Figure 7.7: States and control inputs for the time-varying reference tracking test 3.

In test 1 of figure 7.5 good tracking performance is observed. An initial offset in x due to the zero surge velocity of the initial state, is quickly diminished. An increase in measurement noise is observed on the y state resulting in a slight steady state offset.

In test 2 good tracking performance is observed during the start of the test. However, the tracking error increases quickly after $t = \pm 120$ seconds, after which the vessel diverges from the time-varying reference. Similar behavior is observed in test 3 after $t = \pm 50$ seconds. The behavior is caused by inaccurate heading measurements of the digital compass.

Large changes in the magnetic field in which the basin is situated cause the internal sensor fusion of the BNO055 digital compass to trust solely on the integration of its gyro meter. The returned heading will then start to drift. Incorrect estimation on the heading cause the control system to become unstable. The magnetic field was mapped by measuring the magnetic north using an analog compass and the BNO055 digital compass. Results are displayed in figure 7.8. Measurements were started from node 1 to 9 and a clear drift of the digital compass is observed between successive nodes. The overall change of the magnetic north can be as high as 30 degrees. Bad heading measurements influence the stability of the NMPC controller. At too large heading offset, the vessel becomes instable.

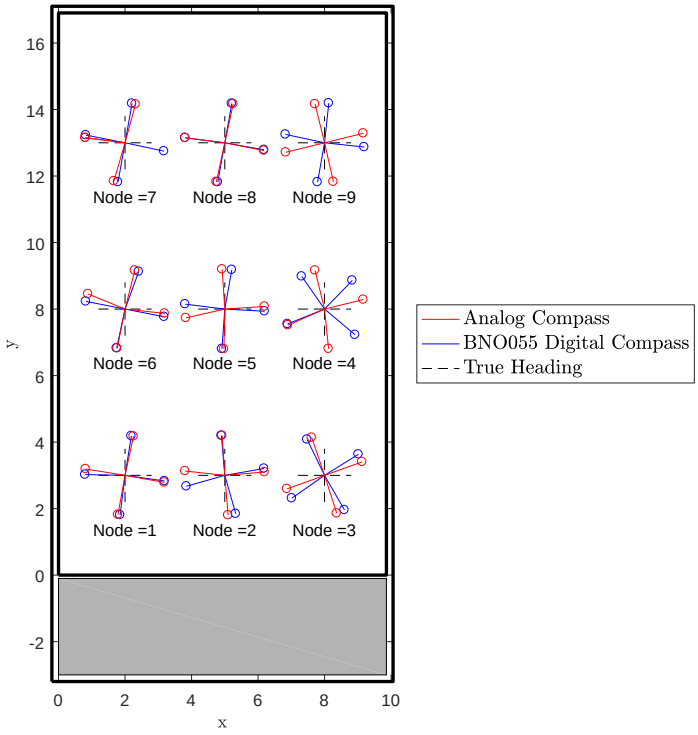

Figure 7.8: The analog compass and BNO055 heading measurements. The BNO055 was re-calibrated before measuring node 1 and 5.

7.6. Conclusion

The static and time-varying reference tracking test show the NMPC is able to control the position and speed of DASh. Static reference tracking test show the superior control capabilities of the NMPC compared to more traditional non-linear PD control. It can be concluded that the identified non-linear vessel model can indeed be used in model predictive control and stabilize the systems, even though long term model predictions are of bad quality, in as shown in section 4.6.1. An important tuning parameter in this case is the prediction horizon, limiting the influence of incorrect long term model predictions within the NMPC optimization.

Secondly, the NMPC is capable to track time-varying references produced by the kinodynamic RRT with only small offsets, as long as correct heading measurements are available. This observation underpins the hypothesis that kinodynamic planning results in executable paths, without large tracking errors.

Regarding the hardware off DASh, the current digital compass slowly drift from the true heading, after which the vessel becomes unstable. In future model tests, a more reliable heading sensor should be implemented. The large measurement noise on the position causes jitter in the control signal. By implementing more reliable sensors, its influence can be reduced.

8

Conclusions & Recommendations for Future Work

The goal of this thesis was to create and implement guidance, navigation and control systems that enable autonomous point-to-point sailing of the scale model called "Damen Autonomous Ship", *aka* DASh, shown in action in figure 8.1. In order to prototype the autonomous vessel, a model-based approach was used. The following research questions were answered

- How should the vessel dynamics be modelled and identified if it is to be used for guidance, navigation and control?

The formulation of a compact non-linear vessel model that captured the most important expected hydrodynamic phenomena, enabled the efficient identification of DASh without the use of towing tank experiments. The identified dynamics were used in the kinodynamic Rapidly-Exploring Random Tree (RRT) path planner to formulate a dynamically feasible and collision free trajectory from the initial state to the goal state.

- How can the dynamics of the vessel be taken into account during point-to-point path planning?

The identified dynamics were used in the kinodynamic Rapidly-Exploring Random Tree (RRT) path planner to formulate a dynamically feasible and collision free trajectory from the initial state to the goal state.

- Can the computational loads of dynamically constrained path planning be reduced to allow on-line implementation?

The computationally expensive kinodynamic RRT was modified to use a learned ideal distance metric based on the cost-to-go, and steering method constrained by a maneuver automaton. Particular attention was given to the influence of distance metrics during kinodynamic planning. The learned, supposedly ideal, cost-to-go distance metric was outperformed by the computationally cheap Dubins curve distance metric, which had superior performance in all simulation environments. The Euclidean metric was shown to be inept in capturing distance in the state space, badly under performing compared to the other distance metrics. The planner allowed differentially constrained paths to be planned in under half a minute using the cost-to-go metric. Using the Dubins curve distance metric planning could be sped up to find paths in under 3 seconds, fast enough to allow on-line implementation, due to the slow dynamics of real-sized vessels.

The navigation and control algorithms were implemented in a real-time manner on-board DASh. It was shown that a predetermined path could successfully be executed under non-linear model predictive control (NMPC), using the identified vessel dynamics. Measurement noise on the vessel position and drift of the heading sensor had a large impact on the performance of DASh, limiting model test durations to ± 1 minute.

Based on those results, it can be concluded that the presented framework for the identification, path planning and control using the identified vessel model, successfully enabled autonomous point-to-point sailing on DASh.

Figure 8.1: DASh at full throttle.

8.1. Implementation on Real Size Vessels

The algorithms found in the proposed framework can be implemented in real size vessels. However, at this scale the influence of environmental forces cannot be negated and should be accounted for in the vessel model. This can be done in a few ways. Firstly, all environmental forces can be grouped and represented as a heading dependent disturbance. Its heading and amplitude can be estimated using an observer [23, 52]. The disturbance can be accounted using a feedforward term in the controller. Environmental disturbances can also be accounted for during planning, ensuring dynamic feasibility of the path beforehand, instead of during path execution.

8.2. Collision Regulations & Other Ship Behavior

On every sea, river or canal, collision regulations (COLREGS) apply. Priority of vessel movements and right-of-way are governed by these regulations. The influence of the COLREGS or the behavior of other ships in reaction to the actions of our own ship, have not been investigated in this work. For real-world implementation it is recommended to modify our proposed approach to include these factors.

8.3. Account for Uncertainty during Planning

In the proposed framework, the kinodynamic RRT planner used a deterministic vessel model to determine future vessel states. In the real world, an inherent uncertainty is present which arises as a result of state estimation, environmental disturbances and modeling errors, as well as changes in the model such as component failures or a change in vessel mass. The uncertainty of model movement can be accounted for during planning by employing a probabilistic approach to the propagation of the vessel and obstacles. One can then plan paths that have the highest probability to be successfully executed.

8.4. Recommendations for DASh

The DASh scale model served as a good platform for the verification of the navigation and control algorithms. The linux-based board computer runs the control loop in efficient C and C++code, allowing real-time execution of the GNC algorithms. However, the current code is difficult to expand upon to test or prototype new GNC algorithms. The writer proposes to install a robotic friendly operation system on the board computer, such as the Robot Operating System (ROS), that allow quick prototyping.

The current hardware of DASh can be improved in several ways. The azimuth thrusters require significant torque in order to rotate, due to static friction. There is also a large amount of friction on the propeller shafts. This results in large overshoots of the propeller rotation speed upon startup, resulting in unwanted thruster forces. It is therefore advisable to install new azimuths, as well as a bow thruster to improve the maneuverability.

A bottleneck during implementation was the drifting heading measurement and noisy position measurements. To improve the performance and reliability of DASh it is recommended to replace the Pozyx sensor with a more robust measurement system using e.g. cameras, radar or LIDAR.

Bibliography

[1] MA Abkowitz. Lectures on ship hydrodynamics—-steering and manoeuvrability, report no. *Hy-5. Hydrodynamics Department, Hydro-and Aerodynamics Laboratory, Lyngby, Denmark*, 1964.

[2] Juan Manuel Ahuactzin, Depto De Ing, and Emmanuel Mazer. Manipulation Planning for Redundant Robots : A Practical Approach. *The International Journal of Robotics Research*, 17(7):731–747, 1998.

[3] Baris Akgun and Mike Stilman. Sampling Heuristics for Optimal Motion Planning in High Dimensions. In *International Conference on Intelligent Robots and Systems*, 2011.

[4] Ross Allen and Marco Pavone. A Real-Time Framework for Kinodynamic Planning with Application to Quadrotor Obstacle Avoidance. In *AIAA Guidance, Navigation, and Control Conference*, pages 5021–5028, 2016.

[5] Ross E Allen, Ashley A Clark, Joseph A Starek, and Marco Pavone. A Machine Learning Approach for Real-Time Reachability Analysis. In *International Conference on intelligent Robots and systems*, pages 2202–2208, 2014.

[6] Mukunda Bharatheesha, Wouter Caarls, Wouter Jan Wolfslag, and Martijn Wisse. Distance Metric Approximation for State-Space RRTs using Supervised Learning. In *International Conference on Intelligent Robots and Systems*, 2014.

[7] S. K. Bhattacharyya and M. R. Haddara. Parametric identification for nonlinear ship maneuvering. *Journal of Ship Research*, 50(3):197–207, 2006.

[8] M. Blanke. *Ship Propulsion Losses Related to Automated Steering and Prime Mover Control*. PhD thesis, Technical University of Denmark, Lyngby, Denmark, 1981.

[9] Morten Breivik, Stig Kvaal, and Per Østby. From Eureka to K-Pos: Dynamic Positioning as a Highly Successful and Important Marine Control Technology. In *IFAC Conference on Manoeuvring and Control of Marine Craft*, volume 48, pages 313–323, 2015. doi: 10.1016/j.ifacol.2016.01.001. URL `http://www.sciencedirect.com/science/article/pii/S2405896316000021`.

[10] Brendan Burns and Oliver Brock. Single-Query Motion Planning with Utility-Guided Random Trees. In *International Conference on Robotics and Automation*, pages 3307–3312, 2007.

[11] Manuel Haro Casado. Identification of the nonlinear ship model parameters based on the turning test trial and the backstepping procedure. *Ocean Engineering*, 32:1350–1369, 2005. doi: 10.1016/j.oceaneng.2004.11.003.

[12] Peng Cheng and Steven M Lavalle. Reducing Metric Sensitivity in Randomized Trajectory Design. In *International Conference on Intelligent Robots and Systems*, 2001.

[13] P.I. Corke. *Robotics, Vision & Control*. 2017, publisher=Springer.

[14] Yuntao Dai, Liqiang Liu, and Shanshan Feng. On the Identification of Coupled Pitch and Heave Motions Using Opposition-Based Particle Swarm Optimization. *Mathematical Problems in Engineering*, 2014: 1–10, 2014.

[15] Christiaan de Wit. Optimal Thrust Allocation Methods for Dynamic Positioning of Ships. *Electric Engineering, Mathematics and Computer Science*, (July), 2009.

[16] Moritz Diehl, Hans Georg Bock, Holger Diedam, Pierre-brice Wieber, Moritz Diehl, Hans Georg Bock, Holger Diedam, Pierre-brice Wieber Fast, and Direct Multiple. Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. 2009.

[17] E W Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, (1):269–271, 1959.

[18] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[19] M Elbanhawi and M Simic. Sampling-Based Robot Motion Planning: A Review. In *IEEE acces*, volume 2, pages 56–77, 2014. ISBN 2169-3536. doi: 10.1109/ACCESS.2014.2302442. URL `files/545/ElbanhawiandSimic-2014-Sampling-BasedRobotMotionPlanningAReview.pdf{%}5Cnfiles/546/articleDetails.html`.

[20] Marie Eve, Thomas Posch, Jakob Pernthaler, and Roland Y Siegwart. Autonomous Inland Water Monitoring: Design and Application of a Surface Vessel. *Robotics & Automation Magazine*, 19(1):62–72, 2012.

[21] K.K. Fedyaevsky and G.V. Sobolev. Control and stability in ship design. 1964.

[22] Thor Fossen and Tristan Perez. Kalman filtering for positioning and heading control of ships and offshore rigs. *IEEE Control Systems Magazine*, 29(6):33–46, 2009. ISSN 08880611. doi: 10.1109/MCS.2009.934408.

[23] Thor I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. 2011. ISBN 9781119991496.

[24] Thor I Fossen. Handbook of marine craft hydrodynamics and motion control, 2011.

[25] Thor I. Fossen, Svein I. Sagatun, and Asgeir J. Sørensen. Identification of dynamically positioned ships. *Modeling, Identification and Control*, 17(2):153–165, 1996. ISSN 03327353. doi: 10.1016/0967-0661(96)00014-7.

[26] Emilio Frazzoli, Munther Dahleh, and Eric Feron. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. Phd thesis, Massachusetts Institute of Technology, 2001.

[27] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed RRT *: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. In *International Conference on intelligent Robots and systems*, 2014.

[28] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch Informed Trees (BIT *): Sampling-based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs. In *International Conference on Robotics and Automation*, 2015.

[29] Reza Ghaemi, Soryeok Oh, and Jing Sun. Path following of a model ship using model predictive control with experimental verification. *Marine Engineering*, pages 5236–5241, 2010.

[30] Elena Glassman and Russ Tedrake. A Quadratic Regulator-Based Heuristic for Rapidly Exploring State Space. In *International Conference on Robotics and Automation*, pages 5021–5028, 2010. ISBN 9781424450404.

[31] Inger Berge Hagen. Collision Avoidance for ASVs Using Model Predictive Control. (February), 2017.

[32] Peter E Hart, Nils J Nilsson, and Raphael Bertrom. Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[33] Gregory Hitz, François Pomerlesau, Francis Colas, and Roland Siegwart. State estimation for shore monitoring using an autonomous surface vessel. *Springer Tracts in Advanced Robotics*, 109:745–760, 2016. ISSN 1610742X. doi: 10.1007/978-3-319-23778-7_49.

[34] S.F. Hoerner. *Fluid Dynamic Drag*. Hartford House, 1965.

[35] Wei Yuan Hwang. Cancellation effect and parameter identifiability of ship steering dynamics. *International Shipbuilding Progress*, 29(332):90–102, 1981.

[36] S Inoue, M Hirano, K Kijima, and J Takashina. A practical calculation method of ship maneuvering motion. *Int. Shipbuild Prog.*, 28(325):207–222, 1981.

[37] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree : A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, 34(7):883–921, 2015. doi: 10.1177/0278364915577958.

[38] Tor A. Johansen and Thor I. Fossen. Control allocation - A survey. *Automatica*, 49(5):1087–1103, 2013. ISSN 00051098. doi: 10.1016/j.automatica.2013.01.035. URL http://dx.doi.org/10.1016/j.automatica.2013.01.035.

[39] Steven G. Johnson. The NLopt nonlinear-optimization package. http://ab-initio.mit.edu/nlopt, 2017. [Online; accessed 21-March-2018].

[40] J.M.J. Journée and W .W. Massie. *Offshore Hydromechanics*. 2001. doi: 10.1016/S0013-4686(01)00879-9.

[41] Sertac Karaman and Emilio Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research*, 30(7):20, 2010. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364911406761. URL http://arxiv.org/abs/1005.0416.

[42] Lydia E Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H Overmars. Probablistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 2002. doi: 10.1109/70.508439.

[43] Jongwoo Kim and James Keller. Design and Verification of Controllers for Airships. In *International Conference on Intelligent Robots and Systems*, volume 1, pages 54–60, 2003.

[44] S Klanke and S Vijayakumar. A Library for Locally Weighted Projection Regression - Supplementary Documentation. http://wcms.inf.ed.ac.uk/ipab/slmc/research/software-lwpr, 2008. [Online; accessed 30-March-2018].

[45] Yoshiaki Kuwata, Michael T. Wolf, Dimitri Zarzhitsky, and Terrance L. Huntsberger. Safe maritime autonomous navigation with COLREGS, using velocity obstacles. *IEEE Journal of Oceanic Engineering*, 39 (1):110–119, 2014. ISSN 03649059. doi: 10.1109/JOE.2013.2254214.

[46] Pozyx Labs. Pozyx Indoor Positioning System. https://www.pozyx.io/, 2017. [Online; accessed 13-December-2017].

[47] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006. ISBN 0521862051.

[48] Steven M. LaValle and James J. Kuffner. Randomized Kinodynamic Planning. *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'99)*, (May):1–7, 1999. ISSN 1050-4729. doi: 10.1109/ROBOT.1999.770022. URL http://ijr.sagepub.com/cgi/content/abstract/20/5/378.

[49] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems systems. In *In Proceedings of the International conference on informatics in control, Automation and Robotics*, pages 1–8, 2004.

[50] Weilin Luo. Parameter Identifiability of Ship Manoeuvring Modeling Using System Identification. *Mathmatical Problems in Engineering*, 2016, 2016.

[51] Weilin Luo, Carlos Guedes Soares, and Zaojian Zou. Parameter Identification of Ship Manoeuvring Model Based on Particle Swarm Optimization and Support Vector Machines. *Volume 5: Ocean Engineering*, 138(October):V005T06A071, 2013. ISSN 0892-7219. doi: 10.1115/OMAE2013-11078. URL http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?doi=10.1115/OMAE2013-11078.

[52] Shah Muhammad. *Dynamic Positioning of Ships: A nonlinear control design study*. PhD thesis, 2012.

[53] Nils J Nilsson. A Mobile Automaton: An Application of Artificial Intelligence Techniques, 1969.

[54] NH Norrbin. Theory and observation on the use of a mathematical model for ship maneuvering in deep and confined waters, 8th symp. *Naval Hydrodynamics, Pasadena, California, USA*, 1970.

[55] P Oltmann. On the influence of speed on the manoeuvring behaviour of a container carrier. 1996.

[56] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H Ang Jr. Perception, Planning, Control, and Coordination for Autonomous Vehicles. *Machines*, 5(6), 2017. doi: 10.3390/machines5010006.

[57] Quang Cuong Pham. A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Transactions on Robotics*, 30(6):1533–1540, 2014. ISSN 15523098. doi: 10.1109/TRO.2014.2351113.

[58] James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990. ISSN 0030-8730. doi: 10.2140/pjm.1990.145.367. URL http://msp.org/pjm/1990/145-2/p06.xhtml.

[59] John H Reif and Micha Sharir. Motion Planning in the Presence of Moving Obstacles. In *25th IEEE Symposium on Foundations of Computer Science*, pages 144–151, 1985.

[60] Samuel Rodriguez, Xinyu Tang, Jyh-Ming Lien, and Nancy M Amato. An Obstacle-Based Rapidly-Exploring Random Tree. In *International Conference on Robotics and Automation*, pages 895–900, 2006.

[61] Anita M Rothblum. Human Error and Marine Safety, 2000. URL www.bowles-langley.com/wp-content/files{_}mf/humanerrorandmarinesafety26.pdf.

[62] D J Scheeres and M J Holzinger. The Control Distance Metric and Constraints on Maneuvering Satellites. In *International Conference on Information Fusions*, pages 2035–2042, 2012.

[63] Matteo Schiaretti, Linying Chen, and Rudy R. Negenborn. *Survey on Autonomous Surface Vessels: Part II - Categorization of 60 Prototypes and Future Applications*. 2017. ISBN 9783319684956.

[64] Roger Skjetne, Øyvind N. Smogeli, and Thor I. Fossen. A Nonlinear Ship Manoeuvering Model: Identification and adaptive control with experiments for a model ship. *Modeling, Identification and Control*, 25 (1):3–27, 2004. ISSN 03327353. doi: 10.4173/mic.2004.1.1.

[65] Roger Skjetne, Øyvind N. Smogeli, and Thor I. Fossen. MODELING, IDENTIFICATION, AND ADAPTIVE MANEUVERING OF CYBERSHIP II: A COMPLETE DESIGN WITH EXPERIMENTS. pages 203–208, 2004.

[66] SNAME The Society of Naval Architects and Marine Engineers. Nomenclature for Treating the Motion of a Submerged Body Through a Fluid, 1950. URL http://kom.aau.dk/{~}nickoe/random/Sname1950.PDF.

[67] Thomas Statheros, Gareth Howells, and Klaus Mcdonald-maier. Autonomous Ship Collision Avoidance Navigation Concepts , Technologies and Techniques. (2008):129–142, 2017. doi: 10.1017/S037346330700447X.

[68] H Taams. Path planning for the asd-tug - literature review. 2017.

[69] Ken Teo, Kai Wei Ong, and Hoe Chee Lai. Obstacle Detection , Avoidance and Anti Collision for MEREDITH AUV. In *OCEANS*, pages 1–10, 2009.

[70] A Tiano and M Blanke. Multivariable identification of steering and roll motions. *Transactions of the Institute of Measurement and Control*, 19(2):63–77, 1997. doi: 10.1177/014233129701900202.

[71] Serge Toxopeus. Validation of slender-body method for prediction of linear manoeuvring coefficients using experiments and viscous-flow calculations. In *ICHD2006 7th International Conference on Hydrodynamics*, pages 589–598, 2006.

[72] Marine Traffic. Automatic Identification System. www.marinetraffic.com, 2017. [Online; accessed 5-Februari-2017].

[73] Ioan A. Şucan and Lydia E. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. *Springer Tracts in Advanced Robotics*, 57:449–464, 2010. ISSN 16107438. doi: 10.1007/978-3-642-00312-7_28.

[74] Jur Van Den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. *Springer Tracts in Advanced Robotics*, 70(STAR):3–19, 2011. ISSN 16107438. doi: 10.1007/978-3-642-19457-3_1.

[75] Ton van den Boom. *Lecture Note for the Course Model Predictive Control*. DCSC Delft University of Technology, 2013.

[76] Aleksander Veksler, Tor Arne Johansen, Francesco Borrelli, and Bjornar Realfsen. Cartesian thrust allocation algorithm with variable direction thrusters, turn rate limits and singularity avoidance. *2014 IEEE Conference on Control Applications, CCA 2014*, (3):917–922, 2014. doi: 10.1109/CCA.2014.6981453.

[77] Sethu Vijayakumar, Aaron D Souza, and Stefan Schaal. Incremental Online Learning in High Dimensions, 2005.

[78] W J Wolfslag, M Bharatheesha, T M Moerland, and M Wisse. RRT-CoLearn : towards kinodynamic planning without numerical trajectory optimization. *IEEE Robotics and Automation Letters*, 3(3), 2018.

[79] Hyeon Kyu Yoon and Key Pyo Rhee. Identification of hydrodynamic coefficients in ship maneuvering equations of motion by Estimation-Before-Modeling technique. *Ocean Engineering*, 30:2379–2404, 2003. doi: 10.1016/S0029-8018(03)00106-9.

[80] H Yasukawa Y Yoshimura. Introduction of MMG standard method for ship maneuvering predictions. *Journal of Marine Science and Technology*, 20(1):37–52, 2015. doi: 10.1007/s00773-014-0293-y.

[81] Huarong Zheng, Rudy R Negenborn, and Gabriel Lodewijks. Trajectory tracking of autonomous vessels using model predictive control. *19 Th World Congress the Internation Federation of Automatic Control*, pages 8812–8818, 2014. ISSN 14746670. doi: 10.3182/20140824-6-ZA-1003.00767.

A

Measurement Setup for Propeller Thrust

The thrust delivered by a propeller depends on the flow speed of the fluid entering the propeller and it rotational speed. Due to the absence of a towing tank, only free running test could be performed. As a result, it was not possible to superimpose an advance speed on the propeller. Instead thrust measurements at zero advance speed were performed. These test are called bollard pull tests and are often performed by tugs.

Using a scale, rope and pulley system shown in figure A.1, the tension in the line as a result of propulsion forces can be measured while the vessel is stationary.

A large weight (20 kg) was placed on the scale. A rope attached to the weight was placed over both pulleys and attached to the deck of DASh, directly above its *COG*. The rope between the weight and the left pulley runs parallel to the direction of gravity. Neglecting the mass of the rope, the tension in the rope will decrease the measured weight on the scale, by and equal amount. The accuracy of the scale is 0.1 gram. Control measurements on the scale, rope and pulley system were performed with known weights, suspended from the right pulley. The decrease in weight on the scale due to the tension in the rope was *exactly* the same as weighing their mass directly.

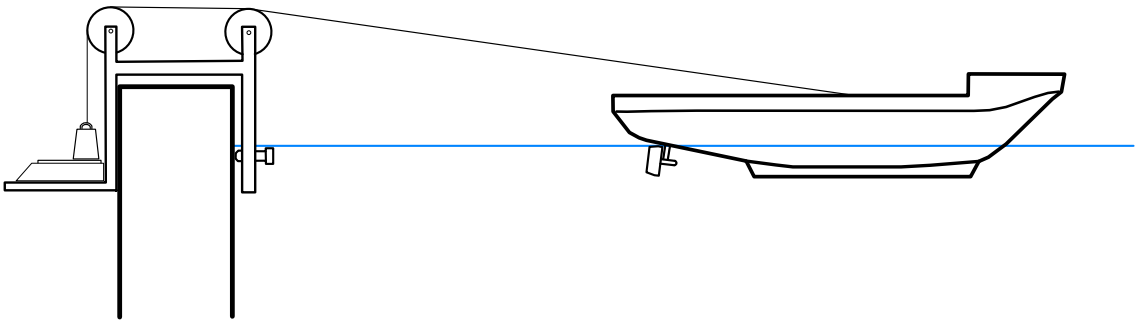

Figure A.1: A scale, rope and pulley force measuring setup to determine propeller thrust at various rotational velocities.

All bollard pull tests were performed by delivering forward thrust with both propellers. Due to the elasticity of the rope, the thrust value oscillated when the thrust changed. The thrust was only measured when the oscillations died down and it reached a constant value.

B

Thrust Allocation

Control algorithms for vessels often return a generalized force command $\boldsymbol{u} = [X \quad Y \quad N]^T$. The generalized force consists of forces and moments that must be exerted on the vessel, in order to achieve its control objective. The task of the thrust allocation algorithm is to generate commands for each individual thruster so that the resultant force matches the generalized force command. Using the same notation for the thruster command as found in section 3.5, the thrust allocation algorithm solves the underdetermined system of linear equations

$$\boldsymbol{u} = \boldsymbol{B_T T} \tag{B.1}$$

which has an infinite amount of solutions. Thrust allocation algorithms fund the optimal solution limits the overall amount of thrust and large input variations. An optimization based solution, similar to [76], is implemented in DASh. The thrust allocation algorithm is ran at every sampling period T_s. The azimuths are constrained by a maximum thrust constraint $\sqrt{T_{i,x}^2 + T_{i,y}^2} \leq T_{max}$ and a maximum rotation constraint $|\dot{\alpha}_i| \leq \dot{\alpha}_{max}$ for thrusters $i = 1, 2$. Both constraints can be linearized:

Maximum thrust can be constrained by defining a set of N linear inequality constraints of the form

$$\begin{bmatrix} \cos(\rho_1) & \sin(\rho_1) \\ \cos(\rho_2) & \sin(\rho_2) \\ \vdots & \vdots \\ \cos(\rho_N) & \sin(\rho_N) \end{bmatrix} \begin{bmatrix} T_{i,x} \\ T_{i,y} \end{bmatrix} \leq T_{max} \tag{B.2}$$

where ρ is an equally spaced vector of N elements in the domain $[0,\ 2\pi)$. For a maximal approximation error of ϵ, the amount of linear equations should exceed $N = \text{round}(\pi / \arccos(1 - \frac{\epsilon}{T_{max}}))$. An elaboration upon the approximation error can be found in [15]. A visual example of the linear approximation for the maximum thrust constraint is shown in figure B.1.

Rotation Constraints are implemented by restricting the rotation per iteration to the largest possible rotation angle $\Delta\alpha = T_s \dot{\alpha_{max}}$. Resulting in the inequality constraints

$$\begin{bmatrix} \sin(\alpha_i - \Delta\alpha) & -\cos(\alpha_i - \Delta\alpha) \\ -\sin(\alpha_i + \Delta\alpha) & \cos(\alpha_i + \Delta\alpha) \end{bmatrix} \begin{bmatrix} T_{i,x} \\ T_{i,y} \end{bmatrix} \leq 0 \tag{B.3}$$

In figure B.1 the reachable azimuth angles are displayed in green.

To avoid an optimization iteration to become infeasible, a slack variable is introduced into to following equality constraint

$$\boldsymbol{u} = \boldsymbol{B_T T} + \boldsymbol{s} \tag{B.4}$$

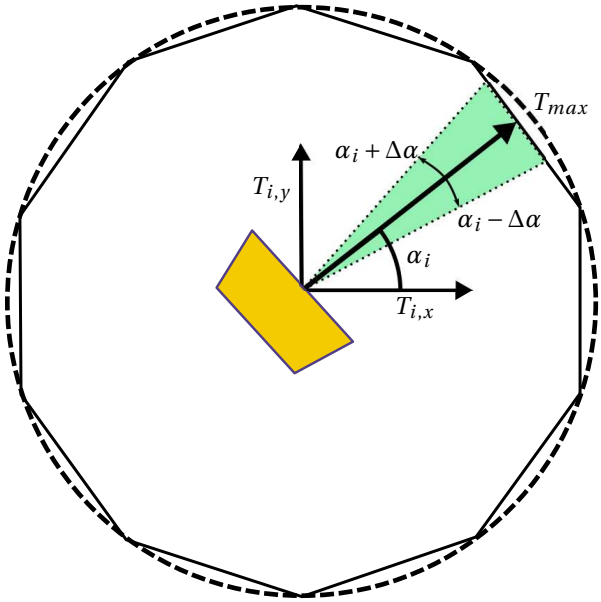

Figure B.1: The maximum thrust constraint of an azimuth thruster is shown as a black dotted line. The linear approximation is shown in a black solid line. Due to a maximum rotations step $\Delta\alpha$ per iteration, only thruster settings in the green cone can be reached within a single sampling period T_s.

The goal is to match the generalized force command using minimal thrust, thus minimizing the entries of $\boldsymbol{T}$ and $\boldsymbol{s}$. Large input changes between the previous input $\boldsymbol{T}_{prev}$ and $\boldsymbol{T}$ increase the wear on the azimuths, and should therefore be avoided. To this end the following cost function is minimized

$$J(\boldsymbol{s},\boldsymbol{u}) = \boldsymbol{T}^T\boldsymbol{H}\boldsymbol{T} + (\boldsymbol{T}-\boldsymbol{T}_{prev})^T\boldsymbol{M}(\boldsymbol{T}-\boldsymbol{T}_{prev}) + \boldsymbol{s}^T\boldsymbol{Q}\boldsymbol{s} \tag{B.5}$$

Let $\boldsymbol{I}$ be an identity matrix of the correct dimension. The matrices $\boldsymbol{H} = \boldsymbol{I}$, $\boldsymbol{M} = 0.1\boldsymbol{I}$ and $\boldsymbol{Q} = \text{diag}([20,\ 40,\ 30])$ are readily tuned diagonal weighing matrices. The resulting optimization problem can be solved using Sequential Quadratic Programming. Implementation of this algorithm on the Beaglebone Black is performed using the open source Non-Linear Optimization Toolbox (NLopt) [39]. The algorithm is terminated when changes of the estimated input are less then $10e^{-5}$, or after a runtime of 0.09 seconds. The latter ensures the algorithm can run in real time at 10 hz, but in some cases sacrifices optimality .

C

Coordinate Transformations

Let $\boldsymbol{p}$ be a point defined in the earth-fixed frame {e} as $\boldsymbol{p}_e = [x \quad y]^T$ and in the body-fixed frame {b} as $\boldsymbol{p}_b = [x_b \quad y_b]^T$. If the coordinates of $\boldsymbol{p}_e$, the origin of the body-fixed frame $O_b^e = [x_{O_b}^e \quad y_{O_b}^e]^T$ expressed in {e} and the rotation θ of {b} relative to {e} is given, $\boldsymbol{p}_b$ can be calculated.

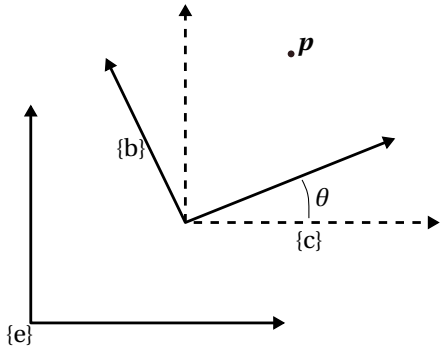

Figure C.1: Point $\boldsymbol{p}$ and the earth-fixed {e}, body-fixed {b} and intermediate coordinate frame {c}.

To simplify the transformation, a intermediate coordinate frame {c} is defined, which has its axis aligned with {e} and its origin overlaps with origin of {b}. A visual representation of the frames is shown in figure C.1. By extending the position vector $p_e = [x \quad y \quad 1]^T$ it can easily be expressed in {c} using the translation matrix

$$\boldsymbol{p}_c = \boldsymbol{T}_e^c \boldsymbol{p}_e = \begin{bmatrix} 1 & 0 & -x_{O_b}^e \\ 0 & 1 & -y_{O_b}^e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{C.1}$$

To express $\boldsymbol{p}$ in the body-fixed frame, only a rotational transform of angle θ remains to complete the coordinate transformation.

$$\boldsymbol{p}_b = \boldsymbol{R}(\theta)\boldsymbol{p}_c = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \tag{C.2}$$

D

LWPR settings

For reproducibility all settings for the training of the LWPR model are provided in table D.1. Based on a training data set of 12198 unique cost-to-go points, 80% was used for training, and 20% for testing. The tunable settings are elaborately discussed in the supplementary documentation [44]. Receptive field sizes are updated using the Incremental Delta Bar Delta (IDBD) algorithm by setting the meta_rate to TRUE. Unmentioned settings have the default values. The resulting model has 2104 receptive fields and a single query is returned within 0.0204 seconds.

Table D.1: Settings of the LWPR model upon training.

Setting	Value
norm_in	$[10 \quad 10 \quad 2\pi]$
init_D	$400\boldsymbol{I}^{\mathbf{3\times 3}}$
diag_only	TRUE
wgen	0.15
w_prune	0.8
meta	TRUE
meta_rate	200
update_D	FALSE
kernel	Gaussian
cutoff	0.01

E

Non-Linear PD Control

In this section the multiple-input multiple-output Non-linear PD Controller with Acceleration Feedback as found in [23] is synthesized for the identified vessel model of chapter 4. The following control law is considered

$$\boldsymbol{\tau}_{control} = -\boldsymbol{H}_m \dot{\boldsymbol{\nu}} + \boldsymbol{R}^T(\psi)\boldsymbol{\tau}_{pd} \tag{E.1}$$

$$\boldsymbol{\tau}_{pd} = -\boldsymbol{K}_p \tilde{\boldsymbol{\eta}} - \boldsymbol{K}_d \dot{\eta} \tag{E.2}$$

where $\boldsymbol{H}_m$, $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are constant positive definite matrices and $\boldsymbol{\tau}_{pd}$ is the feedback from the PD controller based on the state error $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta} - \boldsymbol{\eta}_d$. This results in the following closed loop system

$$M^* \dot{\boldsymbol{\nu}} + \left[\boldsymbol{C}(\boldsymbol{\nu}) + \boldsymbol{D}(\boldsymbol{\nu}) + \boldsymbol{K}_d^*(\psi)\right]\boldsymbol{\nu} + \boldsymbol{R}^T(\psi)\boldsymbol{K}_p\tilde{\boldsymbol{\eta}} = 0 \tag{E.3}$$

where

$$\boldsymbol{K}_d^*(\psi) = \boldsymbol{R}^T(\psi)\boldsymbol{K}_d\boldsymbol{R}(\psi)$$
$$\boldsymbol{M}^* = \boldsymbol{M} + \boldsymbol{H}_m$$

using the following Lyapunov function candidate which consists of a kinetic and potential energy term

$$V(\tilde{\boldsymbol{\eta}}, \boldsymbol{\nu}) = \frac{1}{2}\boldsymbol{\nu}^T\boldsymbol{M}^*\boldsymbol{\nu} + \frac{1}{2}\tilde{\boldsymbol{\eta}}^T\boldsymbol{K}_p\tilde{\boldsymbol{\eta}} \tag{E.4}$$

Time differentiation along $\tilde{\boldsymbol{\eta}}$ and $\boldsymbol{\nu}$ gives:

$$\dot{V} = \boldsymbol{\nu}^T\boldsymbol{M}^*\dot{\boldsymbol{\nu}} + \dot{\eta}^T\boldsymbol{K}_p\tilde{\boldsymbol{\eta}} \tag{E.5}$$

$$= \boldsymbol{\nu}^T\left(\boldsymbol{M}^*\dot{\boldsymbol{\nu}} + \boldsymbol{R}^T(\psi)\boldsymbol{K}_p\tilde{\boldsymbol{\eta}}\right) \tag{E.6}$$

substitute (E.3) into (E.5) yields

$$\dot{V} = \boldsymbol{\nu}^T\left(-\left[\boldsymbol{C}(\boldsymbol{\nu}) + \boldsymbol{D}(\boldsymbol{\nu}) + \boldsymbol{K}_d^*(\psi)\right]\boldsymbol{\nu}\right) \tag{E.7}$$

Since $\boldsymbol{C}(\boldsymbol{\nu})$ is skew symmetric, $\boldsymbol{\nu}^T\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} = 0$ and we obtain

$$\dot{V} = -\boldsymbol{\nu}^T\left[\boldsymbol{D}(\boldsymbol{\nu}) + \boldsymbol{K}_d^*(\psi)\right]\boldsymbol{\nu} \tag{E.8}$$

Using LaSalle's invariance principle, all trajectories of the system will converge to

$$\dot{V} = -\boldsymbol{\nu}^T[\boldsymbol{D}(\boldsymbol{\nu}) + \boldsymbol{K}_d^*(\psi)]\boldsymbol{\nu} \equiv 0 \tag{E.9}$$

for all trajectories in the set

$$\Omega = \{(\tilde{\boldsymbol{\eta}}, \boldsymbol{\nu}) : \boldsymbol{\nu} = 0\} \tag{E.10}$$

For $\boldsymbol{\nu} = \boldsymbol{0}$ implies $\dot{\boldsymbol{\nu}} = \boldsymbol{0}$ and from (E.3) we conclude that $\boldsymbol{M}^*\dot{\boldsymbol{\nu}} = -\boldsymbol{R}^T(\psi)\boldsymbol{K}_p\tilde{\boldsymbol{\eta}}$ only holds for $\tilde{\boldsymbol{\eta}} = \boldsymbol{0}$. Therefore the largest invariant set in Ω is the equilibrium point $(\tilde{\boldsymbol{\eta}}, \boldsymbol{\nu}) = (\boldsymbol{0}, \boldsymbol{0})$. Since $V(\boldsymbol{x}) > 0$, for all $\boldsymbol{x} \neq 0$, $V(\boldsymbol{0}) = 0$ and $V(\boldsymbol{x}) \to \infty$, as $\|\boldsymbol{x}\| \to \infty$, the equilibrium point is Globally Asymptotically Stable (GAS).