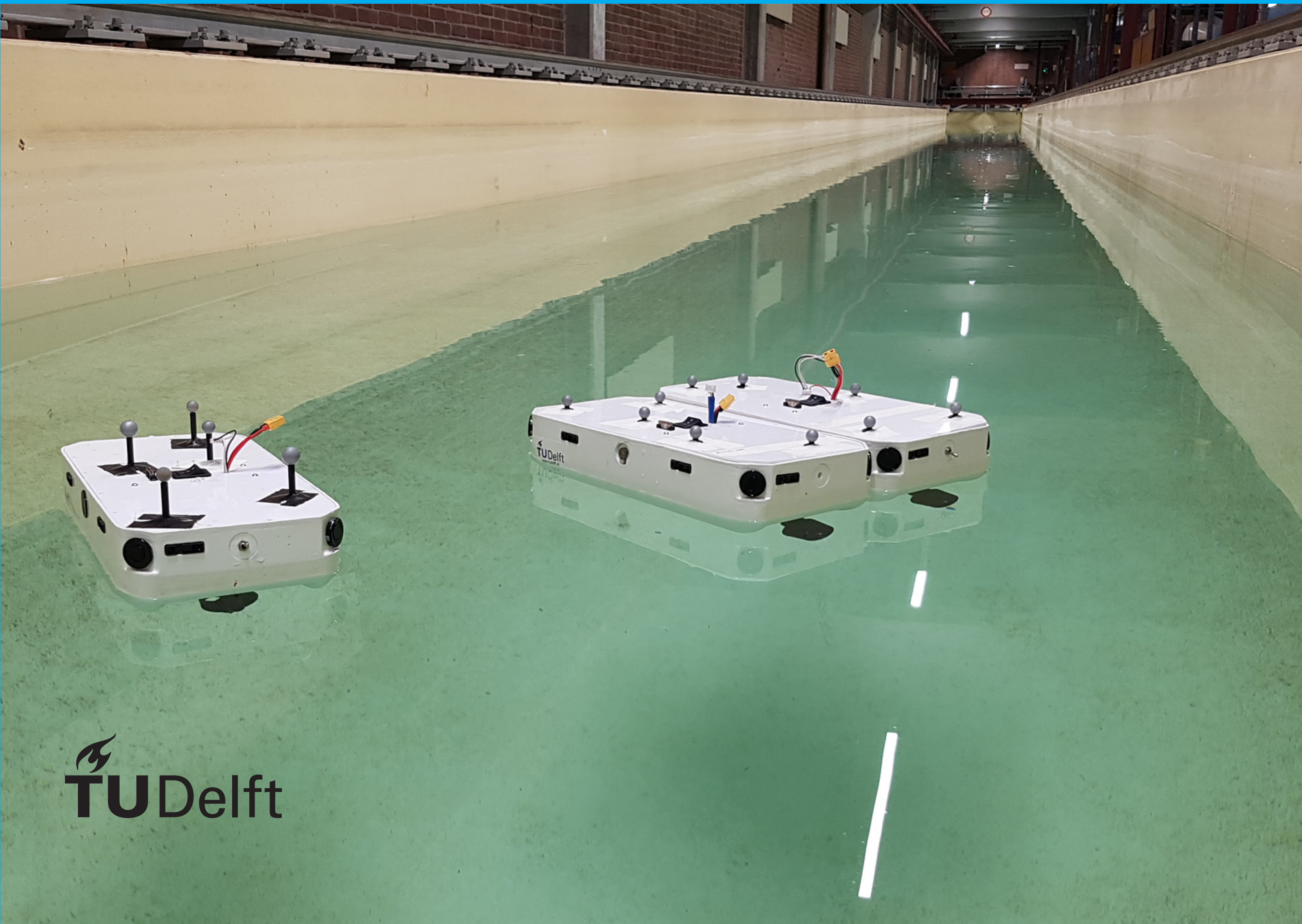
Vessel-Platform Automation: Integrating self assembly with configuration dependent control strategies

Bart Boogmans



Vessel-Platform Automation: Integrating self assembly with configuration dependent control strategies

by

Bart Boogmans

Master Thesis

in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology
to be defended publicly on Wednesday November 17, 2021 at 10:45 AM

Student number:	4156986	
MSc track:	Multi-Machine Engineering	
Report number:	2021.MME.8543	
Supervisor:	Ir. V. Garofano	
	Dr.ir Y. Pang	
Thesis Committee:	Prof.dr. R. Negenborn	TU Delft Committee Chair, MTT, 3ME
	Dr.ir Y. Pang	TU Delft Academic Supervisor, MTT, 3ME
	Ir. V. Garofano	TU Delft Daily Supervisor, MTT, 3ME
	Dr. Andrea Coraddu	TU Delft External Committee Member, SDPO, 3ME
Date	3 November 2021	

Executive Summary

Shipping enterprises are under constant pressure to perform faster and more flexible transportation with lower costs and emissions. This incentive, together with advancements such as in communication and network technologies, have led to various projects that explore vessel control automation in the marine sector as a solution.

In the past decades, many projects worked to further explore automating various parts of vessel control. Further automation of ship control tasks can broaden viable applicable scenarios, reduce the need of human operators, increase transport system effectivity, reliability and safety, while consuming less resources. Realistically, automation will rarely improve all the named benefits at once. Projects tend to aim to enhance several of these measures of performance that are most relevant for a use case, while allowing acceptable losses elsewhere. Automation, collaboration, negotiation and information sharing allow a wider variety of solutions to logistic challenges.

A recent concept is utilization of automated modular waterborne structures to perform tasks such as forming temporary infrastructure or create arbitrary shaped vessels. Such structures on the water surface form vessel platforms, built from a number of smaller connected vessels, and have been an area of research. Such vessel platform systems could in some cases be applied for quick, flexible and affordable solutions to logistic challenges where no other vessel solution is feasible for that task, environment or timeframe.

The main objective of this research is to develop a control framework for a modular fleet capable of assembly and collaborative motion control of connected structures. An approach on combining two systems (automated assembly & collaborative control) is presented throughout this paper supplemented with various insightsto aid further developments. First steps are identification of characteristics of the integrated systems, followed by proposing a representation of multi vessel state notation and platform dynamics, which are subsequentially all used to aid design decisions in developing an experimental setup. Finally the performance of this system is evaluated trough experiments.

Literature Survey

For effective discussion on ship automation good definitions of terminology are required to avoid misinterpretation. Terminology used in this work is thoroughly discussed. Used definitions are stated, supplemented with various other common interpretations. It is shown how interpretation of concepts such as 'autonomy', 'smart-ship' and 'automated' leave much room for discussion, with dangers of misaligned expectations between discussion partners. Ongoing efforts aim to standardize such terminology.

Literature shows various works on automated control of vessel platforming systems, yet it was noticed that the majority of relevant works focus on either the "assembly process" or "collaborative motion control", so they are discussed in those groups.

Two projects have been identified to build up the majority of the literature on automation of fleet assembly into floating structures. The first discussed works are on a project that developed a nameless fleet of container-like modules with rope and hook connectors, while the second works are developed in the "Roboat" project with vessels of similar naming. Both projects worked on model scale and focussed on developing various facets of assembly in multiple papers. Challenges regarding multi-vessel-structure reconfiguration are adressed from a hardware (e.g. connector between ships), latching strategy and task scheduling perspective. A wider variety of approaches to realize automated multi-robot reconfiguration can be found in general robotic science compared to marine systems.

As arbitrary shaped waterborne structures are formed, it can be beneficial to be able to manipulate motion of the combined body with collaborative approaches. Works describing motion control of modular systems are limited, as the majority of the encountered literature focusses on controlling motion of a single large object (e.g. a large unactuated barge) by utilizing combined effort of a fleet of automated vessels (e.g. tugboats). Collaborative control approaches of a single (non-modular) object by multiple entities are discussed as these works present insights and approaches to aid effective design choices. One publication proposes an approach to control a modular structure, utilizing PD control for control effort generation, control effort allocation with energy optimizition and an approximate-platform-model to scale control parameters to variable

configurations.

System Characterization

To look beyond the design solutions from existing projects and create a broader view on the design spectrum, a deeper look is given into the meaning of automated 'reconfiguration' and 'collaborative motion control' of modular vessel structures. Generalized system characteristics are distinct and narrowed down set of solutions that can be considered more feasible to perform in commercial applications, partially based on design choices of other projects. Automated modular vessel platform reconfiguration systems can be categorized as a more general "modular reconfigurable robot" system in a maritime setting. The key characteristic of such systems is described as adaptability of hardware structure to suit a given task or environment. MRR systems differentiate themselves from normal robot systems by determining and executing a course of action, to change it's configuration. Various requirements of floating MRR systems were identified during the course of the project. Some approaches originating from sources about robotics in general were broader than approaches encountered in sources about modular vessel platforming. During elaboration on system characteristics various projects are discussed, more specifically, how they solved their challenges and in which characteristics did this result in.

As various characteristics of automated 'reconfiguration' and 'collaborative motion control' systems are shown, more attention is given to integration of both systems. An additional characteristic is identified as these two behaviors are integrated in one framework. As a platform configuration changes through time, the motion control system needs to be able to support real time varying configuration parameter, such as estimated dynamics, size, shape, or centre of mass, or in some cases even the network topology. Various considerations for multi robot systems have been observed such as control structure topology (e.g. centralized, decentralized, hybrid), Homogeneouity of the fleet (similar modules or varying), actuation (e.g. choice and layout of connectors, thrusters) and utilization of network (task distribution, decentralization).

Multi-vessel system representation

Existing vessel state descriptions are used to formulate a multi-vessel state description. It aims to enable consistent system notation of scenarios pertaining more than a single vessel, as mult-vessel frameworks have more objects that can be referred to. Furthermore, the concept and definition of a platform-coordinate system is introduced. This resulted in three distinct types of coordinate systems; vessels, assemblies/platforms and global (e.g. inertial) frames that define state of an object and coordinate system in which parameters of other objects (pose, velocities, forces) can be expressed.

For collaborative motion control of modular objects an approximate dynamical model is often desired. As model parameter estimation experiments are often infeasible for every configuration of a combined structure, a prediction of dynamical behavior is proposed using the often known dynamics of modules. Assuming rigid connections between modules allow expression of module modules in a single point and orientation (such as the platform frame) analytical by means of coordinate system translation and rotation. A key requirement is that individual module models sufficiently describe the behavior of a model also in very close proximity of other modules, or discrepancies due to module proximity are sufficiently negated by additional compensating terms. Module model terms that are directionally dependent, such as hydrodynamic added mass, remain represented in the proposed platform module, taking into account the modules placement and orientation. The proposed platform model can form a building block for approximated dynamics of the combined structure utilizing obtainable parameters (individual module models), which can be extended to include vessel proximity effects if needed.

Both the multi-vessl state notation and the proposed platform model will be used in the next chapter during the development of a control system of the described structure.

Development of an automated assembling & collaborative motion control system

A multi-vessel platforming system is designed and implemented. Major considerations and characteristics of automated reconfiguration and collaborative motion control systems are used to support design choices to meet the following set criteria; The system is required to perform automated vessel platform reconfiguration, while simultaneously showing collaborative platform motion control behavior. The framework should support a large amount of-, or arbitrary configurations. This creates adaptiveness to a wide set of tasks for modular vessel platforms, which is a key element supporting commercial competitivity. Furthermore, developed solutions are aimed to be general, such that they are applicable or at least meaningful on other ship-systems, environments and scales of operation. This is considered important to stimulate that knowledge gained from the developed experimental setup benefits future commercial implementation. Finally, solutions are aimed to be modular, such that subsystems are designed to be conveniently swapped out for another (perhaps improved and better performing) version, easing future improvements and increasing reusability of work.

The goal of this design is not to optimize an existing system, but to explore a novel combination of behaviors that is expected to be of interest in the near future. The ability of succesful assembly needs to be proved, yet successrate and reconfiguration-speed need only be in reasonable limits and magnitude to reflect commercial implementation. Cooperative motion control needs to show convergence of the system to a desired state while a network of robots operate more effective than the sum of individuals, yet quantitiative motion responses need only be stable and within reasonable timeframe.

Content on design of the experimental setup starts from a high level view by discussing the multi-robot network topology adapting to varying configurations. It continues to explain the general control approach and division in subsystems that each solve a part of the control problem. Design choices of each subsystem are subsequently discussed in terms of model estimation, state estimation, control effort generation, control effort allocation and assembly protocol.

The modules used are Delfia-1* robotic model scale, homogeneous, rectangular vessels equipped with two rotating azimuth thrusters. The design has two axes of symmetry, including weight distribution and thruster placement.

The approach of dividing a vessel control system via the Guidance, Navigation and Control categorization is used to distinguish between different system processes, yet the control for this particular system does refer to that of a single vessel, but rather to a set of vessels, which sometimes are controlled individually (when a module moves alone), and in other times together (in assembled platform). Hence, due to shapeshifting, not only the system behavior changes (such as inertia of a platform of variable shape and size), but also the control structure topology. A layered control structure has been developed, operating on three levels; fleet (guidance), platform (control) & module (actuators & physical layer). The fleet control layer represents a guidance system, responsible for high level task planning and providing motion control objectives. These objectives are realized by the platform-motion-control layer in centralized fashion, where each connected structure has control decisions made by this single entity. Generated actuator commands are then sent to all modules of a platform to be executed.

The guidance layer is implemented rather simplistic as setpoint regulation, where assembly and motion control objectives are provided through scheduling and operator input to provides sufficiently varying tasks for system behavior evaluation. Platform motion control generates control effort with three parralel proportional integral & derivative (PID) controllers that scale output on approximate platform model parameters to remain in control of arbitrary configured structures. The platform motion controlling agent allocates the desired virtual control effort between modules such that is realized as actual resultant forces and torque on the structure.

The experimental setup is designed to operate in the towing tank facility of section Maritime and Transport Technology (section MTT) in the faculty of Mechanical, Maritime and Materials Engineering (Faculty 3ME). One of such tanks is equipped with an optical sensing and interpretation system from the brand Optitrack to provide module state estimation. Communication between entities is facilitated by through topicwise subscriber-publishing protocol of the Robotic Operating System (ROS).

Experimental evaluation

Developed system performance evaluation is done in two steps, namely in terms of assembly and motion control behavior.

Performance of reconfiguration is quantified by evaluating the change in relative pose between neighbouring modules in all considered degrees of freedom. Measured signals showing relative module positions show plateauing behavior indicating hull contact and succesful assembly if it occurs in all degrees of motion. Such sudden stops of motion in all dimentions were found and it was shown how succesful assembly can be found numerically.

Motion control performance is evaluated for two platform configurations; a single vessel scenario and a 3x1 assembled platform. Results of system behavior are presented and performance is quantified. Tests consisted of step inputs in the reference signal on all degrees of motion independently. All motions were an-

alyzed by expressing risetime, settlingtime and overshoot as key performance indicators and compared with design criteria. It is concluded that the proposed multi-vessel control system is able to perform automated reconfiguration and collaborative motion control, as behavior proved in line with design goals.

The developed framework has certain limitations in it's current form of implementation that require further development, both in terms of research and commercial implementation. Yet the overall emergent behavior of the automated multi-vessel system is positively received as a small step towards improving mankind's logistical tools.

Contents

Nomenclature

Symbols

$\mathbf{p}_j^i$	$= [x, y, z]^\top$	Position of point j expressed in coordinate system $\{i\}$. If j is a coordinate system then it refers to its origin.
Θ_{ij}	$= [\phi, \theta, \psi]^\top$	Euler angles between coordinate systems $\{i\}$ and $\{j\}$
$\mathbf{v}_{i/j}^k$	$= [u, v, w]^\top$	Linear velocity of point i with respect to j expressed in coordinate system $\{k\}$
$\omega_{i/j}^k$	$= [p, q, r]^\top$	Angular velocity of object i with respect to j expressed in coordinate system $\{k\}$
f_j^i	$= [f_x, f_y, f_z]^\top$	Force with line of action through point j expressed in coordinate system $\{i\}$
m_j^i	$= [m_x, m_y, m_z]^\top$	Moment about point j expressed in coordinate system $\{i\}$
η	$= [\mathbf{p}, \Theta]^\top$	Generalized position
ν	$= [\mathbf{v}, \omega]^\top$	Generalized velocity
τ	$= [f, m]^\top$	Generalized forces
$\mathbf{R}_i^j$		Rotation matrix to rotate a vector from coordinate system $\{i\}$ to $\{j\}$
$\mathbf{M}$		Matrix representing inertial terms of a body
$\mathbf{M}_{RB}$		Rigid body mass
$\mathbf{M}_A$		Hydrodynamic added mass
$\mathbf{C}$		Matrix representing Coriolis & centripetal forces
$\mathbf{D}$		Matrix representing dampening forces
$\mathbf{S}$		Skew symmetric cross product operator
$\mathbf{o}_i$		Origin of coordinate system $\{i\}$
t		Time
$\mathbf{C}_f, \mathbf{C}_m$		Contribution factor of control effort allocation to a thruster
$\mathbf{H}$		State space coordinate system transformation matrix representing translation
$\mathbf{J}$		State space coordinate system transformation matrix representing rotation

Sub- & Superscripts

$\mathbf{p}_{i/j}^k$	Example parameter p of object $\{i\}$ with respect to object $\{j\}$, expressed in coordinate system $\{k\}$.
$\{n\}$	Earth fixed coordinate system, also referenced to as 'global', and commonly assumed inertial
$\{bi\}$	Body fixed coordinate system of object $\{i\}$
$\{pi\}$	Platform (or assembly) fixed coordinate system $\{i\}$
$\{CG\}$	Object or reference is the Centre of Gravity $\{i\}$

Acronyms

CG	Centre of Gravity
GNC	Guidance Navigation & Control (system categorization)
IMO	International Maritime Organisation
MRR	Modular Reconfigurable Robot
MTT	Department of Maritime & Transport Technology, Technical University Delft
PID	Proportional Integral & Derivative (controller)
RAS	Researchlab Autonomous Shipping, Delft
ROS	Robotic Operating System

1 Introduction

Shipping enterprises are under constant pressure to perform faster and more flexible transportation with lower costs and emissions. This incentive, together with advancements such as in communication and network technologies, have led to various projects that explore vessel control automation in the marine sector as a solution.

Machines started to take over relatively simple control tasks of humans when heading control of merchant ships introduced automation in the early 1920's. Nowadays, auto-pilot systems are commonly applied to follow courses as set out in a voyage plan to relieve crew of a burdening task, as well as increase fuel efficiency. These systems are of temporal nature, and are applied in open areas where there is little interaction between vessel and other objects, while humans control vessels in narrow or busy waters, such as ports or canals.

In the past decades, many projects worked to further explore automating various parts of vessel control [38]. Further automation of ship control tasks can broaden viable applicable scenarios, reduce the need of human operators, increase transport system effectivity, reliability and safety [11], while consuming less resources. Realistically, automation will rarely improve all the named benefits at once. Projects tend to aim to enhance several of these measures of performance that are most relevant for a use case, while allowing acceptable losses elsewhere.

Commercially applied vessel systems that are mainly controlled by a machine are scarce, although various research projects recognise potentials of further vessel control automation. Construction of the container sized ship, Yara Birkeland, [2] is a prime example of such projects that aim to redesign commercial transport systems to furher extend control automation. As technical feasibility increases, regulatory frameworks come under discussion to be adapted. A prime example is the International Maritime Organisation developing terminology and definitions suitable for more advanced shipping automation and autonomy [19].

Next to individual operation, interest is given to behavior of a fleet of automated vessels. Collaboration, negotiation and information sharing allow a wider variety of solutions to logistic challenges. In particular, formation and control of vessel platforms, built from a number of smaller connected vessels, has been an area of research. Such vessel platform systems could in some cases be applied for quick, flexible and affordable solutions to logistic challenges where no other vessel solution is feasible for that task, environment or timeframe. For example, temporary bridges could be formed on demand from a set of machine operated smaller modules, allowing rapid logistical solutions for crossing a river. Another use could be a transportation job on a site that is unreachable by a sufficiently large vessel. Both examples could then further benefit from automation of the fleet's control systems, to reduce the need of humans present to benefit safety, applicability and cost.

Automated vessel platform assembly has been explored in various reasearches [36] [40] [43] , to allow formation without the need of a large team of vessel operators.

Assembled vessels together form a platform, which can, depending on the stiffness of connections, be considered a single body. Collaboration between vessels at platform level can improve dynamic response of the connected structure. Controlling a vessel platform that can have a different size and shape in an effective manner is a challenge which has been discussed in [42] [31]. As two or more vessels connect into a platform, accurate modeling, controlling and coordinating become more complex. If the amount of configurations gets significantly high, gathering model parameters for every configuration becomes impossible, while also platform control effort needs to be allocated onto a large set of actuators.

Consider the two systems earlier described, which can further be adressed as A and B and are illustrated in figure 1.1 and 1.2 respectively:

- System A) An automated modular multi vessel system that performs self assembly.
- System B) An assembled vessel platform where it's actuators are controlled by using information of the configuration.

Benefits of both systems are desirable within one framework, as combining these systems into one allows engineers to utilize benefits from both.

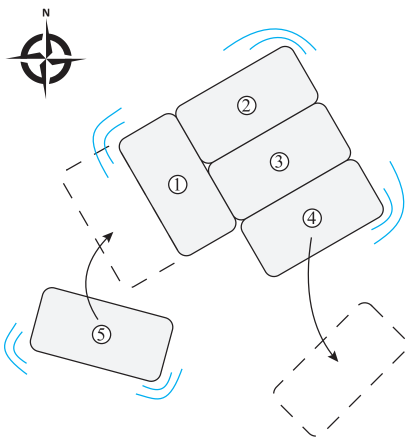


Figure 1.1: Behavior A: Automatic reconfiguration of a vessel platform. Vessel 1,2,3 & 4 are connected such that they form a platform. Arrows indicate movement of vessel 5 to connect, and disconnecting of vessel 4.

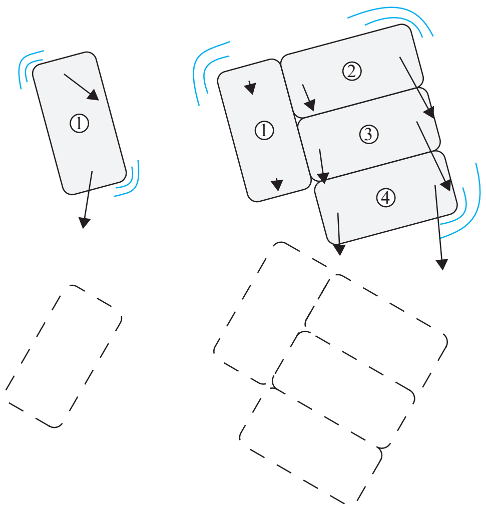


Figure 1.2: Behavior B: Fleet control that adapts on platform configuration. The left vessel is configured in the simplest way, and operates alone. Four vessels on the right collaborate to move the connected vessel platform. Arrows illustrate forces generated by thrusters.

1.1. Problem statement

It would be useful for developers to know what system requirements, characteristics and constraints emerge from implementing a framework with behavior of system A and B. This information can be used to make well informed, effective, scalable, interoperable and long term beneficial design choices.

Literature on both systems is available, but there is a lack of content that describes the effect of combining both systems into one, which is the gap of knowledge that this thesis will adress. The research objective of this work is to develop, implement and evaluate a fleet control system performing self assembly and configuration adaptive control within a single framework.

1.2. Approach

To move towards realizing implementation of automated vessel platforming systems, this paper strives to fill the gap of knowledge on implementation of automated assembling vessel platforms with configuration adaptive control strategies by seeking to answer the following questions:

Main research question

- How can a fleet of modular surface platform vessels be controlled to achieve automated assembly and configuration-dependent platform control?

Sub-questions

1. What is the state of the art within automated vessel platforming systems?
2. What characteristics does a vessel system have that integrates automated assembly with configuration dependent control?

3. How can the dynamics of the multi-vessel system be represented?

4. How can a fleet control framework be developed that performs automated platform assembly and configuration adaptive platform control?

5. What is the performance of the developed system?

Subquestion 1 will be answered by means of literature research. This will cover definitions used in the marine control sector, literature related to automated vessel platform assembly systems, and literature related to configuration adaptive platform controllers. The focus is on marine applications, altough general robotic sources are also considered.

Subquestion 2 will be answered by analyzing fundamentals of system A and B. Information from literature on both system A and B will be used with a systematic approach to map and explain fundamental system characteristics (e.g., functions, requirements, limitations, indicators of performance) of A and B. Thereafter, predictions will be done how a integration of A and B result in a set of characteristics due to inheritance and interference. Attention is given to scoping down the full range of design choices to a region that is considered to have more potential to be realistically implemented in the near future.

Subquestion 3 is answered by formulating a multi-vessel state description, formulating a platform-state description and proposing an approach to predict dynamics of a multi-vessel structure from individual module models.

Subquestion 4 will be answered by developing and implementing a model scale system performing the mentioned behavior. The design is based on formulating a system that is predicted to be most likely implemented in the future from a commercial point of view. Emphasizing advantages of modular vessel platforms will be prioritized, while some increases of disadvantages are deemed acceptable, as modular vessel structures are expected to be only feasible in scenarios where a specific set of performance indicators are prioritized far above others.

Subquestion 5 will be answered by evaluating performance of the developed system. This is done according to performance indicators that resulted from answering subquestion 2. Behavior that does not directly impact performance will also be presented if it is relevant to sketch implementational challenges for future marine automation projects.

Considerations and findings during the whole design process will be documented, to use for future projects, and to formulate into recommendations and discussion on suitable control system designs.

1.3. Scope

The control system for the model scale lab setting will be developed to make a certain configured platform for transporting a large object. For this, three Delfia-1* model vessels (see fig 1.3) from the Reasearchlab Autonomous Shipping (RAS) Delft are used as experimental model scale vessels, which are expected to automatically assemble into a single rigid body and perform several maneuvering tasks in a single assembled configuration. The control system will not be reliant on data from parameter estimation efforts, as this is also not what commercially implemented systems are expected to have. System dynamics are simplified to three degrees of motion in on the surface (x,y, yaw).

This project is limited to finding one solution to the design challenge of the experimental setup. This research will include results gathered during the design process, together with system behavior data of the final framework implemented with the desired behavior A and B.

All design choices of the final framework will be discussed. Choices of these design choices are affected by the available timeframe, but are required to be able to result into a framework with behavior A and B.

The performance indicators that are deemed most relevant for systems developed in this thesis are: Vessel response, reliability, modularity. Design choices in the implementation of the vessel system will be based on these indicators.

Assembling systems can work with or without using a target configuration. Self assembly without target configuration can be such as organic growth, or flocking. This paper focusses on systems that generate a target configuration or uses a given desired configuration, that both use a concept of a desired configuration.

Self-assembly is a form of a more general behavior, self-reconfiguration, which also includes changing of shape and disassembly. This work focusses mainly on the assembly aspect.

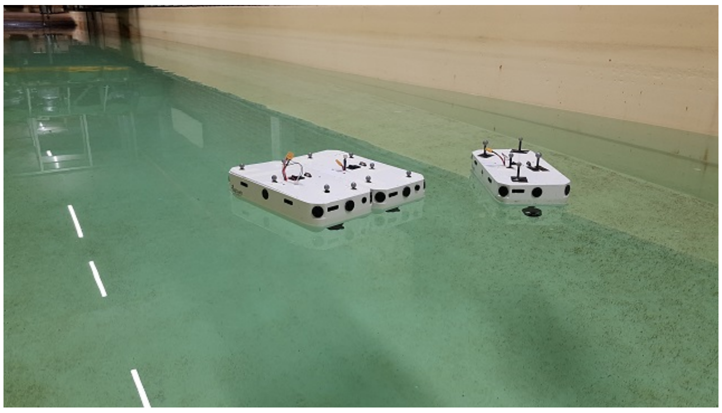


Figure 1.3: Three Delfia-1* modules from RAS Delft in the MTT towing tank facility at 3ME

1.4. Thesis outline

The content of this report is as follows: Chapter 2 answers subquestion 1 by means of literature research on vessel platforming systems and related topics. Chapter 3 shows system analysis of behavior A & B independently, to then predict how functions, requirements, constraints and other characteristics translate and interact in a single system that shows both behaviors which answers subquestion 2. Chapter 4 describes the proposed multi-vessel and platform state description and a novel approach to predicting dynamics of combined waterborne structures from module models. Chapter 5 shows the process of designing an implementation of such a system for model scale vessels, to answer subquestion 4. Design choices and considerations are discussed from a high level (a nework) to a low level (behavior of agents within the network) view. Chapter 6 aims to analyse performance of the developed framework, by evaluating behavioural responses, thus answering subquestion 5. Chapter 7 concludes and discusses the findings of this thesis, and finalizes with future recommendations and vision.

Furthermore appendices include (A)a technical paper that summarizes the work in this project, (B) derivation of the approach to approximate platform dynamics and C sourcecode of the main algorythms that make up the implemented control software.

2 Literature survey on modular vessel platform automation

This chapter summarizes literature relevant to automated vessel platforming systems to answer subquestion 1:

- What is the state of the art within automated vessel platforming systems?

For effective discussion on ship automation good definitions of terminology are required to avoid misinterpretation. Definitions that are used in this work are described in section 2.1, supplemented with various other common interpretations.

This literature review shows various works on automated control of vessel platforming systems, yet it was noticed that the majority of relevant works focus on either the "assembly process" or "control of a platform with adaptable configuration", so they will be discussed in those groups in section 2.2 and 2.3 respectively.

Chen et al. [9] provides an overview on cooperative control methods for waterborne transport where vessel-to-vessel cooperation is classified in three categories; formation control, cooperative collision avoidance, and cooperative manipulation. Vessel platform motion control would fall into the category of cooperative manipulation (a fleet of vessels coordinate their actions to fulfill certain tasks[9]). Platform self assembly has facets overlapping with formation control (steering a group of vessels to form a specific geometric configuration and control their coordinated collective motion [9]) and cooperative manipulation of the multi-robot fleet. As sources that specifically describe modular vessel platforming systems are limited, so a broader range of works is discussed in this review, such as fields of cooperative manipulation, formation control, and reconfiguration automation from general robotic science.

Discussed projects are schematically shown in section 2.4, where this chapter also elaborates on the resulting gap of knowledge that drives this paper. Various works that are introduced and discussed in this chapter are further used in chapter 3 to analyze system characteristics and discuss design choices and considerations of these projects.

2.1. Terminology

Definitions used to describe automation of various vessel processes differ a lot troughout the marine sector. Relevant terminology and their interpretation will be discussed here.

- **Automation** refers to the full or partial replacement of a function previously carried out by the human operator. This implies that automation is not all or none, but can vary across a continuum of levels, from the lowest level of fully manual performance to the highest level of full automation (definition according to Parasuraman et al. [41]).

- **Control** is purposeful action on or in a process to meet specified objectives, which can be exercised by a human or by automation (defnition according to [19]). Control, pertaining to a ship, often primarily refers to the motion of the hull, altough other tasks (such as connecting in this paper) can refer to this as well.

- **Modular reconfigurable robot** (MRR) systems are made up of many repeated modules (or units) that can be rearranged or can rearrange themselves into different configurations depending on the task the robot is to solve at the time (definition according to Seo et al. [45])
- **Configuration** in reconfigurable robotics is commonly generalized to incorporate the connectivity of the modules (which module is connected to which, represented as an adjacency matrix, a linked list, and the like) into the conventional robotics definition of the term that refers to just the pose of the robot (the full set of the joint angles of the robot) (definition according to Seo et al. [45]).
- **Reconfiguration** is the process of changing connectivity Seo et al. [45], which may include **assembly** and/or **disassembly**.
- **Vessel-platform**, or sometimes referred to as platform, is a combination of vessels, or **modules** that are connected to form a floating structure.
- A **cybernetic** system is self-regulating. The term cybernetics was developed by Norbert Wiener in 1948 Wiener [54]. This regulation is done by the system according to a given (control) law. Smithers [47] describes the difference between cybernetics and automation: " Cybernetic systems are thus self-regulating systems, systems that are not just able to move or to act by themselves but are also able to regulate and control their movements or actions so as to maintain their effectiveness in the face of disturbances and perturbations, according to some predefined control law or rule of regulation"
- An **autonomous** system is considered self-governing, thus capable of constructing its own laws, according to Smithers [47]. In practice the interpretation of "autonomous ship" differs among organisations, generally ranging from being interchangably used with "automated" or "cybernetic". Smithers [47] argues that the beforementioned definition is more consistent to use in other sciences while it also functions as an independant important concept; "Autonomous systems are different from cybernetic systems, and they form a more general class of self-regulating systems, those that form their own laws of regulation as well as regulate their behavior with respect to these self-made laws." Indiscriminate use of the term "autonomous" can cause confusion, misinterpretation and false expectations. For further use in this paper, the term autonomous is avoided as it is deemed too ambiguous at this point and other terminology (cybernetics and automation) suffice.
- An **unmanned** surface vessel (USV) is a ship with no humans on board [19]. According to this definition being unmanned does not characterize the nature of the controller. The vessel can be controlled by a human, via remote control, or it can be machine controlled to meet its goals. 'Reduction of human presence' and 'reduction of human control' often go hand in hand [27] but are not equal as can be seen from the example of remote-, human-controlled systems.
- A ship's **power source** provides energy to fuel its actuators. There exists a trend in developing electric powered vehicles. Electrification of vehicle's power source is often done with climate influencing emisions in mind. Vehicle power source, whether fueled by fossil fuels, electric or by other means has no direct influence on the degree of automation. Goals of automation and electrification often overlap, and there are some shared benefits, but they are distinct concepts.

For further reading on definitions, interpretation, categorization and terminology, the following sources are recommended; Vagia et al. [50] discusses the evolution of use of 'levels of autonomy' and 'levels of automation' in robotics. It is shown how authors use terminologies and taxonomies in different ways. Smithers [47]'s describes common use of terms "cybernetics", "automation" and "autonomy", and discusses ways of interpretation.

The International Maritime Organisation (IMO) shows ongoing changes to the proposed definition of 'autonomy' in the contexts of shipping for Standardization [19] in an effort to standardize usage. The last proposed definition of autonomy appears to mean something different than how it is used in other sciences such as philosophy, biology and psychology. How the marine sector started to adopt the word 'autonomy' and how the use will develop is unclear, but it will need a concrete definition of an overarching organisation such as IMO or ISO to avoid discussion.

Use of the terminology: "unmanned surface vehicle" (USV) does not always capture the essence of projects that strive towards automation. The term USV could also contain vehicles that are human controlled from a distance by means of remote control. For control purposes it might be more suitable to categorize according to the nature of the controller (human controlled, machine controlled or hybrid), instead of whether there are humans on board or not.

It is found that when discussing "vessel automation", various topics can stay especially vague. The following questions are considered key to get insights on a so called 'automated vessel system':

- **What functions are exactly automated?** Generally the main function of a vessel is considered movement, thus the motion control is often seen as the main function. Other tasks such as power management (e.g., fueling), system health monitoring, maintenance or on-board activities could also be implied to all be automatic on an 'automated ship', although they are often not. Environmental awareness is likely neccesary for vessel automation on busy waterways, as the ship needs to take other entities into account to safely perform its voyage, although not a requirement for basic motion control.

- **To what degree are these functions automated?** One automated vessel might be to only able to control its position to a reference given by an operator, while more advanced systems equipped with a planning and guidance layer might be able to plan and control a trajectory through dynamic waterways. Various categorizations of automation and autonomy exist, each with their own purpose and complexity, of which many are compared in Vagia et al. [50], for the interested reader.

For a history lesson and some fundamentals about vessel control theory, consider reviewing Clarke [10]. For a regulatory flavour of challenges for vessel automation consider consulting Komianos [34] as the author examines operational, regulatory and quality assurance challenges due to development and deployment of autonomous vessels. Many projects that work on automated, unmanned or autonomous vessels are summarized, of which most are specifically aiming at deployment at sea. Existing regulations where adaptation is deemed required to allow further ship automation are being discussed. Specific comments about required regulatory changes are discussed on: the Convention of life at sea (SOLAS), the International Convention on Standards of Training, Certification and Watchkeeping for Seafarers (STCW), the Convention on the International Regulations for Preventing Collisions at Sea (COLREG), the International Convention for the Prevention of Pollution from Ships (MARPOL), the International Convention on Maritime Search and Rescue (SAR). Various interpretations of autonomy and resulting characteristics are quoted, although no single definition is stated. Interpretation ranges from "unmanned" to "machine controlled".

2.2. Automated reconfigurating vessel systems

Two projects have been identified to build up the majority of the literature on automation of fleet assembly into floating structures. The first discussed works are on a project that developed a nameless fleet of container-like modules with rope and hook connectors, while the second works are developed in the "Roboat" project with vessels of similar naming.

O'hara et al. [40] and Paulos et al. [43] record developments of their fleet of container-like modules. The vessels are rectangles with dimentions of the same ration of standard shipping containers, but at a model scale (Fig. 2.1). The specific shape is aimed to demonstrate how an upscaled system could be rapidly deployed from standard container carriers.

O'hara et al. [40] greatly illustrates potential of modular marine structures to solve logistical challenges by forming a bridge that allowed crossing of a remote controlled toy car (figure 2.2). The structure formed consists of partially functioning modules in between. Which functions were active on these modules is not mentioned. It is expected that connection systems were functional on all vessels, while other functions, such as positioning, were active on a part of the fleet.

The Roboat project aims to develop a floating combined infrastructure from modules which are also reffered to as 'Roboats', where the main collaborators are Massachusetts Institute of Technology (MIT) and the Amsterdam institute for Advanced Metropolitan Solutions (AMS). Wang et al. [51] initially presents the Roboat system equipped with a real-time nonlinear model predictive control system. The vessel design has four thrusters configured in a cross to produce holonomic motions. The design is based to form a framework upon which further tests can be performed for transportation and self-assembly to floating infrastructures. This paper is the first published document on development of the "Roboat" concept, while the project aims to scale up from model vessels to full scale operation [18]. Early scale up is shown in Wang et al. [52] to a 1.0x2m version at solo operation. All following resources that use Roboat equipment are however based on the smaller models.

Mateos et al. [36] shows development of a latching system consisting of male/female ball/cone components. This paper discusses latching hardware and shows experimental results of a vessel system that per-

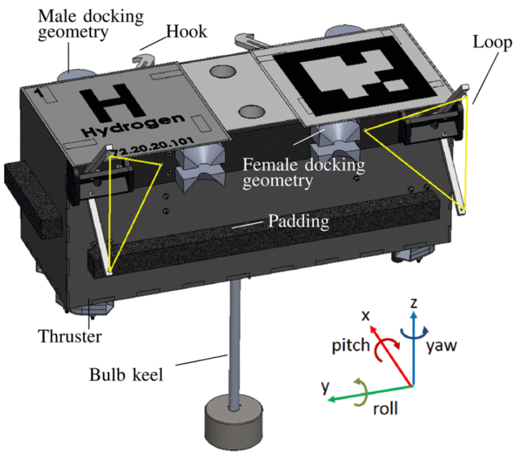


Figure 2.1: Container-like platform module used in [40] and [43]

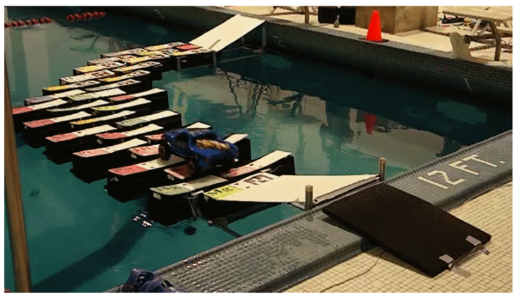


Figure 2.2: Static floating structure formed from 33 modules [40]

Figure 2.3: Roboat's concept of creating floating structures from square modules Wang et al. [51].

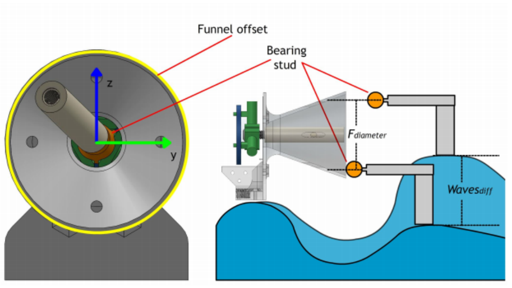


Figure 2.4: Mateos et al. [36] shows a ball and funnel connection mechanism tested on Roboat vessels.

forms platform assembly with the implemented latching hardware.

Gheneti et al. [23] and Kelly [32] present trajectory planning algorythms for reconfiguration of modular surface structure components. The proposed core logic of the shapeshifting algorythm is in finding the largest overlap between current and desired configuration. Overall phases of the proposed reconfiguration scheduler are shown in figure 2.5. Gheneti et al. [23] show approaches and experimental results of platform shapeshifting, by distincting the whole reconfiguration problem into task planning, trajectory planning and trajectory tracking (see Fig. 2.6).

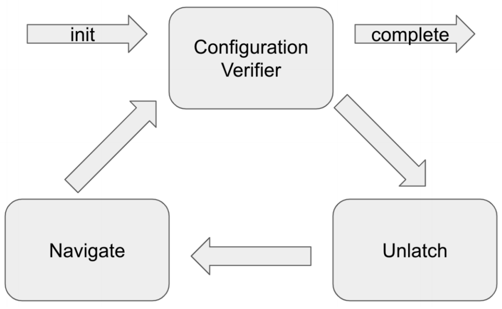


Figure 2.5: Platform reconfiguration system phases from Kelly [32]

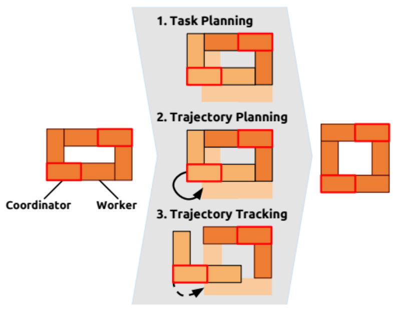


Figure 2.6: Shapeshifting process of Gheneti et al. [23] in Three Stages: 1) Task Planning finds a decomposition for unlatching into two assemblies and relatching them in a desired shape. 2) Trajectory Planning computes a trajectory for one assembly from one latching point to another one. 3) Trajectory Tracking controls the assembly along the trajectory.

The literature on non-maritime multi robot assembly proved to be more expansive, thus readers are encouraged to take a further look at this if they are interested to get inspired from a broader body of literature. Two of such works on assembly of general robotic systems are as follows.

Tuci et al. [49] describes key literature on assembly of multi-robot systems. The author noticed the pattern in studies that gave reason to distinct three physical connection strategies in multi-agent systems: Pushing only, grasping, and caging. This reseach had no focus on vessels or orher marine applications, but describes many great works from a broader, general robotics body of literature.

Miyashita et al. [37] shows the relation between morphology on the success of assembly in the eventually desired shape of stochastic robots. This work took inspiration from biological examples of self-assembly to design and build a water-based modular robotic system consisting of plastic tiles capable of aggregation on the water surface. An externally applied electric potential controlled the self-assembly of the aggregates. This paper is particularly interesting as it shows a completely different approach (stochastic vs deterministic) and operational scale than the container like modules and the Roboat project, illustrating the vast amount of possible solutions to self-assembly challenges.

2.3. Collaborative motion control

As arbitrary shaped waterborne structures are formed, it can be beneficial to be able to manipulate motion of the combined body. This section discusses works that focus on cooperative manipulation of waterborne objects. Modular strucures are of particular interest, although works describing motion control of such systems are limited. The majority of the encountered literature focusses on controlling motion of a single large object (e.g. a large unactuated barge) by utilizing combined effort of a fleet of automated vessels (e.g. tugboats).

Johansen and Fossen [30] distincts three levels of hierarchy for motion control of over-actuated systems, as shown in Fig. 2.7. A high level motion control system generates virtual control efforts to meet control objectives. A second system allocates the desired virtual control effort between actuators, possibly taking into account issues such as actuator saturation and meeting secondary objectives such as power efficiency optimization. A third layer operates on the level of a single actuator such that it meets allocated state (e.g. propeller speed, actuator orientation). Johansen and Fossen [30] provides a varied survey on control allocation algorithms from aerospace, maritime, automotive and mechatronic industries. Algorithms are classified in two main classes based on using linear or nonlinear models. Johansen and Fossen [30] mentions that the

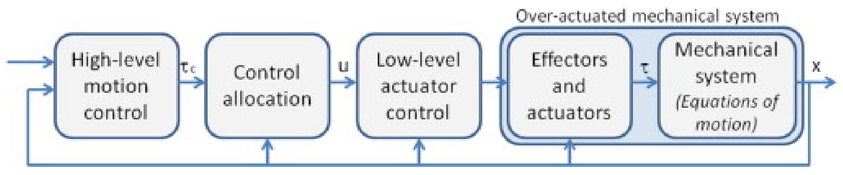


Figure 2.7: Johansen and Fossen [30] Control system structure including control allocation. The vector denotes commanded virtual control effort (generalized forces), while are the actual allocated control effort.

design on the control allocation algorithm and the high-level motion control algorithm cannot always be independent, and lack of feasibility of the control allocation should be observed and handled by the high-level motion control algorithm in order to avoid unacceptable degradation of performance in such cases. This division of the control system generally pertains a more common single vessel scenario, yet these principles are applied throughout the majority of multi-vessel scenarios as well.

Various works describe different approaches for controlling motion of a larger vessel using a fleet of smaller modules (see Feemster et al. [16],Braganza et al. [6], [17], Smith et al. [46], Bidikli et al. [4], Esposito et al. [14], Du et al. [12], and more). A generic usecase is motion of an uncontrolled ship with a fleet of small tugboats, as shown in Fig 2.8. These works bear significant similarities with motion control of a modular platform, as both challenges are a form of collaborative manipulation. However, they differ as these works assume constant shape and dynamics of the controlled vessel. This often allows to reasonably assume that ship dynamics are known, which cannot easily be said for modular structures. A selection of these works that were found to illustrate a specific challenge or approach are briefly discussed.

Feemster et al. [16] considers the control problem of cooperative manipulation of a larger object by a team of smaller robotic vessels using a decentralized architecture. This work focusses on controlling only rotation, yet shows approaches and considerations to avoid a single point of failure from a centralized architecture. Other works also explore distributed control approaches, such as Braganza et al. [6],Habibi et al. [26] and Chen et al. [8].

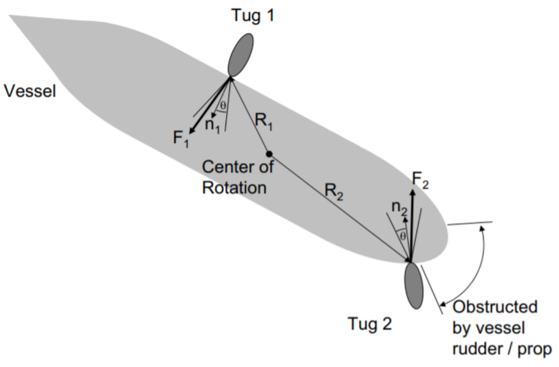


Figure 2.8: Illustration of swarm forces on disabled vessel (Feemster et al. [16])

Feemster and Esposito [17] presents a trajectory tracking framework of cooperative ship manipulation. Primary challenges are identified and described as: (1) the actuators are unidirectional and experience saturation; (2) the hydrodynamics of the system are difficult to characterize; and (3) obtaining acceptable performance under field conditions. An experimental setup shows a proposed control approach with separate trajectory generation, tracking control, and force allocation taking into account saturation of actuators. The controller employs an adaptive feedback law to compensate for unknown—difficult to measure—hydrodynamic parameters, employing a 3-DOF planar second order dynamical model.

Various connection mechanisms are described in existing literature, each with different characteristics. Smith et al. [46] investigates utilization of a swarm of automated tug boats to manipulate an object, where the tugboats are constrained to only exert a pushing force. Chen et al. [8] sheds light on contact dynamics in a cooperative manipulation challenge where tugboats can only exert pulling force, while also taking into account connector dynamics of an elastic towing line.

Bidikli et al. [4] proposes an approach to control a large object with pushing tugboats, while his framework supports changing configuration as a novelty. The tugboats exert forces on the barge, and are treated as actuators of the large ship with time varying thruster-configuration. The controlled object is assumed under influence of hydrodynamic mass effects and a self-tuning control gain strategy is employed. Efficiency of the presented controller is demonstrated trough simulation.

Du et al. [12] describes such a multi-vessel control architecture where tugboats are designed to satisfy allocated towing forces and angles. This formalized layers illustrate how tugboats effectively function as acuators with their own specific dynamics. This work is extended in Du et al. [13] to incorporate environmental disturbances.

Contrary to previously mentioned works, Park et al. [42] describes an approach of controlling a modular structure. The vessels that combine into the combined body are by themselves already fully actuated. The main control approach used is a PD controller to generate control effort, which is said be applicable to any configuration and reference. Allocating control effort between thrusters is implemented as an optimizer minimizing energy use to find actuator response. Experimental results show effectiveness of the proposed control strategy on three different configurations. The control system design is based on an approximate model, which scales according to a variable amount of vessels that are coneccted into the platform, but not to configuration shape. Translational inertia (directional dependent masses in x and y direction) is assumed to scale with n amount of vessels, while rotational inertia was scaled quadratic by n^2. System inertia is represented by constant directional dependent mass as the sum of rigid body and hydrodynamic-added mass.

Scenarios where smaller vessels manipulate a significantly larger object, dynamics of the manipulators might be neglectible. However, assemblies with comparably sized modules will have rigid body dynamics affected by the configuration. Park et al. [42] distinguishes itself from the previously mentioned works as it attempts to predict dynamics of the assembly based on the configuration, albeit only on numbers and not on shape. Park et al. [42] proposes scaling rules to formulate the approximate platform model, as this is a key challenge that needs to be faced for model based controllers for modular strucures.

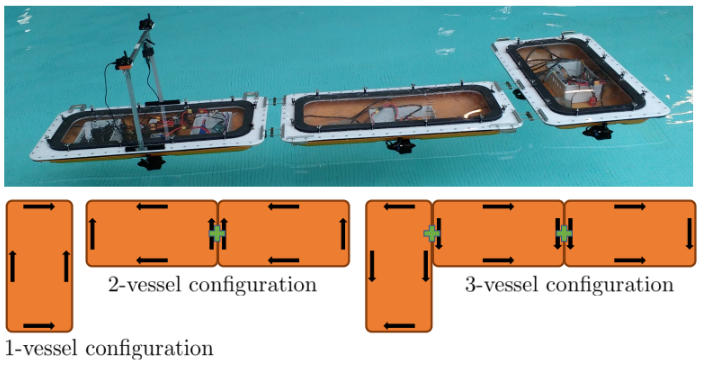


Figure 2.9: (Top) L-shape configuration of the connected-vessel platform and (Bottom) 3 different configurations used in the experiments. Park et al. [42]

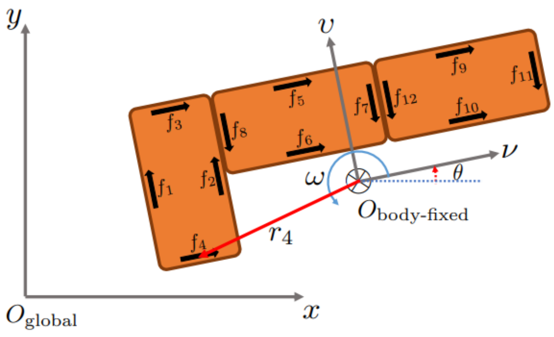


Figure 2.10: An example of a rigid body of three connected vessels, where each black arrow drawn on the body of a vessel represents force exerted from a propeller. Park et al. [42]

2.4. Gap in knowledge

Works on modular vessel platform automation found through this literature survey originated from two project concepts; container shaped modules [40] shown in Fig 2.1 and Roboat Wang et al. [51] shown in Fig 2.3). Contributions using Roboat vessels are credited to Advanced Metropolitan Solutions (AMS), Massachusetts Institute of Technology (MIT) or their collaboration. More literature exists on concepts that share similarities with vessel platform automation, but miss characteristics to classify as such. The majority of this relevant literature describe approaches for controlling a non-modular object using a fleet of smaller automated vessels.

The discussed projects have a variety of works that explore facets on vessel platform automation, also showing many different design choices and approaches. Projects regarding automation of a multi-robot system performing reconfiguration or configuration dependent control are shown in table 2.1 to aid comparison. Works on multi-robot assembly and collaboration from aerospace and automotive branches have also been included in this schematic to give some perspective on developments in other fields.

From this literature research it was concluded that no information is found of an modular fleet system that incorporates automated reconfiguration with collaborative configuration dependent control strategies in a single framework. Table 2.1 shows that works focus on either reconfiguration or collaborative control, not on combining the two, thus consequences of integrating the two behaviors in a single system are unmapped. Gathering and documenting approaches and experiences on integration can pave the road towards effective implementations to fully benefit from automated structure assembly while motion control of the combined platforms are enhanced by effective collaborative approaches.

Title	Author	Year	Land, naval or aerial	Simulation or experimental	Automated assembly	Changing control system during operation	System is config. dependent	Controlled object is modular
Autonomous self-assembly in swarm-bots	Groß et al. [24]	2006	Land	Both	Yes	No	No	Yes
Object transport by modular robots that self-assemble	Groß et al. [25]	2006	Land	Experiment	Yes	Yes	No	Yes
A novel autonomous self-assembly distributed swarm flying robot	Wei et al. [53]	2013	Aerial + land	Both	Yes	Yes	Yes	Yes
Self-assembly of a swarm of autonomous boats into floating structures	O'hara et al. [40]	2014	Naval	Experiment	Yes	No	No	Yes
Automated Self-Assembly of Large Maritime Structures by a Team of Robotic Boats	Paulos et al. [43]	2015	Naval	Experiment	Yes	No	No	Yes
Cooperative Multi-Vessel Systems for Waterborne Transport	Chen [7]	2019	Naval	Simulation	No	No	No	No
Coordinated Control of a Reconfigurable Multi-Vessel Platform: Robust Control Approach	Park et al. [42]	2019	Naval	Both	No	No	Yes	Yes
Autonomous latching system for robotic boats	Mateos et al. [36]	2019	Naval	Experiment	Yes	No	No	Yes
Trajectory Planning for the Shapeshifting of Autonomous Surface Vessels	Gheneti et al. [23]	2019	Naval	Experiment	Yes	No	No	Yes
Algorithms for planning and executing multi-roboat shapeshifting	Kelly [32]	2019	Naval	Experiment	Yes	No	No	Yes
Manipulation of large objects by swarms of autonomous marine vehicles. Part 1-Rotation	Feemster et al. [16]	2006	Naval	Neither	Noy	No	Yes	No
Positioning of Large Surface Vessels using Multiple Tugboats	Braganza et al. [6]	2007	Naval	Simulation	No	No	Yes	No
Comprehensive framework for tracking control and thrust allocation for a highly overactuated autonomous surface vessel	Feemster and Esposito [17]	2011	Naval	Experiment	No	No	Yes	No
Swarn Manipulation Of An Unactuated Surface Vessel	Smith et al. [46]	2007	Naval	Simulation	No	No	Yes	No
Robust dynamic positioning of surface vessels via multiple unidirectional tugboats	Bidikli et al. [4]	2016	Naval	Simulation	No	No	Yes	No
Cooperative manipulation on the water using a swarm of autonomous tugboats	Esposito et al. [14]	2008	Naval	Experiment	No	No	Yes	No
Distributed model predictive control for cooperative floating object transport with multi-vessel systems	Chen et al. [8]	2019	Naval	Simulation	No	No	Yes	No
Cooperative control of autonomous tugs for ship towing	Du et al. [12]	2020	Naval	Simulation	No	No	Yes	No

Table 2.1: Comparison of various projects related to vessel platform self assembly and/or collaborative platform motion control

3
System Analysis

This chapter formulates characteristics of the two considered systems and their combination, to answer sub-question 2:

- What characteristics does a vessel system have that integrates automated assembly with configuration dependent control?

Answering this question will bring better understanding of the meaning of "a system that performs self-assembly" and "a multi-robot-assembly performing collaborative and coordinated control". The aim is to look beyond the design solutions from the projects described in the previous secion, and create a broader view on the design spectrum.

The first section explores characteristics of systems performing automated assembly (section 3.1) and configuration dependent control (section 3.2). This chapter concludes by mapping system characteristics as a result of integrating both behaviors (section 3.3).

This section uses various external sources to aid formulation of characteristics, some of which have been discussed in the previous chapter. The overall approach is to distinct generalized system characteristics as well as a narrowed down set of solutions that can be considered more feasible to perform in commercial applications, also taking into account design choices of other projects.

3.1. Analysis of first required behavior: Automatic vessel platform reconfiguration

Automated modular vessel platform reconfiguration systems can be categorized as a more general "modular reconfigurable robot" system. Modular reconfigurable vessel systems can be evaluated from a non-maritime perspective, so where we are not directly focussing on vessels, but on more general modular reconfigurable robot (MRR) systems. Seo et al. [45] describes an MRR system as made up of many repeated modules (or units) that can be rearranged- or can rearrange themselves into different configurations depending on the task the robot is to solve at the time. The key characteristic of such systems is described as adaptability of hardware structure to suit a given task or environment. MRR systems differentiate themselves from normal robot systems by determining and executing a course of action, to change it's configuration.

Various requirements of floating MRR systems were identified during the course of the project. Some approaches originating from sources about robotics in general were broader than approaches encountered in sources about modular vessel platforming. This section aims to conclude with two things. Firstly, it searches for fundamental characteristics of an automatic reconfiguring vessel platforming system. Secondly, the whole range of design choices and approaches is scoped down to a set that is expected to have the highest commercial feasibility. This is based on design choices of prior projects, supplemented with the authors argumented vision. During elaboration on system characteristics, prior projects will be discussed, more specifically, how they solved their challenges and in which characteristics did this result in. Automated vessel platform reconfiguration characteristics are discussed in 2 aspects; Strategies and Actuation.

3.1.1. Strategies

Yim et al. [55] describes a taxonomy of architectures of modular robots, which is adopted to the use case of surface vessel platform reconfiguration. MRR system architectures are classified in three generally observed classes. They are described as follows [55]

- **Lattice Reconfiguration Architectures** have units that are arranged and connected in some regular, three-dimensional pattern, resulting in a relatively simple and easily scalabe system.

- **Chain or Tree Architectures** have units that are connected together in a string or tree topology. This chain or tree can fold up to become space filling, but the underlying architecture is serial.

- **Mobile Architectures** have units that use the environment to maneuver around and can either hook up to form complex chains or lattices or form a number of smaller robots that execute coordinated movements and together form a larger "virtual" network.

We can see that O'hara et al. [40] uses a lattice reconfiguration architecture in the horizontal surface plane (figure 3.1). Roboat shows L configurations in various publications, which can be considered a chain architecture, as configurations are used that do not fit in a repeating (lattice) pattern. Roboat reduces the amount of possible configurations by utilizing the square shape of the vessels to reduce the amount of orientations to steps in relative angle of 90 degrees.

Another classification can be found in the way structures are (re)configured. Yim et al. [55] distincts two approaches:

- **Deterministic Reconfiguration** uses modules that are purposely manipulated to a target location.

- **Stochastic Reconfiguration** relies on statistics of a set of modules to configure in a more desirable configuration. System design is aimed to reach acceptable configuration without the need of predefined reconfiguration planning.

Stochastic reconfiguration can be achieved with very limited functionality per module. Due to robot simplicity, this approach can very desirable on micro scale which can be scaled to great numbers. Deterministic reconfiguration needs more functionality embedded in a module. This system requires an agent that makes some form of plan, which is then to be executed. As a plan is formed and executed, this approach does give the opportunity to give more guarantees that a desired configuration is reached within a timespan.

Due to technological advances, communication and computational hardware has become increasingly affordable. If the amount of modules in a system is relatively small, then equipping each individual with communication, computation and control systems is feasible. This allows creation of networks of robots that can share information, collaborate, negotiate and utilize benefits of all sorts of layered control architectures. Both O'hara et al. [40] and Roboat utilize a deterministic approach to reach assembly.

A common approach to solving the task of deterministic reconfiguration is dividing it in several tasks. O'hara et al. [40] describes the following states to reach assembly:

- Generation of desired configuration. O'hara et al. [40] describes the result as a blueprint with a map of relative boat positions.

- Connecting sequence selection. An entity selects a sequence in which the assembly is to happen.

- Positioning of a module. A trajectory of a vessel to an assembly location is generated and realized by actuators.

- Docking Sequence. As a vessel reaches a docking site within an area of acceptance, the docking sequence runs, which should finalize the docking of that vessel.

O'hara et al. [40] greatly illustrates how they formulated strategies for generating connection sequence and module positioning (figure 3.1 and 3.2)

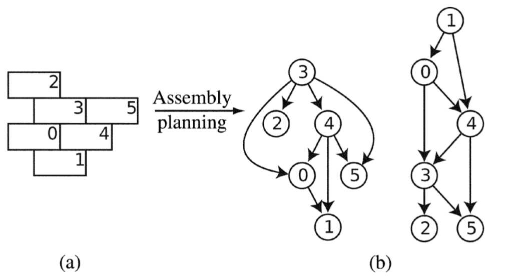


Figure 3.1: The assembly planning stage of O'hara et al. [40], showing desired platform blueprint (a) and two cannidate assembly sequences (b).

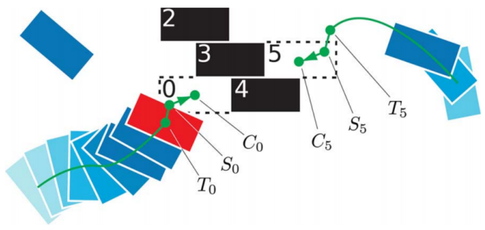


Figure 3.2: The trajectory of assembling modules positioning to continue into a docking sequence [40].

3.1.2. Actuation

Two tasks are identified that must be realized by actuators of the system:

- means of relative repositioning modules during reconfiguration
- means of maintaining acceptable relative position between modules when connected

Altough these tasks can theoretically be done by a single system, Seo et al. [45] notices that MRR modules generally have two actuator systems to adress these tasks; a large main actuator to move itself with respect to other modules, and a smaller actuator that connects or disconnect two modules.

The main actuator of MRR systems can be measured in nondimensional characteristic length, being "the number of modules that can be supported in a cantilever fashion under gravity"[45]. This measure is perhaps useful for some robotic systems, but does not make sense to use for vessel platforming systems, as gravitation is cancelled out by buoyancy and hydrodynamic forces. These forces are defined as Fossen [21]:

- Buoyancy force due to the hydrostatic pressure (proportional to the displacement of the ship).
- Hydrodynamic force due to the hydrodynamic pressure (approximately proportional to the square of the relative speed to the water).

Vessel platforms are not considered excelling at tasks that require large speeds, but rather tasks that benefit from versatility, robustness and low cost[45]. Vessels platforms with relatively low operational speeds can be classified as "Displacement Vessels" Faltinsen [15]

$$Fn = \frac{U}{\sqrt{gL}} < 0.4 \quad \text{for vessels categorized as "Displacement Vessels"} \tag{3.1}$$

The nature of displacement vessels allow horizontal movement of masses (the ship itself and cargo) much higher with respect to robotic systems that lift, which results in a completely different order of magnitude of the ratio between propulsion and inertia. Vessel modules do need means of relocation with respect to other vessels with enough strength to overcome reasonable disturbances, such as wind or current. Further increasing the magnitude of forces that relocate the module reduces the achievable time that a module needs to relocate, which is also dependent on inertia and drag. Strength of main actuator and module mass are affected by various design choices, but optimal choices vary widely per use case. Systems that fulfull the task of moving the vessel in the surface plane come in many forms and configurations. Common actuators are propellers, rudders or fins. It is often considered desirable to have a set of actuators that allow imposing forces on all the three degrees of freedom independently, such that we can consider the module fully actuated. Figure 3.3 and 3.4 show actuator setups of the Roboat and O'hara et al. [40], which can both be considered fully actuated.

A vessel platforming system needs means to maintain its configuration, which can be achieved using physical restraints that limit relative motion between modules for some period of operation. Movement of the platform as a whole can be desired, for instance when a vessel platform is performing a task of moving to a certain point or orientation. Undesired movement can be caused by disturbances. Alhough maintaining configuration of a vessel platform can theoretically be realized by the main actuators of the module, it is often seen that a second type of actuator is applied, specifically designed to reduce unwanted relative motion

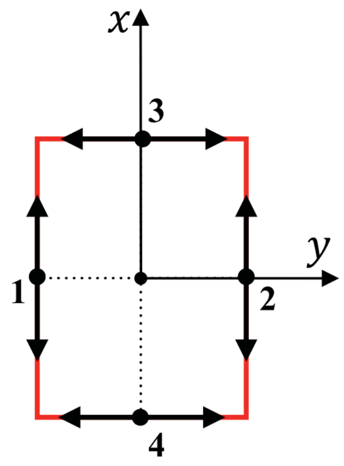


Figure 3.3: The "+" shaped thruster setup of Roboat [51]

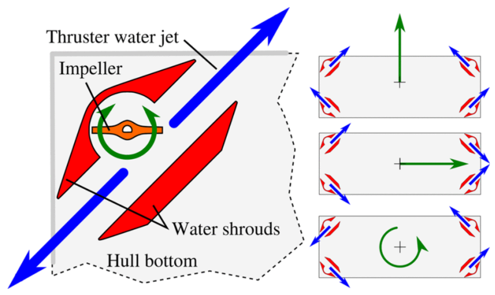


Figure 3.4: The "x" shaped hruster setup of "Tactically expandable maritime platform module" [40]

between modules by means of physical constraint. Generally once such systems are implemented, often by mechanical [36] [40] or magnetic [32] means, the configuration is considered an assembly.

The application of MRR systems on vessel platforms is a niche which generally has specific goals, constraints, environments. The majority of the motion is on the surface plane, often allowing engineers to consider only three degrees of freedom. The environment already enforces some degree of dampening to the system, as various hydrodynamic dampening effects are always present on ships. Robots performing self-reconfiguration is a big topic, on which the most relevant concepts for vessel platforming are elaborated in this section. For further reading on general (not vessel platforming) MRR systems, consider Yim et al. [55] and Seo et al. [45] and references therein.

3.2. Analysis of second required behavior: Configuration adaptive vessel platform control

3.2.1. Problem Description

We have a team of n modules connected into a platform. n can vary through time as modules connect or disconnect. There exists an expression of the state of the platform expressed similar to single vessels as:

$$\eta_p = \begin{bmatrix} \mathbf{p}_p^n \\ \Theta_{np} \end{bmatrix} = \begin{bmatrix} x_p^n \\ y_p^n \\ \Psi_p^n \end{bmatrix} \tag{3.2}$$

$$\nu_p = \begin{bmatrix} \mathbf{v}_{p/n}^p \\ \omega_{p/n}^p \end{bmatrix} = \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{3.3}$$

Where η_p and ν_p describe generalized positions and velocities of the platform coordinate system origin. If the platform connections can be assumed rigid, a platform fixed coordinate system can be made that defines platform state in the above mentioned η_p and ν_p. Other definitions of platform state can be fixed to a single vessel, which does not assume rigid connections, or by coinciding the platform origin with its estimated overall centre of mass. Figure 3.5 illustrates an example of a platform coordinate system fixed to the vessels, where in this definition the origin of the platform is even a little outside the body of the platform.

Each module m_i has $r_i = 0, 1, 2, ...$ actuators that are able to impose forces on that module to move it, such as propellers. The amount of actuators that modules have may not be equal for all vessels. This may be unequal by design (as in a heterogenous fleet) , or as a response to a noticed malfunctioning thruster. The total amount of actuators is the summation of the actuators of all vessels.

$$r = \sum_{i=1}^{n} r_i \tag{3.4}$$

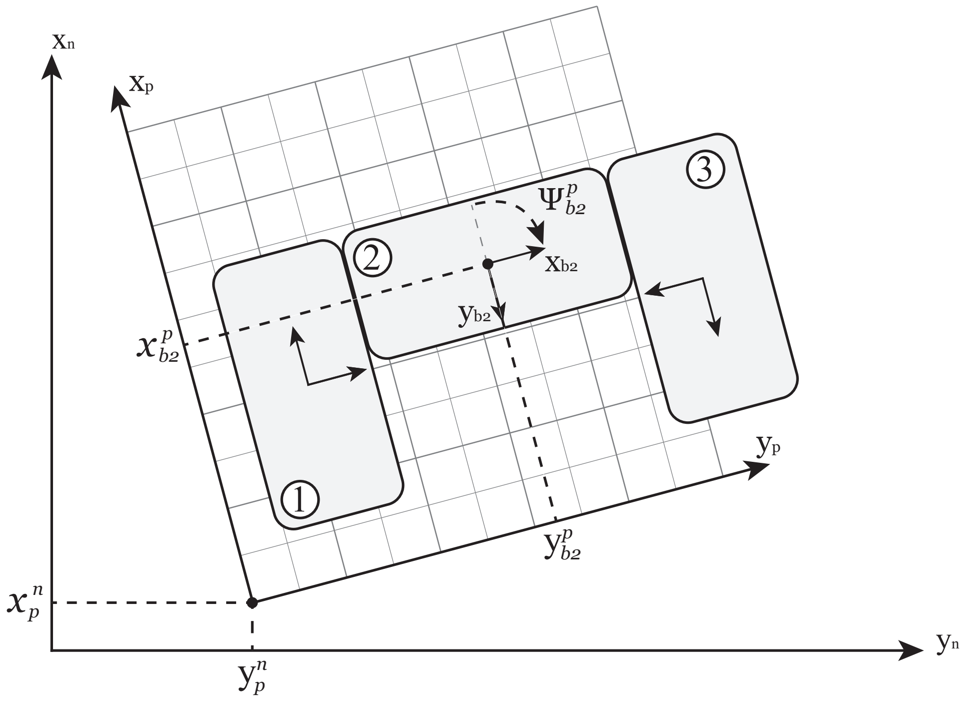


Figure 3.5: A platform consisting of three modules. Expression of pose of vessel 2 in the platform coordinate system ($\{p\}$) is illustrated, as well as position of the platform frame origin in global ($\{n\}$) frame. This example shows the origin of the platform coordinate system outside of the physical body of the platform, but this placement can be chosen elsewhere at the convenience of the designer.

Actuators are bound to various constraints such as direction and maximum force output. Acuators that make forces between vessels to connect them could be described as actuators as well, which can be benificial if connections are not considered rigid. Actuators have dynamics of their own, meaning that they often do not respond instantaneous. However, modelling the dynamics of actuators can overcomplicate a total system model. If a model is to be kept simple, one can assume that execution of actuator reference is done sufficiently accurate and timely. It is a good idea to reflect if this assumption is reasonable by evaluating the performance of used actuators to accurately follow control inputs, and judge whether inacurracy or delay will significantly affect overall system behavior.

3.2.2. Motion control systems

Waterborn vehicle dynamics can be described according to Fossen [21], where the author shows that translational motion of a vessel about CO satisfies

$$m[\dot{\mathbf{v}}^b_{b/n} + \dot{\omega}^b_{b/n} \times \mathbf{r}^b_g + \omega^b_{b/n} \times \mathbf{v}^b_{b/n} + \omega^b_{b/n} \times (\omega^b_{b/n} \times \mathbf{r}^b_g)] = \mathbf{f}^b_b \tag{3.5}$$

and that rotational motion of a vessel about CO satisfies

$$\mathbf{I}_b\dot{\omega}^b_{b/n} + \omega^b_{b/n} \times \mathbf{I}_b\omega^b_{b/n} + m\mathbf{r}^b_g \times (\dot{\mathbf{v}}^b_{b/n} + \omega^b_{b/n} \times \mathbf{v}^b_{b/n}) = \mathbf{m}^b_b \tag{3.6}$$

where components in equation 3.5 and 3.6 are defined as

f^b_b	=	$[X, Y, Z]^\top$	- Force with line of action through point o_b expressed in coordinate system $\{b\}$
m^b_b	=	$[K, M, N]^\top$	- Moment about point o_b expressed in coordinate system $\{b\}$
$\mathbf{v}^b_{b/n}$	=	$[u, v, w]^\top$	- Linear velocity of point o_b with respect to o_n expressed in coordinate system $\{b\}$
$\omega^b_{b/n}$	=	$[p, q, r]^\top$	- Angular velocity of $\{b\}$ with respect to $\{n\}$ expressed in coordinate system $\{b\}$
$\mathbf{r}^b_g$	=	$[x_g, y_g, z_g]^\top$	- Position of centre of gravity expressed in coordinate system $\{b\}$

Fossen [20] expresses equation 3.5 and 3.6 vectorial form

$$\mathbf{M}_{RB}\dot{\nu} + \mathbf{C}_{RB}(\nu)\nu = \tau_{RB} \tag{3.7}$$

where $\mathbf{M}_{RB}$ is the rigid body inertia matrix, $\mathbf{C}_{RB}$ is a matrix of rigid-body Coriolis and centripetal forces due to the rotation of $\{b\}$ about the inertial frame $\{n\}$, ν is a vector with generalized velocities expressed in $\{b\}$ and τ_{RB} is a vector with generalized forces expressed in $\{b\}$.

Dampening forces can be modelled linear with constant D matrix as

$$\mathbf{M}_{RB}\dot{\nu} + \mathbf{C}_{RB}(\nu)\nu + \mathbf{D}\nu = \tau \tag{3.8}$$

Or nonlinear as:

$$\mathbf{M}_{RB}\dot{\nu} + \mathbf{C}_{RB}(\nu)\nu + \mathbf{D}(\nu)\nu = \tau \tag{3.9}$$

Which can both be estimated by model identification experiments. Whether linear or nonlinear dampening models are suitable depends on the accuracy of tests and reasonabe operating range. It is common to model some hydrodynamic forces as constant additive accelleration coefficients. Humphreys and Watkinson [28] states that the forces and moments represented by the acceleration hydrodynamic coefficients can, to a very great extent, be modeled as potential flow phenomena, yielding

$$\mathbf{M}\dot{\nu} + \mathbf{C}(\nu)\nu + \mathbf{D}(\nu)\nu = \tau \tag{3.10}$$

where

$$\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A \tag{3.11}$$

Where $\mathbf{M}_{RB}$ represents inertia of the rigid module and $\mathbf{M}_A$ represents hydrodynamic added mass. Similarly

$$\mathbf{C}(\nu)\nu = \mathbf{C}_{RB}(\nu)\nu + \mathbf{C}_A(\nu)\nu \tag{3.12}$$

Where $\mathbf{C}_{RB}(\nu)\nu$ and $\mathbf{C}_A(\nu)\nu$ represent coriolis and centripetal forces of the rigid body and hydrodynamic added mass.

Control forces of all actuators (of all modules) can be described in a vector format as:

$$\mathbf{f} = [F_1, F_2, F_3, ...F_r]^\top \tag{3.13}$$

Where F_i is the control force generated by actuator r_i. If the relation between control input and resulting force can be assumed linear, the control forces can be expressed as:

$$\mathbf{f} = \mathbf{K} * \mathbf{u} \tag{3.14}$$

Where $\mathbf{u}$ is a vector of control inputs and $\mathbf{K}$ is a diagonal force coefficient matrix (see Fossen [21])

Control forces applied across the body can be expressed in total control forces:

$$\tau = \mathbf{T}(\alpha)\mathbf{f} \tag{3.15}$$

Where $\mathbf{T}(\alpha) \in \mathbb{R}^{n\times r}$ is the thrust configuration matrix, dependent on angles $\alpha = [\alpha_1, ...\alpha_p]^\top \in \mathbb{R}^p$ of p rotatable thrusters. Fossen [21] shows how the use of angles of rotatable thrusters can be avoided in the above expression by decoupling the force of a rotatable thruster in two separate forces (generaly decoupled in x and y for surface vessels). Decoupling x and y components of a rotating thruster can simplify math challenges, but does require to take into account physical constraints of absolute maximum thrust of the actuator.

The team of connected modules has a single, dynamic, multi-robot task, namely the control of actuators to satisfy a time varying platform state. What a satisfactory state actually is depends on the designer, but this is usually defined as a reference state, such as desired positions, and/or velocities. Utilizing the concept of a reference state in a multi-robot collaborative system means that the controlling entity of a module or platform needs means to aquire the control objective. Various design options are possible to have a set of robots collaborating towards achieving a single goal. The control objective can be created by various entities.

- The robot can generate its own control objective, sensing the environment and following a decision protocol.
- The control objective can be generated by another entity, such as another robot or an operator. This would require means of communication.

Parasuraman et al. [41] describes an approach on dividing a control problem in four stages with similarities to human decision making. This approach is general and intended to be applicable to a wide spectrum of robot systems, as shown in figure 3.6, yet can be applied to a multi-robot system as well. The amount of agents in the multi robot system that collaboratively perform functionality of one of the stages can vary. Some tasks can be done by a single agent, while other tasks can be achieved with multiple. For example; sensing fleet state can be done by a single centralized agent, sending sensor data to on board decision makers, or each vessel can have its own means of sensing its surroundings.

A collaborative multi vessel control system can thus have its tasks distributed across agents in many ways. To control n vessels where each vessel is equipped with r_i actuators, the following system components can be identified for a certain design divided as shown in Parasuraman et al. [41]

- i agents that aquire information about the state, or called sensing.
- j agents that interpret sensed information, converting it into concepts, such as a state estimate.
- k agents that formulate sets of control options
- l agents that decide a control option from the set

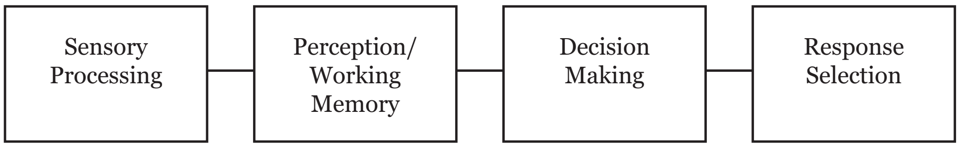


Figure 3.6: Four-stage model of human information processing. [41]

Classic marine control theory pertains only a single ship. A common division of the overall control system into subsystems is according to the Guidance, Navigation and Control (GNC) scheme [21]. The navigation system uses sensors and an observer to generate a state estimate. The guidance system performs task planning and continuously generates a control objective or reference. This is used by the (motion) control system to coordinate responses of available actuators, such that the vessel response follows the reference.

3.2.3. Multi-robot Cooperation

Various challenges in cooperative multi robot systems are systematically described in existing literature. A section of relevant terms and concepts will be discussed to adress the problem of this section.

For a multi-robot system, the division of tasks to a set of robots is referred to as "task allocation" Lerman et al. [35]. This is extended to "dynamic task allocation" if the assignment of tasks to robots needs to be adjusted continuously due to changes in task environment and system response. Lerman et al. [35] describes a "distributed multi-robot system" as a MRS that does not have a central coordinator. Distribution and centralization of control structure both have their benefits and disadvantages. Decisions in distributed control systems need to be made with incomplete information, while a centralized decision making agent has the potential to use all available data. Centralized control structures have, however, a single point of failure and can encounter scaling issues.

As multiple robots collaboratively seek to perform the common task of platform motion control, there are varying approaches to allocating control efforts with similar results, as a vessel platforms actuators are likely far more numerous than the (debatably consistent) degrees of freedom. Fossen [21] shows how a vessel motion control block is commonly divided into a dedicated subsystem that generates 'control efforts', and a block that allocates the desired control effort, by dividing it over the available actuators, schematically shown in figure 3.7. This approach is also used for the configuration adaptive vessel platform controller in Park et al. [42], while therein control allocation is referred to as "coordination" and generation of control efforts is referred to as "robust control".

To use this methodology for control of vessel platform motion, the state estimation would need to pertain platform state, and not that of an individual vessel. This is generally positions η_p and possibly velocities ν_p, which are to be controlled to a desired state. Which parts of the state are to be controlled depends on the approach and complexity of the control system. Dynamic positioning, for instance, controls pose ($\eta = [x, y, \Psi]$) but not directly velocities ($\nu = [u, v, r]$).

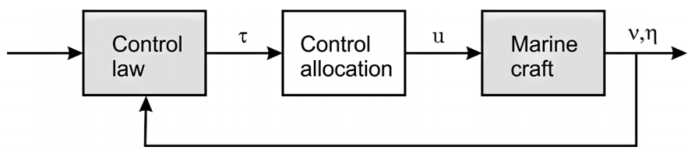


Figure 3.7: Block diagram showing the control-allocation-block in a vessel motion control system. [21].

Park et al. [42] utilizes a centralized control approach to generate control efforts that need to be applied on the platform, while their approach assumes rigid connections. Control effort generation was achieved by means of a PI controller. Control decisions are made centralized on one vessel that is elected as coordinator, which divides the control effort over platform modules. A schematic of the control loop of Park et al. [42] is shown in figure 3.8, showing various elements that could be sensibly interpreted with both the four stage model of Parasuraman et al. [41] and the GNC scheme as described by Fossen [21].

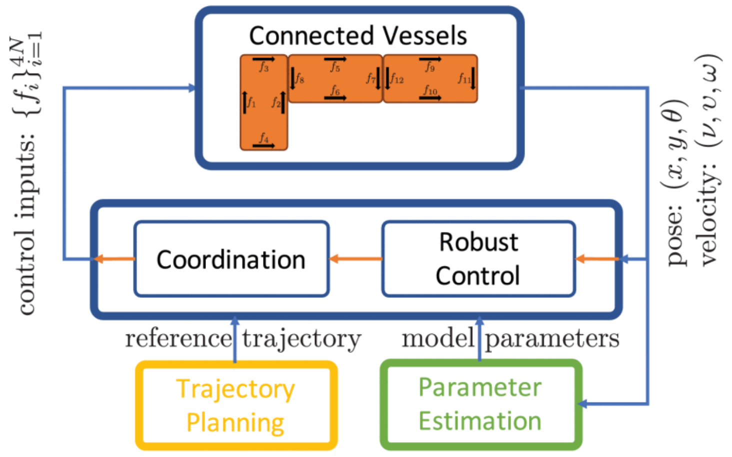


Figure 3.8: Multi-vessel navigation system that consists of parameter estimation, trajectory planning, and coordinated robust control. [42]. Note that the reference trajectory signal is not fed into the coordination block, although it might be interpreted as such on a first glance.

3.3. Effects of combining systems

The previous sections discussed system analysis of modular surface vessel platforming systems that perform automated reconfiguration (section 3.1), and configuration adaptive platform control approaches (section 3.2). This section combines the analysis presented on these two systems by assessing how the functionality of these two systems can be combined. Various design considerations have been identified of which some will be discussed again in this section in a more pragmatic way. Fundamental requirements to achieve the system demands will be presented, yet more emphasis will be given on predicting what choices will be more relevant for commercial logistic applications instead of general robotics.

3.3.1. Requirements

In order to realize both behaviors, various subfunctions have been identified in section 3.2 and 3.1, which are summarized. Table 3.1 presents characteristics of a system that performs Automatic vessel platform reconfiguration, together with the approach and design choices of projects that published about such systems. A similar table on characteristics of vessel platform systems that perform configuration adaptive control strategies are shown in table 3.2:

Fundamental Characteristic or requirement	Roboat project [32] [36]	ISO container module assembly project [40] [43]	Notes
A set of modules	Homogeneous fleet	Homogeneous fleet	Approaches and goals of implementing heterogenous and homogeneous systems differ significantly.
A strategy to reconfigure.	Deterministic	Deterministic	Various approaches are possible which can be categorized as stochastic or deterministic
Means of repositioning modules	Cross shaped non roatable thruster setup. Individual vessels are fully actuated	Plus shaped non roatable thruster setup. Individual vessels are fully actuated	Modules can be designed to perform independent or with help from other agents. Individual modules can be under- or fully actuaded, while the observed trend is towards the latter.
Means of maintaining configuration	Physical ball-cone joint connection [36] or by means of magnets	Physical rope-hook joint connection. Variable stiffness	Various solutions are possible, depending on scale, goal and environment.

Table 3.1: Fundamental characteristics on automated modular vessel platform reconfiguration systems

Fundamental Characteristic or requirement	Park et al. [42]	Notes
Connected robots share a single objective	The platforms position is given as a reference.	Other approaches may also attempt to more directly control speed besides position, or integrate higher level planning (e.g. guidance) tasks.
Robot actions are coordinated	A platform has a centralized controller, deciding and coordinating all connected modules.	Coordinated behavior is aimed to improve performance with respect to the sum of individuals, yet comes at the cost of increased complexity.
The decision making protocol functions on a wide variety of configurations	Control approach uses an approximate model based on the number of connected modules (however, not the configuration shape). The control system yields a single control effort for the entire platform that is subsequently distributed among modules.	Criteria of motion control performance will vary between usecases, as well as the amount of considered configurations.

Table 3.2: Fundamental characteristics on collaborative and coordinated vessel-platform motion control systems

An additional characteristic is identified as these two behaviors are integrated in one framework. As a platform configuration changes through time, the motion control system needs to be able to support real time varying configuration parameter, such as estimated dynamics, size, shape, or centre of mass, or in some cases even the network topology.

3.3.2. Common Considerations

Various considerations for multi robot systems have been observed as key to characterizing a design and are discussed here.

Choosing centralized or decentralized control structures affect many design options for both platform asembly and control. Decentralization generally facilitates great scalability, such that they can be deployed at small scale and in great numbers. A centralized entity can however make decisions based on a more complete state of the overall system, and make more effective decisions. Centralized systems have a single point of failure, which can be undesirable. Centralized systems are also reliant on a communication network with a certain reliability, latency and bandwith.

Homogeneous robot systems consist of robots which are designed to be similar, while heterogeneous systems have modules equipped with different shape, size or abilities. Heterogenous robots can use a wider variety of features without adding too many features per module, although a system topology can get more complex. Homogeneous robots can be interchanged by any other if desired, and are possibly easier to produce in large numbers. All sources that published results on systems that performed automated reconfiguration and configuration adaptive control have been applied using a homogenous fleet.

Vessels that operate individually can be underactuated or fully actuated. Control of underactuated systems is generally more complex, though not impossible. Ashrafiuon et al. [3] reviews different approaches of controlling automated underactuated surface vessels. This work focusses on control approaches which are categorized in "setpoint", "trajectory tracking" and "path following" approaches. Light is shed on advantages and disadvantages of various mentioned approaches. It is convenient to have fully actuated modules for vessel platforming purposes, which is shown to be feasible by projects that use fully actuated vessels for both automated reconfiguration and configuration adaptive control.

Designing towards utilizing extensive amounts of communication in a multi robot system drastically changes availability of further design options. A vessel system connected into a network that continuously shares information can become reliant on the presence and performance of the network. Facilitating connectivity comes to cost at various facets. Modules need this extra feature built in, which makes a module more complex, expensive, power consuming, while adding another component that can break. Nihilistic approaches that only add essential features to a module will increase scalability, due to low cost, replacability and reliability.

To facilitate collaborative platform control, some communication between operator and module is always necessary, as the task of the platform needs to be communicated from human to the vessel system. Many multi robot systems that are nowadays developed often have a high reliance on communication network availability, as task distribution has many benefits and opens up new doors in terms of design choices and possible emergent behaviors.

This chapter explored the meaning two behaviors of automated vessel platform control systems reffered to as ’automated reconfiguration’ and ’collaborative and coordinated platform motion control’. Various approaches to describing and designing such systems were discussed to deepen understanding of the behaviors and broaden the view on potential design solutions. Fundamental characteristics of both evaluated systems have been identified to aid development of a framework integrating the two in the next chapter.

4

Multi-vessel System State Description

This chapter builds on existing vessel state descriptions to formulate a multi-vessel state description. Furthermore, a novel approach to predict multi-vessel platform dynamics from individual models is proposed to answer the research subquestion:

- How can the dynamics of the multi-vessel system be represented?

The described multi-vessel state description as described in Sec. 4.1 aims to enable consistent system notation of scenarios pertaining more than a single vessel, as mult-vessel frameworks have more objects that can be referred to. The concept and definition of a platform-coordinate system is also introduced.

For collaborative motion control of modular objects an approximate dynamical model is often desired. As model parameter estimation experiments are often infeasible for every configuration of a combined structure, a prediction of dynamical behavior is proposed using the often known dynamics of modules in Sec. 4.2

4.1. System State Notations

A sensible notation of a system aids implementation and consistent discussion of motion control systems. Vessel automation is a topic where robotic and maritime science overlap. System description of vessel systems from a control perspective is described in Fossen [21], which are used as a foundation of the multi-vessel state description

4.1.1. Single vessel state description

For control purposes, ships are generally considered rigid bodies. A rigid body has 6 degrees of freedom (DOF) in which can all be evaluated or simplified to a reduced set of motions such as 3-DOF planar motion. For ships, this is commonly expressed according to the notation of SNAME [48] with respect to various coordinate systems. Components of orientation, velocities and forces are shown in table 4.1.

DOF		Positions and Euler Angles	Linear and angular Velocities	Forces and Moments
1	Surge	x	u	X
2	Sway	y	v	Y
3	Heave	z	w	Z
4	Roll	ϕ	p	K
5	Pitch	θ	q	M
6	Yaw	ψ	r	N

Table 4.1: SNAME notation for marine vessels

A North-East-Down coordinate system will be used as reference to a global coordinate system as $\{n\}$, while a body fixed coordinate system is referred to with $\{b\}$. This body fixed frame defines the motion of the vessel, such that every position or speed of a vessel is defined in the motion of the body frame. Figure 4.1 shows vessel motions expressed on the body fixed frame. Motion of surface vessels is commonly simplified to 3 degrees of freedom in the surface plane, as is shown in figure 4.2. This assumes effects of heave, roll and pitch can be neglected. This simplification of planar motion is adopted for the developed control system in this paper as described in Chap. 5.

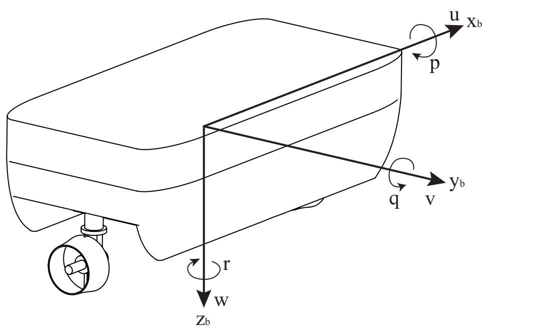


Figure 4.1: Six degrees of motion depicted of a Delfia vessel expressed with respect to the body-fixed coordinate system {b} in convention with SNAME [48]

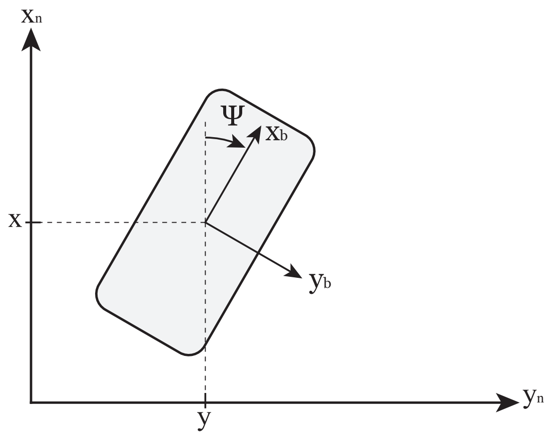


Figure 4.2: Three degrees of freedom depicted as only motion in the surface plane is considered

Position, orientation, velocities, forces and moments are further noted as:

$\mathbf{p}_j^i$	=	Position of point j expressed in coordinate system {i}
Θ_{ij}	=	Euler angles between coordinate systems {i} and {j}
$\mathbf{v}_{i/j}^k$	=	Linear velocity of point i with respect to j expressed in coordinate system {k}
$\omega_{i/j}^k$	=	Angular velocity of object i with respect to j expressed in coordinate system {k}
f_j^i	=	Force with line of action through point j expressed in coordinate system {i}
m_j^i	=	Moment about point j expressed in coordinate system {i}

As a result of the reduction from 6 to 3 degrees of freedom, vectorial expressions of our system become:

NED position	$\mathbf{p}_b^n$	=	$\begin{bmatrix} x_b^n \\ y_b^n \end{bmatrix}$	Attitude (Euler angles)	Θ_{nb}	=	$\begin{bmatrix} \Psi_b^n \end{bmatrix}$
Body-fixed linear velocity	$\mathbf{v}_{b/n}^b$	=	$\begin{bmatrix} u \\ v \end{bmatrix}$	Body-fixed angular velocity	$\omega_{b/n}^b$	=	$\begin{bmatrix} r \end{bmatrix}$
Body-fixed force	f_b^b	=	$\begin{bmatrix} X \\ Y \end{bmatrix}$	Body-fixed moment	m_b^b	=	$\begin{bmatrix} N \end{bmatrix}$

General motion of vessels in 3 degrees of freedom are described by the following generalized positions and velocities [21]

$$\eta = \begin{bmatrix} \mathbf{p}_b^n \\ \Theta_{nb} \end{bmatrix} = \begin{bmatrix} x_b^n \\ y_b^n \\ \Psi_b^n \end{bmatrix} \tag{4.1}$$

$$\nu = \begin{bmatrix} \mathbf{v}_{b/n}^b \\ \omega_{b/n}^b \end{bmatrix} = \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{4.2}$$

$$\tau = \begin{bmatrix} f_b^b \\ m_b^b \end{bmatrix} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} \tag{4.3}$$

Where η describes orientation, v describes velocities and τ describes forces acting upon the system. To avoid clutter, when it is deemed clear to in which reference frame a position or velocity is expressed, the subscript and superscript will not always be shown. This is for instance the case for generalized positions as in equation 4.1 that is most of the time with respect to $\{n\}$ such that it is just expressed in $[x, y, \Psi]^\top$. By default, v, τ are expressed in $\{b\}$, and η is described in $\{n\}$.

Velocities expressed in $\{b\}$ and $\{n\}$ frame are related as:

$$\dot{\eta} = R(\Psi)v \tag{4.4}$$

Where R is the rotation matrix of Ψ around the z axis. For pure motion in the 3 surface plane degrees of freedom, this becomes:

$$R(\Psi) = \begin{bmatrix} cos(\Psi) & -sin(\Psi) & 0 \\ sin(\Psi) & cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Coordinate system origin and centre of mass are referred to as:

$\mathbf{o}_i^j$	=	Origin of coordinate system i expressed in coordinate system $\{j\}$
$\mathbf{p}_{cm}^i$	=	Position of the centre of mass expressed in coordinate system $\{i\}$

4.1.2. Multivessel system notation

As fleet systems transition from describing a single vessel to a system of many, it becomes necessary to differentiate between local frames of different vessels. We change subscripts referencing to a single vessel to a notation for n vessels. For example, referencing to a single body coordinate system as $\{b\}$ changes to $\{b1\}, \{b2\}, \{b2\} \dots \{bn\}$ in order to indicate which body frame is referenced to. Relative orientations and velocities are considered to become more relevant as vessels have to operate in proximity or even contact to realize assembly into platforms. Figure 4.3 illustrates how position of a vessel can be described in a local body fixed frame, which may be more relevant and intuitive for automated assembly purposes. The position of origin of body 2 expressed in $\{b1\}$ illustrated in figure 4.3 becomes:

$$\mathbf{o}_{b1}^{b2} = \begin{bmatrix} x_{b1}^{b2} \\ y_{b1}^{b2} \end{bmatrix} \tag{4.6}$$

Furthermore, the notations of motion shown in equation 4.1, 4.2 and 4.3 by default refer to a specific frame. Pose η is expressed globally in $\{n\}$. Velocities v and forces τ are expressed in the body fixed frame $\{b\}$. It can be convenient to express motions to different frames as well. Take the example of two moving vessels docking to eachother while moving. To answer the question "How close is a vessel to a docking position?" a relative expression of position is required. The convention of subscripting to refer to a coordinate system is extended to η, v and τ as well as follows:

$$\eta_b^n = \begin{bmatrix} \mathbf{p}_b^n \\ \Theta_{nb} \end{bmatrix} = \begin{bmatrix} x_b^n \\ y_b^n \\ \Psi_b^n \end{bmatrix} \tag{4.7}$$

Where the sub- and superscript of η_b^n refer to object b expressed in $\{n\}$. This convention allows expression of relative motion between vessels by writing η_{b2}^{b1} which indicates pose of body 2 expressed in the local frame of body 1, also illustrated in 4.3. Note that velocities and forces are regularly referred to as "body-fixed", but that is not necessarily the case anymore with this convention.

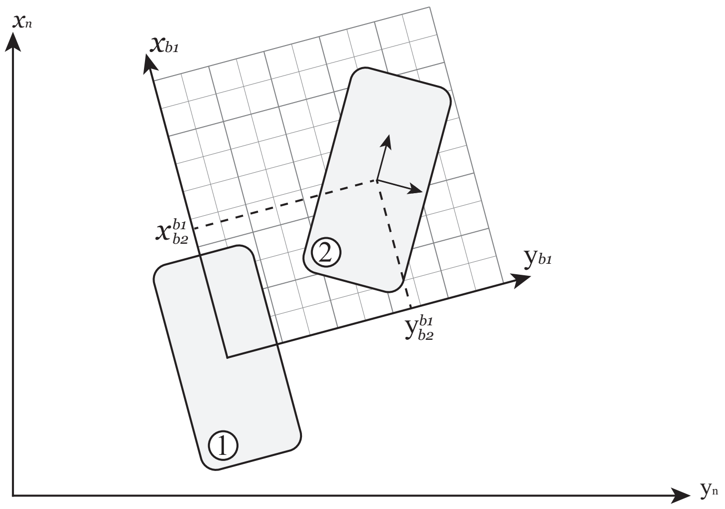


Figure 4.3: Two vessels are shown from above, illustrating interpretation of expressing a module's state (of module 2) in another module's body-fixed coordinate system (of module 1).

4.2. Dynamical model of a Platform

An approximate model for the platform dynamics is used by the platform controller in the process of generating control effort. When the platform control agent is notified that its configuration is changed, it recomputes various parameters for the dynamical model that describe the new rigid body dynamics. The platform control agent keeps track of the following

- Connectivity of modules
- Configured pose connected modules
- Parameters that describe dynamics of individual modules

Use-cases where vessel assemblies are formed in many varying configurations can make experimental model parameterization infeasible for all reasonably forseeable configurations. Predicting a dynamical model by combining models of modules can provide a quick, cheap and scalable solution with respect to performing parameter estimation experiments. The approach to estimating the dynamical model of a platform is explained in this section, but also more elaborately described and discussed in appendix B

A dynamic model of the platform is formed by expressing all models of the modules in the same point and coordinate system, which are then combined. It is shown how terms from multiple module models can be grouped for convenient expression, and how the centre of mass of the combined structure can be found.

Models of modules are expressed in platform frame origin by (1) translating the expressions to o_p as a reference point and (2) rotating the expressions to match coordinate system $\{p\}$. This approach works also for models of which the origonal inertial matrix is not defined in the CG, and/or for models that have directional dependent mass (e.g., hydrodynamic added mass).

Generalized positions and velocities of the platform are described respectively as

$$\eta^n_{p/n} = \begin{bmatrix} \mathbf{p}^n_p \\ \Theta_{np} \end{bmatrix} \tag{4.8}$$

$$\nu^p_{p/n} = \begin{bmatrix} \mathbf{v}^p_{p/n} \\ \omega^p_{p/n} \end{bmatrix} \tag{4.9}$$

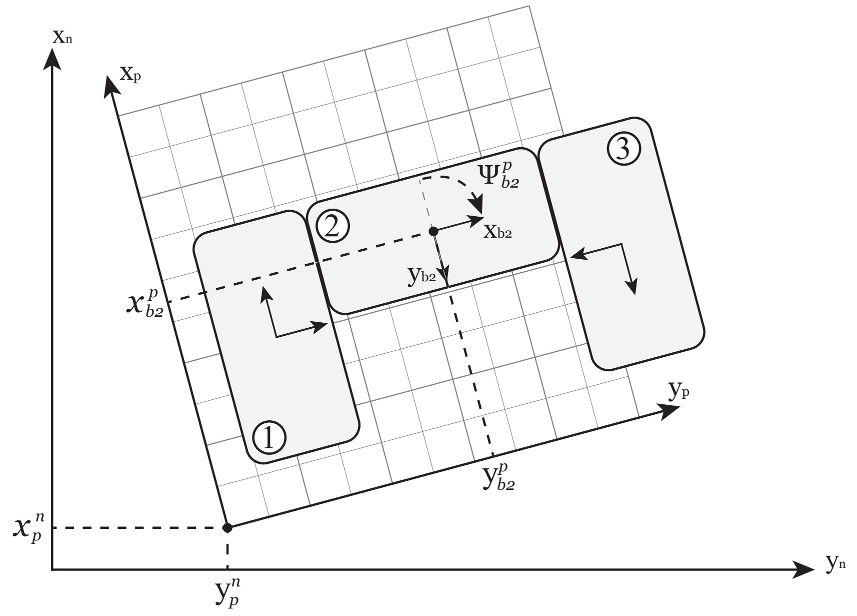


Figure 4.4: An assembly of three vessels in 2d. Components from equation 4.10 that define configured pose of vessel 2 in the platform coordinate system ({p}) are illustrated, as well as position of the platform frame origin in ({n})

The placement of all vessels (defined by the position and orientation of coordinate system {b}) within the assembly is known and can be described with respect to {p} expressed in {p} as

$$\eta^p_{b/p} = \begin{bmatrix} \mathbf{p}^p_b \\ \Theta_{pb} \end{bmatrix} \tag{4.10}$$

altough, instead of euler angles Θ_{pb} to express relative orientation, the rotation matrix $\mathbf{R}^p_b$ from coordinate system {b} to {p} will be more often used throughout this work. The assembly and the placement of the platform frame are considered rigid, thus there is no motion between them, such that relative velocity of a module expressed in rotating frame {p} equals (if derivative is taken in rotating, non-inertial frame{b} or {p})

$$\nu^p_{bi/p} = \nu^p_{bi/bj} = \begin{bmatrix} \mathbf{v}^p_{bi/p} \\ \omega^p_{bi/p} \end{bmatrix} = 0 \tag{4.11}$$

and

$$\frac{d}{dt}\mathbf{R}^p_b = 0 \tag{4.12}$$

Translating and rotating velocities allows us to express module motion in terms of generalized coordinates of the platform

$$\mathbf{v}^b_{b/n} = R^b_p[\mathbf{v}^p_{p/n} + \mathbf{S}(\omega^p_{p/n})\mathbf{p}^p_{b/p}] \tag{4.13}$$

$$\omega^b_{b/n} = \mathbf{R}^b_p\omega^p_{p/n} \tag{4.14}$$

Similarly, for forces and moments can be expressed in other frames of reference as

$$\mathbf{f}^p_p = \mathbf{R}^p_b\mathbf{f}^b_b \tag{4.15}$$

$$\mathbf{m}^p_p = \mathbf{R}^p_b(\mathbf{m}^b_b + \mathbf{p}^b_{p/b} \times \mathbf{f}^b_b) \tag{4.16}$$

Velocities and generalized forces can be converted to vector notation as

$$\nu^b_{b/n} = \begin{bmatrix} \mathbf{v}^b_{b/n} \\ \omega^b_{b/n} \end{bmatrix} = \mathbf{J}^b_p\mathbf{H}(\mathbf{p}^p_{b/p})\nu^p_{p/n} \tag{4.17}$$

$$\tau_p^p = \begin{bmatrix} \mathbf{f}_p^p \\ \mathbf{m}_p^p \end{bmatrix} = \begin{bmatrix} \mathbf{R}_b^p \mathbf{f}_b^b \\ \mathbf{R}_b^p(\mathbf{m}_b^b + \mathbf{p}_{p/b}^b \times \mathbf{f}_b^b) \end{bmatrix} = \begin{bmatrix} \mathbf{R}_b^p & 0 \\ 0 & \mathbf{R}_b^p \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{S}(\mathbf{p}_{p/b}^b) & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}_b^b \\ \mathbf{m}_b^b \end{bmatrix} = \mathbf{J}_b^p \mathbf{H}^\top(\mathbf{p}_{p/b}^b)\tau_b^b \tag{4.18}$$

where coordinate system transformation between rotated frames $\{p\}$ and $\{b\}$ is done by operator

$$\mathbf{J}_b^p = \begin{bmatrix} \mathbf{R}_b^p & 0 \\ 0 & \mathbf{R}_b^p \end{bmatrix}, \qquad \mathbf{J}_b^{p\top} = \begin{bmatrix} \mathbf{R}_b^p & 0 \\ 0 & \mathbf{R}_b^p \end{bmatrix} \tag{4.19}$$

and translation of forces is represented by operator (Fossen [21])

$$\mathbf{H}^\top(\mathbf{p}_{p/b}^b) = \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{S}(\mathbf{p}_{p/b}^b) & \mathbf{I} \end{bmatrix}, \qquad \mathbf{H}(\mathbf{p}_{p/b}^b) = \begin{bmatrix} \mathbf{I} & -\mathbf{S}(\mathbf{p}_{p/b}^b) \\ 0 & \mathbf{I} \end{bmatrix} \tag{4.20}$$

Motion of an individual rigid body can be described in $\{b\}$ as (Fossen [21])

$$\mathbf{M}\dot{\nu}_{b/n}^b + \mathbf{C}(\nu_{b/n}^b)\nu_{b/n}^b = \tau_{res} \tag{4.21}$$

where $\mathbf{M}$ represent inertia of the rigid body and constant hydrodynamic added mass, and $\mathbf{C}(\nu_{b/n}^b)\nu_{b/n}^b$ represent coriolis and centripetal forces.

Coriolis and centripetal forces arise due to the rotation of $\{b\}$ with respect to the inertial frame, and are fully determined by the inertial matrix. Fossen [21] shows how an energy approach, using Kirchhoff's equations is a convenient way to find the coriolis and centripetal matrix. If kinetic energy of vessel and added mass is written in quadratic form (Kirchhoff [33])

$$\mathbf{T} = \frac{1}{2}\nu_{b/n}^{b\ \top}\mathbf{M}^b\nu_{b/n}^b \tag{4.22}$$

where inertial matrix and velocities are described in $\{b\}$, and inertial matrix $\mathbf{M}^b$ contains inertia of the rigid body and added hydrodinamic mass. Substituting 4.17 gives

$$\mathbf{T} = \frac{1}{2}\nu_{p/n}^{p\ \top}\mathbf{H}^\top(\mathbf{p}_{b/p}^p)\mathbf{J}_p^{b\top}\mathbf{M}^b\mathbf{J}_p^b\mathbf{H}(\mathbf{p}_{b/p}^p)\nu_{p/n}^p \tag{4.23}$$

Which can be rewritten as

$$\mathbf{T} = \frac{1}{2}\nu_{p/n}^{p\ \top}\mathbf{M}^p\nu_{p/n}^p \tag{4.24}$$

where the inertial matrix of a module is expressed in platform coordinates as

$$\mathbf{M}^p = \mathbf{H}^\top(\mathbf{p}_{b/p}^p)\mathbf{J}_p^{b\top}\mathbf{M}^b\mathbf{J}_p^b\mathbf{H}(\mathbf{p}_{b/p}^p) \tag{4.25}$$

Equation 4.24 can be substituted in Kirchhoff's vector equations (Kirchhoff [33])

$$\frac{d}{dt}\left[\frac{\partial \mathbf{T}}{\partial \nu_1}\right] + \mathbf{S}(\nu_2)\frac{\partial \mathbf{T}}{\partial \nu_1} = \tau_1 \tag{4.26}$$

$$\frac{d}{dt}\left[\frac{\partial \mathbf{T}}{\partial \nu_2}\right] + \mathbf{S}(\nu_2)\frac{\partial \mathbf{T}}{\partial \nu_2} + \mathbf{S}(\nu_1)\frac{\partial \mathbf{T}}{\partial \nu_1} = \tau_2 \tag{4.27}$$

where $\nu_1 = \mathbf{v}_{p/n}^p$, $\nu_2 = \omega_{p/n}^p$, $\tau_1 = \mathbf{f}_p^p$ and $\tau_2 = \mathbf{m}_p^p$ to obtain the equations of motion of a module expressed in platform coordinates. Notice that the expression of inertial matrix in equation 4.25 is constant, due to rigid body assumptions. This allows formation of the (tranlated and rotated) dynamical model in a 'normal' fashion, such as shown in Fossen [21], where terms that are not dependent on accelleration, but on velocity are grouped to form the coriolis-centripetal matrix, which can be represented in many forms. Various works describe options parameterizations such as skew-symmetric Sagatun and Fossen [44] or velocity independent Fossen and Fjellstad [22], which can be chosen to best suit a project.

The assumption of a rigid assembly allows summation of forces on modules in a platform, given that they are expressed in the same point and coordinate system. If n connected modules generate a generalized force

τ_{bi} expressed in the same point and coordinate system, the forces can be added to find the total force for the entire platform as

$$\tau^p_p = \sum_{i=1}^{n} \tau^p_{bi} \tag{4.28}$$

Which can allow convenient reformulation of a total model by grouping certain terms. Grouping of terms allows expression of 'platform inertia', 'total dampening' or 'total control-effort', to name only some. For instance, expressing inertia of various modules in a the same platform coordinates allows us to express total platform-inertia

$$\mathbf{M}^p_{platform} = \sum_{i=1}^{n} \mathbf{M}^p_{bi} = \sum_{i=1}^{n} \mathbf{H}^\top(\mathbf{p}^p_{bi/p}) \mathbf{J}^{bi}_p{}^\top \mathbf{M}^{bi}_{bi} \mathbf{J}^{bi}_p \mathbf{H}(\mathbf{p}^p_{bi/p}) \tag{4.29}$$

which can also conveniently be used to compute terms regarding coriolis and centripetal forces for the complete platform in one go, instead of obtaining it by summating the coriolis and centripetal matrices of all modules.

Other forces can be expressed in platform coordinates by substituting equation 4.18 and 4.17. For example, forces due to linear viscous dampening can be described as

$$\tau^b_{damp} = \mathbf{D}^b \nu^b_{b/n} \tag{4.30}$$

Substitution yields

$$\tau^p_{damp} = \mathbf{J}^b_p{}^\top \mathbf{H}^\top(\mathbf{p}^b_{p/b}) \mathbf{D}^b \mathbf{J}^b_p \mathbf{H}(\mathbf{p}^p_{b/p}) \nu^p_{p/n} \tag{4.31}$$

Similar forces acting on the platform, such as dampening, can be grouped.

$$\tau^p_{damp,total} = \sum_{i=1}^{n} \tau^p_{damp,i} \tag{4.32}$$

In the case of linear dampening, these terms result in a constant dampening matrix.

$$\tau^p_{damp,total} = \mathbf{D}_{total} \nu^p_{p/n} \tag{4.33}$$

where

$$\mathbf{D}_{platform} = \sum_{i=1}^{n} \mathbf{J}^{bi}_p{}^\top \mathbf{H}^\top(\mathbf{p}^{bi}_{p/bi}) \mathbf{D}^{bi} \mathbf{J}^{bi}_p \mathbf{H}(\mathbf{p}^p_{bi/p}) \tag{4.34}$$

Combining equation 4.29, an accompanying expression for coriolis and centripetal forces, and other forces in platform frame yield the expression of overall platform dynamics. This becomes with, for example linear dampening from equation 4.33

$$\mathbf{M}^p_p \dot{\nu}^p_{p/n} + \mathbf{C}_p(\nu^p_{p/n}) \nu^p_{p/n} + \mathbf{D}_p \nu^p_{p/n} = \tau^p_{res} \tag{4.35}$$

From the expression of total platform inertial tensor as equation 4.29 we can find the centre of gravity of the platform. Recall that the centre of gravity is the point of a rigid object, if a (gravitational) force is applied, this force creates no resultant torque, and thus no angular accelleration. This effectively means that if we find the position of platform centre of gravity $\mathbf{p}_g$, and express our platform model in that point (similar as in equation 4.25), no coupling between rotation and translation should exist in the inertial matrix. Expressing the inertial matrix in CG_p can be done by:

$$\mathbf{M}^{CG}_p = \mathbf{H}^\top(\mathbf{p}^p_{g/p}) \mathbf{M}^p_p \mathbf{H}(\mathbf{p}^p_{g/p}) \tag{4.36}$$

No coupling in $\mathbf{M}^p_p$ between rotation and translation means that the off-diagonal quadrants are zero, thus if

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \tag{4.37}$$

then

$$\mathbf{M}^{CG}_{12} = \mathbf{M}^{CG}_{21} = 0_{3x3} \tag{4.38}$$

evaluating the resulting upper right quadrant of equation 4.36 and equation 4.38 gives

$$\mathbf{M}_{CG,12} = \mathbf{M}_{p,12} - \mathbf{M}_{p,11}\mathbf{S}(\mathbf{p}_g^p) = 0_{3x3} \tag{4.39}$$

$$\mathbf{S}(\mathbf{p}_g^p) = \mathbf{M}_{p,11}{}^{-1}\mathbf{M}_{p,12} \tag{4.40}$$

which allows us to easily extract the center of mass for the combined structure by substituting known inertial parameters and solving for $\mathbf{p}_g^p$.

Notes on representativeness of this model
It can be debated whether the summation of individual ship dynamic models sufficiently represents overall stucture dynamics. Humphreys and Watkinson [28] stated (about individually operating vessels) that forces and moments represented by acceleration hydrodynamic coefficients can, to a very great extent, be modeled as potential flow phenomena.

Yet platforms have vessels operating in very close proximity, which might significantly affect the boundary layer size and shapes. This can affect terms representing added mass both positively and negatively, which is illustrated with two examples.

- Nearby connected modules that 'trap' a volume of water between the hulls can cause the mass of this volume to be effectively added to the combined body, as this volume now moves together with the platform, thus raising added mass terms.
- Nearby connected modules that have overlapping boundary layers can decrease effective added mass, as there is less volume trapped between modules than would elswise form a boundary layer.

The interaction of boundary layer shape and size with vessels operating in close proximity can have great effects on effective added mass terms of combined structures. The impact of module proximity on added mass terms should be considered while taking into account required model accuracy needed for the control algorythm. Models for other forces, such as dampening, should also be evaluated whether they are reasonably representative for operation of modules in close proximity.

This chapter described the multi-vessel state notation that allows consistent referencing to objects in a scenario pertaining multiple waterborn structures. Parameters that describe a parameter of a ship (such as forces, positions and velocities) are identifyable with subscripts indicating which object it pertains. The same approach has been applied to platform state description, such that various body ($[\{b_1\}, \{b_2\}, ..., \{b_i\})$, platform ($[\{p_1\}, \{p_2\}, ..., \{p_i\})$ and global (inertial frame $\{n\}$) are distinguished. Furthermore, an approach to predicting dynamcis of a combined waterborne structure from has been proposed that uses often known models of individual modules as building blocks. Both the multi-vessl state notation and the proposed platform model will be used in the next chapter during the development of a control system of the described structure.

5 System Development

This chapter describes development of a multi-vessel platforming system. Major considerations for developing vessel-platform systems that are self-assembling and show collaboration and coordinated motion control as described in Chap. 3 are used for decisionmaking on design and implementation. The multi-vessel-system notation and approach to predict dynamic behavior of a combined waterborne structure as described in Chap. 4 are used to develop the control algorythm in this chapter. The first part of this chapter shows conceptual, high level design choices, such as network topology, control approach and subdivision of the entire system into modular subsystems. The second part of this chapter focusses on implementation of the design by giving information of the technologies, hardware and software structure that give physical form to the distinguished subsystems.

5.1. Design

The overall design of the control framework is created with the aim to meet the following set criteria.

- The system is required to perform automated vessel platform reconfiguration.
- The system needs to perform motion control in a collaborative manner, where multiple robots work together to achieve a single goal.
- The system needs to support both above mentioned behaviors simultaneous.

Besides hard constraints, several concepts play a key role for making design decisions.

- The framework should support a large amount of-, or arbitrary configurations. This creates adaptiveness to a wide set of tasks for modular vessel platforms, which is a key element supporting commercial competitivity.
- Developed solutions are aimed to be general, such that they are applicable or at least meaningful on other ship-systems, environments and scales of operation. This is considered important to stimulate that knowledge gained from the developed experimental setup benefits future commercial implementation.
- Solutions are aimed to be modular, such that subsystems are designed to be conveniently swapped out for another (perhaps improved and better performing) version, easing future improvements and increasing reusability of work.

The goal of this design is not to optimize an existing system, but to explore a novel combination of behaviors that is expected to be of interest in the near future. The ability of succesful assembly needs to be proved, yet successrate and reconfiguration-speed need only be in reasonable limits and magnitude to reflect commercial implementation. Cooperative motion control needs to show convergence of the system to a desired state while a network of robots operate more effective than the sum of individuals, yet quantitiative motion responses need only be stable and within reasonable timeframe. For quantitative performance evaluations (Chap. 6), criteria are set representing demands of a logistical usecase, shown in Tab. 5.1.

Key performance indicator	Criteria
Risetime*	t_r <10s
Settlingtime*	t_s <40s
Overshoot*	ov<50%
Relative motion between connected modules	≈ 0

Table 5.1: Performance criteria of the designed system. Risetime, Settlingtime and Overshoot pertain criteria for motion control system evaluation, while relative motion between connected modules indicates assembly success. (* evaluated for step inputs 1.0 & 0.5m translation and 90grad rotation as characteristic motions)

Description on design of the experimental setup starts from a high level view by discussing the multi-robot network topology adapting to varying configurations. It continues to explain the general control approach and division in subsystems that each solve a part of the control problem. Design choices of each subsystem are subsequently discussed in terms of model estimation, state estimation, control effort generation, control effort allocation and assembly protocol.

5.1.1. Fleet Network Topology

The modules that are used will be rectangular, equipped with two rotating azimuth thrusters. All modules are identical, so a homogeneous fleet is formed. The design has two axes of symmetry, including weight distribution and thruster placement. The origin of the vessel's body fixed coordinate system will be defined where the planes of symmetry coincide. The Reseachlab Autonomous Shipping (RAS) Delft has a fleet of such vessels available, named Delfia's (figure 5.1). The vessels dimensions are approximately 380 mm long and 200mm wide. The latest version (the Delfia-1*) is intended to be equipped with a Raspberry-Pi to perform some on board computation tasks whilst also enabling communication via WiFi.

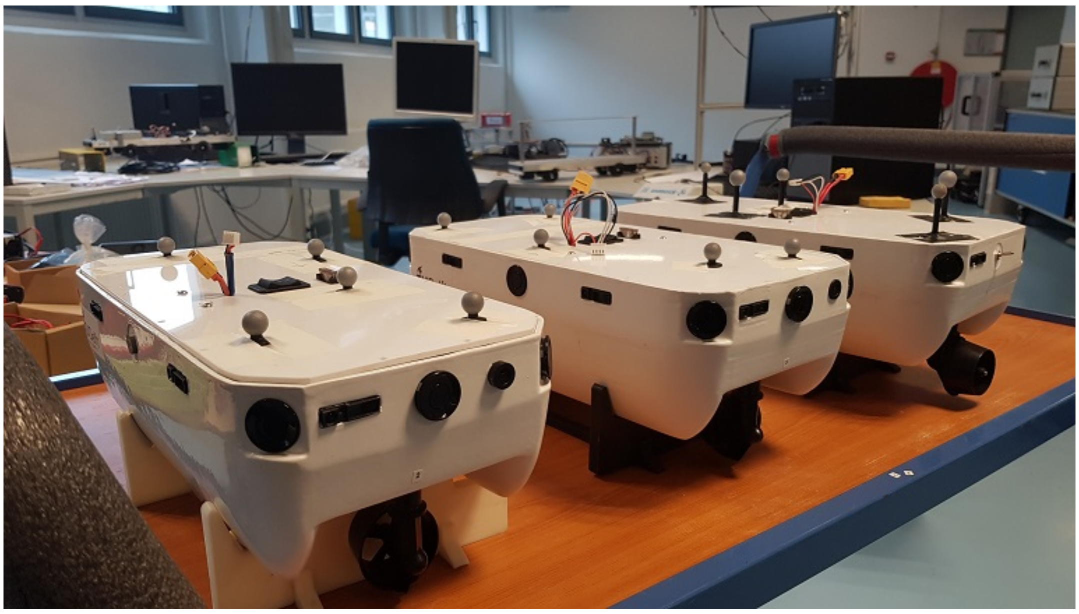


Figure 5.1: Delfia-* vessels, that will act as modules of the fleet system, to perform automated assembly and control.

The approach of dividing a vessel control system via the Guidance, Navigation and Control categorization (Fossen [21] & section 3.2) is used to distinguish between different system processes. However, the control for this particular system does refer to that of a single vessel, but rather to a set of vessels, which sometimes are controlled individually (when a module moves alone), and in other times together (in assembled platform). Hence, not only the system behavior changes (such as inertia of a platform of variable shape and size), but also the control structure.

A control structure has been developed, working on three levels, depicted in figure 5.2. The concept of a 'platform' and a corresponding platform-controller are key to the functioning of this system. Various tasks of the system are divided across agents that can be roughly described as follows:

- A fleet manager
- A platform manager
- A module

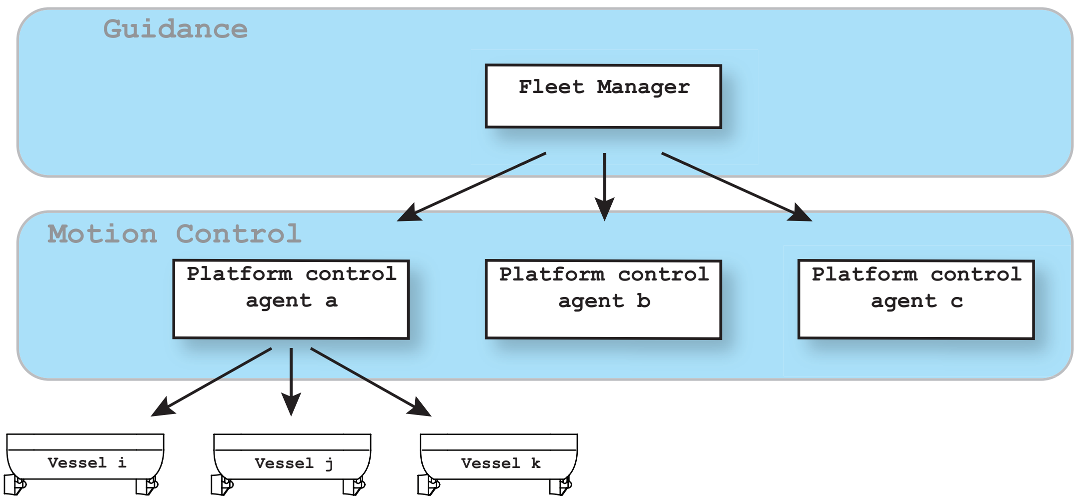


Figure 5.2: Hierarchical topology of the network. A single agent manages various platforms control agents. Platform control agents each manage a unique set of modules.

A single fleet manager performs vessel guidance and coordination of the assembly process. These tasks are designed to operate as a rather simplistic state machine that runs trough phases based on system triggers or operator input.

Tasks generated by the fleet manager are passed onto a set of Platform-controllers, which manage motion control of a set of connected modules in a centralized fashion. For every platform, a single agent is responsible for making decisions for all modules in the platform. This platform controller uses a reference state and a platform-state estimation to generate actuator responses for all modules. These are sent to the corresponding modules that make up the platform, which are then interpreted and executed. The platform controlling agent can be embodied by a computer anywhere on the network, given that it has sufficient computational power and the network is stable and reliable.

The amount of modules that an platform-control-agent manages varies over time as vessels attach or disconnect. Thus all functions of the platform controller needs to be able to handle a variable amount of modules and configured shape.

Communication between agents is facilitated by WiFi and the Robotic Operating System (ROS) as middleware. ROS facilitates communication between agents in a multi-robot system in an interoperable and modular way. This is thanks to the publisher-subscriber approach, which allows a variable amount of agents to listen (subscribe) to a dedicated datastream (topic). Similarly, a variable amount of agents can send (publish) various datatypes to such topics. This middleware has several benefits, one of which is the ability to set up various topics that represent some system information, on which agents running on various operating systems can send and receive data. The publisher-subscriber approach allows agents to only be subscribed to topics which are relevant. Furthermore, ROS also has many standardized message formats, libraries, and a large and rapidly developing userbase that further enhances interoperability.

State estimation of modules is done by an on-shore optical tracking system. Each vessel has a dedicated topic on which these estimates are published, which are accessible for all entities within the network by subscribing to that particular topic.

The fleet manager has, besides creating references for the platforms, also the task of coordinating module ownership. As (dis)assembly criteria are met, platform controllers exchange ownership of a module. This informs a platform-controller that a module is connected, and that it now has access to utilizing that module's actuators. Figure 5.3 and 5.3 illustrate transfer of ownership of a module between two platform controllers, which can leave platform-controller-agents without modules under it's control, rendering it inactive. Each module will be controlled by a single platform-controller at a time. The platform controller will have full knowledge of the determined pose of each module in the platform with respect to the platform-coordinate system.

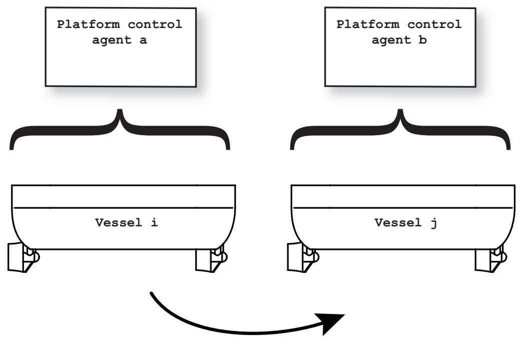


Figure 5.3: Two vessels that are both controlled by a separate control agent. A change in control structure can transfer module ownership (arrow).

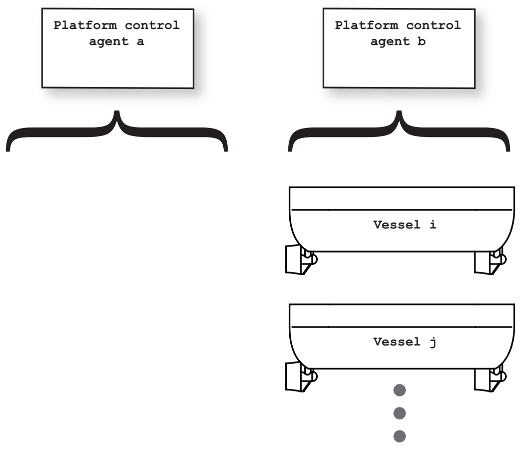


Figure 5.4: Two vessels that formed into a platform, after which they are both controlled by the same agent.

5.1.2. Platform Level Control Approach

The platform controller has the goal of generating and publishing actuator commands for connected modules such that the platform motion follows the reference given by the fleet manager. The combined structure formed by the assembling modules will have a changing size and shape . This results in variable platform dynamics and number of available actuators. The overall approach on controlling the platform is illustrated in figure 5.5. As modules connect or disconnect, the platform controller forms a model of the structure in the new configuration. The approach for estimating this model is elaborated in section 4.2. Motion control is based on state feedback, where control effort generation and allocation are separated. The control effort generation block uses the estimated model to adapt actuator behavior to maintain performance while the system dynamics change. This control approach considers planar motion in 3-DOF (x, y, and yaw), neglecting pitch, heave and roll.

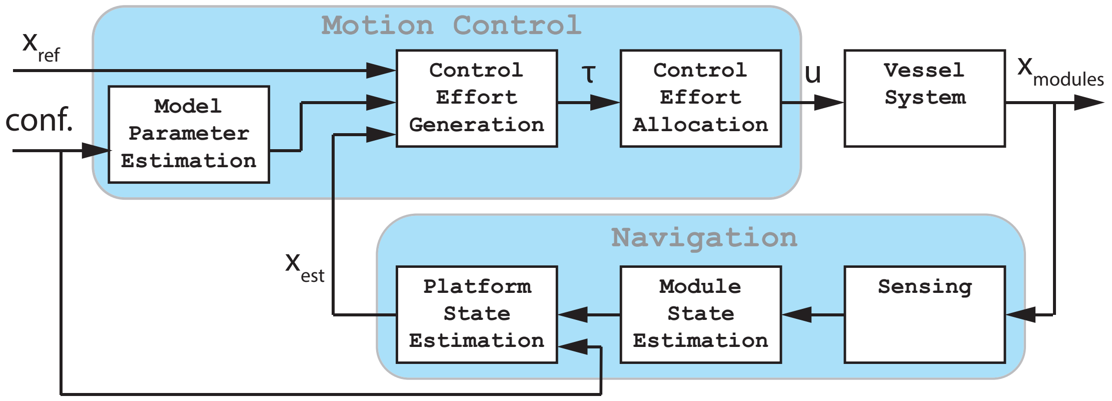

Motion Control
X_ref
conf.
Model Parameter Estimation
Control Effort Generation
τ
Control Effort Allocation
u
Vessel System
X_modules
Navigation
X_est
Platform State Estimation
Module State Estimation
Sensing

Figure 5.5: The designed platform-motion control loop. State reference X_{ref} is the main input from the Fleet manager (guidance system) and output is vessel state X. Occasional changes to configuration are communicated to the platform motion controller for adapting control strategy.

The goal is to find actuator commands for all modules under command of the controller such that the platform state approaches a reference state given by the fleet manager. The reference state will be a position and heading, noted as

$$X_{ref} = \eta_{ref} = \begin{bmatrix} x_{ref} \\ y_{ref} \\ \Psi_{ref} \end{bmatrix} \tag{5.1}$$

which is expressed in $\{n\}$. Note that other approaches might also include velocities as a state that is to be controlled, making the reference, estimated and actual state X a [1x6] vector for motion in the surface plane instead of the [1x3] description as shown in equation 5.1.

The control problem is solved in the following steps

- An approximate model of the combined body is formed. Control gains are adjusted to estimated model parameters.
- Platform state is estimated
- From this model, reference state and estimated state - control efforts are generated by means of a proportional-integrator-derivative (PID) controller.
- Control efforts are allocated between actuators on the modules.

The state of the platform is defined as the origin and orientation of platform-body-fixed coordinate system $\{p\}$, expressed as

$$\eta^n_{p/n} = \begin{bmatrix} x^n_{p/n} \\ y^n_{p/n} \\ \Psi^n_{p/n} \end{bmatrix} \tag{5.2}$$

which is expressed in $\{n\}$. Note that platform origin is not defined to coincide with the centre of gravity. Connection of a module to the platform is considered binary, meaning a module would be either connected or disconnected, and cannot be 'connected a little'. Any change in platform configuration would result in an instantaneous displacement of the centre of gravity. As continuity of estimated state is desirable, the use of the centre of gravity was avoided to use as the definition of $\{p\}$.

5.1.3. State Estimation

The platform state is but a concept to represent a collection of modules and is thus not directly measurable. It is however estimated by using a feedback signal of module positions and their known, constant placement within the body. Signal from a separate system performing state estimation of modules is translated to yield platform state estimates. Module localization can be done on board, on shore and with a variety of sensor systems, yet only needs to be consistently communicated on the robot network. Consider an update of the position and orientation of a module as

$$\eta^n_{b/n} = \begin{bmatrix} \mathbf{p}^n_{b/n} \\ \\ \Psi_{b/n} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^n_{b/n} \\ \\ \mathbf{y}^n_{b/n} \\ \\ \Psi_{b/n} \end{bmatrix} \tag{5.3}$$

The known position of modules with respect to platform coordinate system as described by equation 4.10 is used to transform module to platform pose by subsequent rotation and translation. For the three considered degrees of surface plane motion the platform state becomes

$$\eta^n_{p/n} = \begin{bmatrix} \mathbf{p}^n_{p/n} \\ \\ \Psi_{p/n} \end{bmatrix} \tag{5.4}$$

where

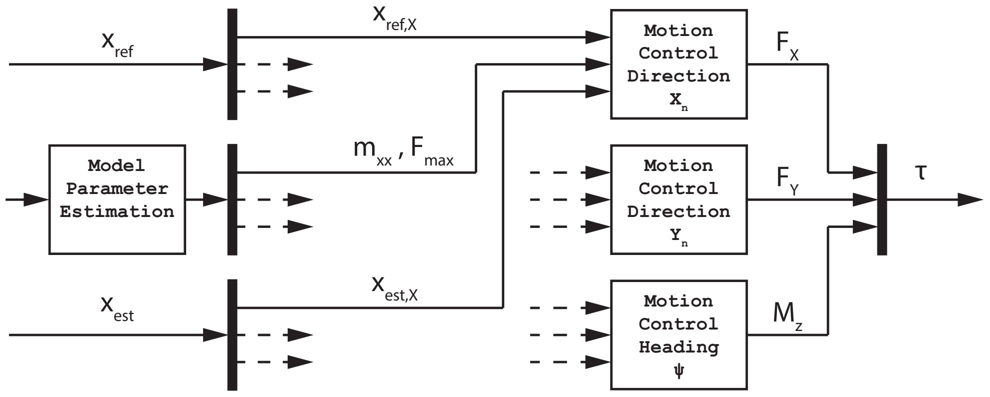
$$\Psi_{p/n} = \Psi_{b/n} - \Psi_{b/p} \tag{5.5}$$

and

$$\mathbf{p}^n_{p/n} = \mathbf{p}^n_{b/n} - \mathbf{p}^n_{b/p} = \mathbf{p}^n_{b/n} - \mathbf{R}(\Psi_{p/n})\mathbf{p}^p_{b/p} \tag{5.6}$$

5.1.4. Control Effort Generation

The control block is based on a Proportional-Integrator-Differential (PID) controller, designed to scale to model parameters which are estimated as described in section 4.2. Three parralel controllers are used to control each individual degree of freedom, as illustrated in 5.6



x_{ref}
$x_{ref,X}$
Model Parameter Estimation
m_{xx}, F_{max}
x_{est}
$x_{est,X}$
Motion Control Direction X_n
Motion Control Direction Y_n
Motion Control Heading ψ
F_X
F_Y
M_z
τ

Figure 5.6: Parralel controller setup

Control gains of the control-effort-generation block are designed to scale such that, once tuned for a single configuration, show similar behavior for any other configuration or platform size. For this, the following configuration dependent parameters are used.

- Position of the centre of gravity
- Linear inertia, or mass
- Angular inertia
- Maximum available control effort for translation (maximum force)
- Maximum available control effort for rotation (maximum torque)

The centre of mass of the platform is found using equation 4.40. It can then be substituted in equation 4.36 to express the equations of motion in that particular point. For three degrees of freedom in the surface plane this becomes shaped as

$$\mathbf{M}_p^{CG} = \begin{bmatrix} m_{xx} & m_{xy} & 0 \\ m_{yx} & m_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{5.7}$$

where

$$\mathbf{M}_{11} = \begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix} \qquad \mathbf{M}_{12} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{M}_{12} = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad \mathbf{M}_{22} = I_{zz}$$

Which shows the estimated moment of inertia I_{zz} in the bottomright corner. If modules have hydrodynamic added mass being modelled as a constant, direction-dependent constant, the off-diagonal elements m_{xy} and m_{yx} may be nonzero. This can also result that masses in xx and yy direction may be unequal, which can feel rather counter intuitive, as this is never the case with normal rigid body motion. The cause of this still originates from the origonal form of module inertia matrix, as this inherited by using such models.

The controllers responsible for linear motion will use an estimation of the mass of the platform. Rotating the inertial matrix can be done such that the off diagonal elements m_{xy} and m_{yx} become zero. The magnitude of the diagonal elements can be easily found, as they are the eigenvalues of M_{11}. The average of the eigenvalues is used as the estimated omni-directional mass for adapting controller behavior. For linear motion in 2 degrees of freedom (x and y) this becomes

$$m_p \approx \frac{1}{2}\sum Eig(\mathbf{M}_{11}) \tag{5.8}$$

As the fleet utilizes rotatable azimuth thrusters, the maximum force is generated by the propellers on full power in a single direction. The maximum force that the platform can generate can be found by summation of maximum thruster force of all thrusters

$$\mathbf{f}_{p,max} = \sum_{i=1}^{n_{thr}} f_{i,max} \tag{5.9}$$

where n_{thr} refers to the total amount of thrusters, and $f_{i,max}$ is the maximum force that the ith propeller can supply. The homogeneous fleet has two identical propellers per module, such that.

$$\mathbf{f}_{p,max} = 2 * n * f_{prop,max} \tag{5.10}$$

Maximum torque is generated when all thrusters supply maximum force in the direction perpendicular to a vector between CG and the thruster. For a single vessel, this becomes

$$\mathbf{m}_{i,max} = |\mathbf{r}| f_{i,max} = |\mathbf{p}^p_{CG/p} - \mathbf{p}^p_{thr,i/p}| f_{i,max} \tag{5.11}$$

where $\mathbf{p}^p_{CG/p}$ is the position vector of the platform CG and $\mathbf{p}^p_{thr,i/p}$ is the position of the ith thruster. The latter is usually given in local frame of a module, but can be converted to platform coordinates by matrix rotation and a translation as:

$$\mathbf{p}^p_{thr,i/p} = \mathbf{R}^p_{bj}\mathbf{p}^{bj}_{thr,i/bj} + \mathbf{p}^p_{bj/p} \tag{5.12}$$

where $\mathbf{p}^{bj}_{thr,i/bj}$ is the position of thruster i, mounted on module j, expressed in the body fixed coordinate system of module j. $\mathbf{p}^p_{bj/p}$ is the position of module j expressed in platform coordinate system. Summation of equation 5.11 over all modules yields total maximum torque generated by actuators for a given configuration as

$$\mathbf{m}_{p,max} = \sum_{i=1}^{2n} \mathbf{m}_{i,max} \tag{5.13}$$

It should be noted that the equation 5.10 and 5.13 show absolute maxima, which need full participation of all actuators to be reached. These maxima can not be obtained in different degrees of motion simultaneously, as outputs will be saturated. To avoid unpredictable control effort generation, actuator operation near output saturation is avoided during implementation.

The control blocks for each degree of freedom, as shown in figure 5.6 operate based on PID control. Generic PID control starts by computing the state error by taking the difference between reference and state feedback. This error is fed parralel trough three different gains, where one signal is integrated and one is differentiated, until the signals are summed to yield the control output.

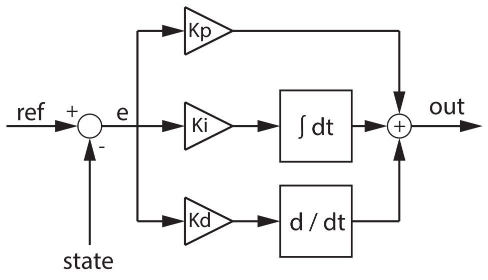


Figure 5.7: A generic Proportional-Integral-Derivative control structure.

The control gains of the three parralel PID controllers are tuned to a single reference configuration. As the configuration changes the control gains will adapt to the newly estimated dynamics. The control gain scaling is based on the assumption that a configuration will have a response comparable to the reference, but in a different time scale. Typical errors that are fed into PID gains will thus be of comparable magnitude. System responses of current and reference configurations are thus aimed to be approximately comparable such that

$$\eta_c(t) \approx \eta_{ref}(C * t) \tag{5.14}$$

If control forces are a dominant factor to the response time of a step input on the system, a characteristic accelleration of a configuration can be defined as

$$a = \frac{Force}{Inertia} \tag{5.15}$$

This is used to estimate the time scaling factor is estimated as the ratio of maximum accelleration of a configuration with respect to the reference configuration.

$$C_c = \frac{a_c}{a_{ref}} = \frac{F_c\, I_{ref}}{F_{ref}\, I_c} \tag{5.16}$$

where a_c and a_{ref} are the characteristic accellerations of the current and reference configuration respectively.

All gains is designed to scale to the maximum obtainable control effort. The base value of contribution is determined in the gain tuning process of the reference configuration. The eventual output of any proportional integral and derivative gains is multiplied by the maximum control effort in that dimention.

Output of the control effort from proportional gain scales to the magnitude of the error, which assumed comparable in all configurations. This could result in, for example, a proportional gain that is to contribute 60% of the maximum obtainable control effort at an error of $e = 1.0$. The gain would become

$$K_p = K_{p,base} * \tau_{max} = 0.6 * \tau_{max} \tag{5.17}$$

such that the control effort contributed by the proportional block becomes

$$\tau_{i,prop} = e * K_p = 0.6 * \tau_{max} \tag{5.18}$$

Integral control is however affected by the time in which the system responds. A system that responds slower ($C_c < 1$) than the reference configuration will encounter additional integrator buildup. Time factor C_c compensates for change of integrator output due to response time by adjusting integral gain as

$$K_i = C_c\, K_{p,base}\, \tau_{max} \tag{5.19}$$

such that integral control output becomes

$$\tau_{i,int} = K_i \int_0^t e\, dt = C_c\, K_{p,base}\, \tau_{max} \int_0^t edt \tag{5.20}$$

Derivative control output is also affected by time, but scales inversely to time factor C_c with respect to integral control. Imagine, for example, a mass (*cough* *cough* vessel) that approaches the reference state, which would make the time derivative of the error negative. Derivative control would attempt to slow the mass down as it approaches it's desired state to avoid overshoot. An object, such as a container vessel, with low maximum control forces relative to the large mass would have to use take this speed more serious than highly manouverable vessels. Derivative adapts to a configuration as

$$K_i = \frac{K_{d,base}\, \tau_{max}}{C_c} \tag{5.21}$$

Base values of controller gains ($K_{p,base}$,$K_{i,base}$,$K_{b,base}$) will be set while revieuwing responses of the system in reference configuration. A PID controller can be manually tuned, or with the help of many tools such as automated PID tuning software. Linear motion in x and y direction will have identical control settings, as dependency on the orientation of reference frame $\{n\}$ is considered undesirable.

5.1.5. Control Effort Allocation

Control effort, as shown in the previous section, needs to be allocated onto the platform's actuators. The amount of thrusters available varies per configuration. Also placement and orientation of actuators can differ. A platform needs to be able to sufficiently control it's motion in all reasonably forseeable configurations. As the amount of different configurations is rather large, a general solution is used, that can solve the control effort allocation problem for all possible configurations. The designed control effort allocation protocol relies on the following main principle: "The contribution of an actuator to a desired resulting force or moment is proportional to its ability to contribute relative to that of the combined set of actuators."

This principle manifests particularly in rotational motion, as the ability of a thruster to generate torque relies on its placement with respect to the centre of gravity. Linear motion turns out not to exhibit such dependencies, as all thrusters are equal in strength, and possible orientation. To compute actuator commands that satisfy the desired control effort, it is allocated in each degree of freedom individually and finally combined.

Position of a thruster with respect to CG can be expressed in $\{p\}$ by

$$\begin{aligned}\mathbf{p}^p_{t/CG} &= \mathbf{p}^p_{t/b} + \mathbf{p}^p_{b/p} - \mathbf{p}^p_{CG/p} \\ &= \mathbf{R}^p_b \mathbf{p}^p_{t/b} + \mathbf{p}^p_{b/p} - \mathbf{p}^p_{CG/p}\end{aligned} \tag{5.22}$$

Thruster force vector can be expressed in $\{p\}$ in 3 degrees of freedom as

$$\mathbf{f}^p_t = \mathbf{R}^p_b \mathbf{f}^p_t \tag{5.23}$$

The total resultant control effort from all modules in the centre of mass can be found by

$$m_{CG} = \sum_{i=1}^{n_{thrusters}} \mathbf{p}_{ti/cg} \times \mathbf{f}_{ti} \tag{5.24}$$

$$f_{CG} = \sum_{i=1}^{n_{thrusters}} \mathbf{f}_{ti} \tag{5.25}$$

or in vector form as

$$\tau_{CG} = \sum_{i=1}^{n_{thrusters}} \mathbf{H}^\top(\mathbf{p}^p_{t/CG}) \begin{bmatrix} \mathbf{f}^p_{ti} \\ 0_{1x3} \end{bmatrix} \tag{5.26}$$

where $\mathbf{p}_{ti/CG}$ is the position of the ith thruster with respect to the platform's centre of gravity, and $\mathbf{f}_{ti}$ refers to the force vector applied by the ith thruster. Where it should be noticed that the zeros represent torque applied by the propeller, as propellers are modelled as a forcevector applied in a point. A resulting moment can be created due to the fact that thrusters are placed at a distance from CG.

The approach on generating control effort that results into torque is as follows. A force at a distance from CG creates a higher resulting torque. The linear relation between torque and distance between applied point and CG can be seen in equation 5.24. Only considering forces in the xy plane gives

$$m_{zz} = x_{ti/CG}\, f_y - y_{ti/CG}\, f_x \tag{5.27}$$

So at a constant force, the generated moment proportionally increases with distance. For a single direction (x or y) this would be shaped as shown in figure 5.8. Thruster contibution to torque that is to be deployed proportional to the effectiveness of the thruster results in a quadratic shape. Figure 5.9 shows how the quadratic contribution of a thruster at varying distance

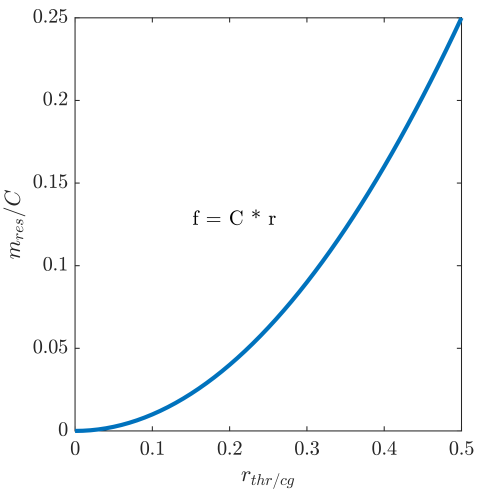


Figure 5.8: Linear relation between resulting moment and thruster distance from CG, while thruster force is constant

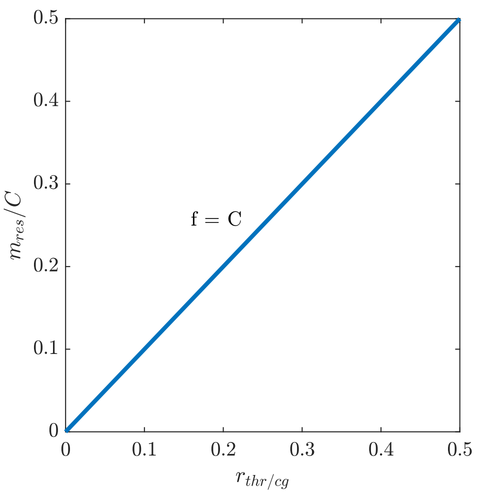


Figure 5.9: Quadratic relation between resulting moment and thruster distance from CG, due to a thruster force being controlled to be proportional to distance

Thruster force to contribute to realizing a desired resulting moment is set as

$$f_x = C_m\, m_d\, (-y_{ti/CG}) \tag{5.28}$$

$$f_y = C_m\, m_d\, x_{ti/CG} \tag{5.29}$$

where C_m is a configuration dependent participation factor for torque and m_d is the desired resulting moment of the combined structure. The resulting moment of a thruster becomes (by substituting 5.28 and 5.29 in 5.27)

$$m_{ti} = C_m\, m_d\, [{x_{ti/CG}}^2 + {y_{ti/CG}}^2] \tag{5.30}$$

The resulting moment of all thrusters satisfies

$$\begin{aligned} m_{res} = m_d &= \sum_{i=1}^{n_{thrusters}} m_{ti} \\ = C_m\, m_d &\sum_{i=1}^{n_{thrusters}} [{x_{ti/CG}}^2 + {y_{ti/CG}}^2] \end{aligned} \tag{5.31}$$

From which the total configuration dependent participation factor can be found, as

$$C_m = \frac{1}{\sum_{i=1}^{n_{thrusters}} [{x_{ti/CG}}^2 + {y_{ti/CG}}^2]} \tag{5.32}$$

This constant ends up scaling all thruster contribution such that the overall relationship between thruster contribution and distance is quadratic, and that the resultant moment matches the desired moment.

For linear motion a similar principle of contribution to effectiveness is applied, yet turns out much simpler. The homogeneous fleet has thrusters that are all equal in strength, and able to turn in all directions. As all thrusters are thus equally able to contribute to linear motion, total desired thrust is equally divided between all thrusters such that

$$f_d = \sum_{i=1}^{n_{thrusters}} * f_{ti} \tag{5.33}$$

$$f_{ti} = C_f\, f_d\, f_{max} \tag{5.34}$$

$$C_f = \frac{1}{n_{thrusters}} \tag{5.35}$$

All degrees of freedom that are affected by elements from the vector of desired control effort are evaluated independently, resulting in a forcevector for every thruster for every degree of freedom. The forcevectors pertaining to a thruster in all degrees of freedom are summed to obtain the overall contribution of that thruster. For three degrees of freedom this becomes

$$\mathbf{f}_{ti} = \mathbf{f}_{ti,fx} + \mathbf{f}_{ti,fy} + \mathbf{f}_{ti,m} \tag{5.36}$$

Fig. 5.10 illustrates how the control allocation problem is solved by combining actuator responses from elements of the desired control effort vector.

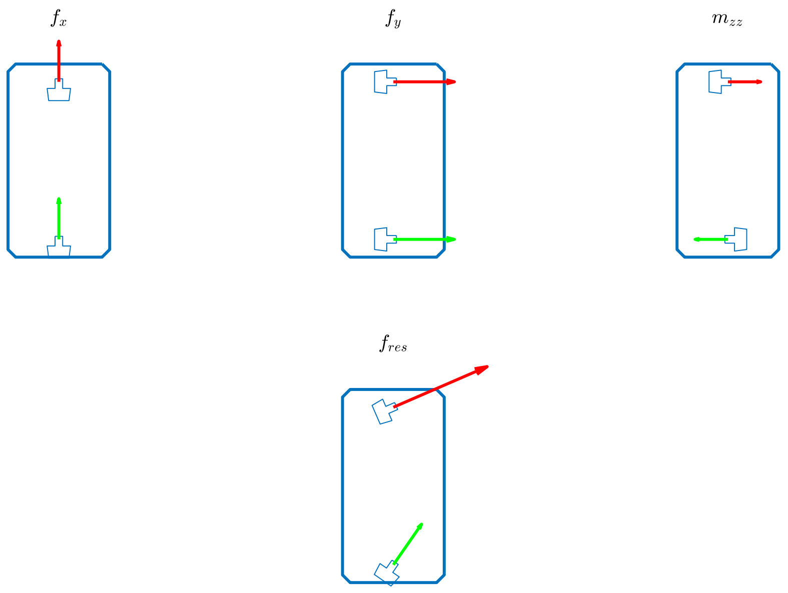


Figure 5.10: The approach of solving control allocation in each dimention separately, illustrated for a single vessel configuration. Three dimensions along which control effort is required are solved (above). The forces of the three solutions are combined to yield the final solution (below).

5.1.6. Assembly Protocol

Modules will assemble in a lattice structure, using active magnets to remain configured. Task execution, including assembly, operates in a phasewise fashion. Generating the desired configuration (blueprint) and assembly planning are done by the operator for this project. The following steps are considered to achieve assembly of a module or platform to another:

- The connecting object lines up with the assembly at a short distance, such that it can freely reach the connection site. During this process both objects are controlled by different controllers.
- The connecting object moves to the connection site. Magnet connectors within area of acceptance of a connection site make contact, and connect.
- Success of connectivity is evaluated before continuing
- If connected, ownership of the connecting bodies is transferred to the controller of the assembly. The other controller becomes inactive. (see figure 5.3 and 5.4)
- The controller of the assembly recomputes configuration dependent parameters on:
  - The estimated platform model, as in section 4.2
  - The control effort generation protocol, by adapting controller gains to the newfound model estimate, as in section 5.1.4

  - The control allocation protocol, to divide generated control effort over the new configuration as described in section 5.1.5.

- The assembly runs the adapted control scheme for further tasks.

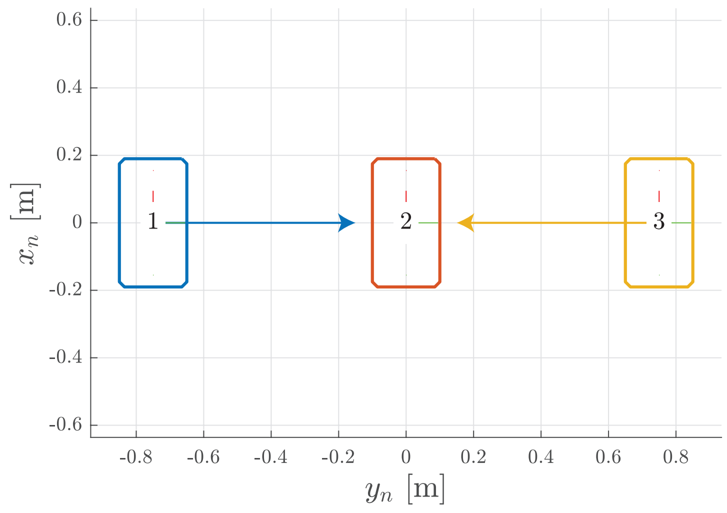


Figure 5.11: Platform assembly protocol from top vieuw. Vessel 1 and 3 are lined up to approach connecting to vessel 2 on either side. Once connectors are within range, the magnets snap into place, fixing relative motion.

5.2. Implementation

The conceptual design elements that are explained in the previous section have been implemented to represent the envisioned framework. This section illustrates how designed concepts are realized in various hardware and software components.

Physical system components are first introduced to shed light on the experimental facility, localization system and give more details on low level actuator control systems running on the modules. The multi-robot network setup is explained showing how various system components communicate. A object oriented Matlab software-framework been developed aimed to be reusable and intererable for other multi-vessel experiments within RAS. This framework, its structure, relation between classes, and means of implementing the designed control approach are explained in section 5.2.3.

Throughout developments many iterative changes were made to the system, of which two facets are discussed in section 5.2.4 starting with the process of tuning control gains and secondly an improvement of the assembly protocol using normal forces between modules to aid connectors reaching area of acceptance.

Behaviors and responses of the system in development are occasionaly shown to support design choices. Behavior of the system in final stage of development is shown and evaluated in the next chapter.

5.2.1. System Components

The implemented control system consists of three key components

- An optical tracking system for module localization.

- A computer that executes the control protocol and distributes generated tasks to the entire fleet.

- A set of Delfia-1* modules

This section provides details on the experimental setup of the optical tracking system and modules. Functionality of the control software is discussed in a separate section (5.2.3).

The experimental setup is designed to operate in the towing tank facility of section Maritime and Transport Technology (section MTT) in the faculty of Mechanical, Maritime and Materials Engineering (Faculty 3ME). One of such tanks is equipped with an optical sensing and interpretation system from the brand Optitrack. A 40 meter section of the tank is covered by cameras. Optitrack's motion capture software supports defining a set of infra-red reflectors as a rigid body. The camera setup was set up to interpret images to estimate state of the defined bodies (modules), and broadcast them on dedicated ROS topics at a frequency of 30hz.

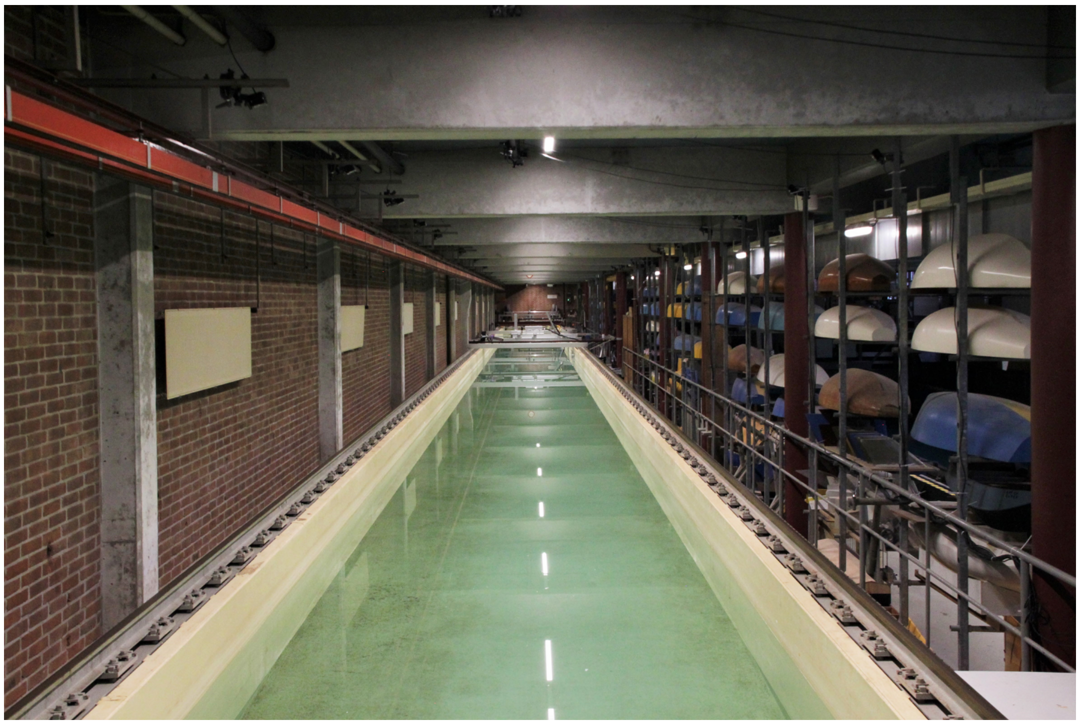


Figure 5.12: The MTT towing tank facility at 3ME.

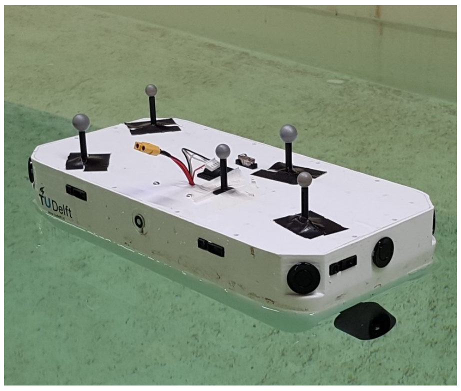


Figure 5.13: Five infra-red reflectors on top of a Delfia-1* module.

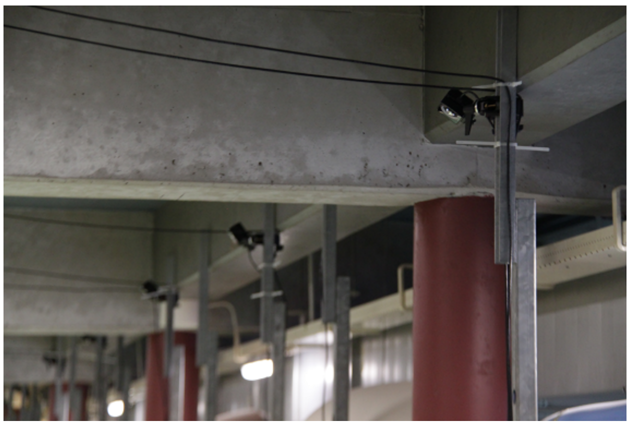


Figure 5.14: Motion tracking cameras mounted along the experimental setup.

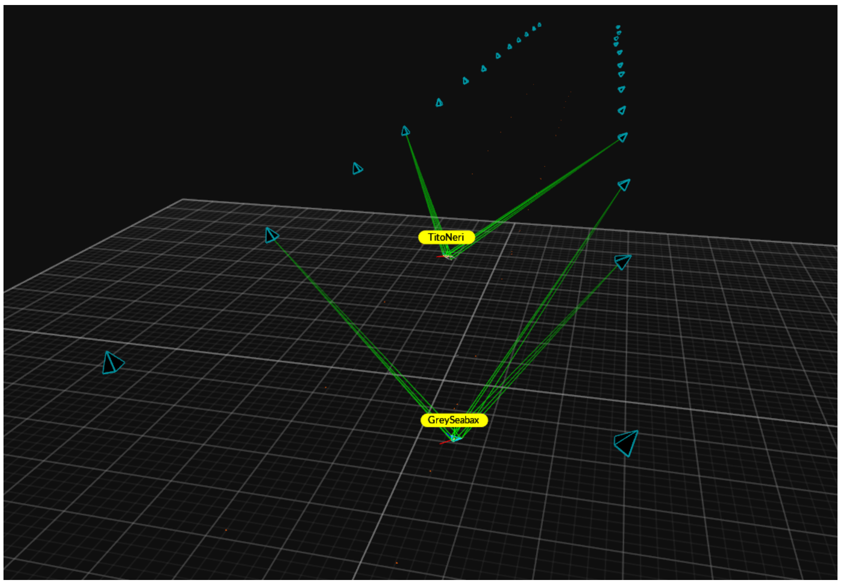


Figure 5.15: Optitrack motion tracking with the towing tank setup, showing two deployed vessels. Green lines depict rays of reflectors that are succesfully being tracked

The Delfia-1* vessel (also refferd to as 'Delfia') is a model scale, electrically powered ship, equipped with two 360 degree rotatable azimuth thrusters. There are a total of four main actuators that are controlled by an on board system, referred to as the 'low-level control system'. The two azimuth thrusters are identical, and both have their orientation and propeller speed managed by a single microprocessor of the open-source Arduino project. Control of propeller speed and thruster angle is done with different hardware components, yet both rely on similar PID feedback control principles. Figure 5.16 illustrates the states that the low level control system needs to manage for one thruster.

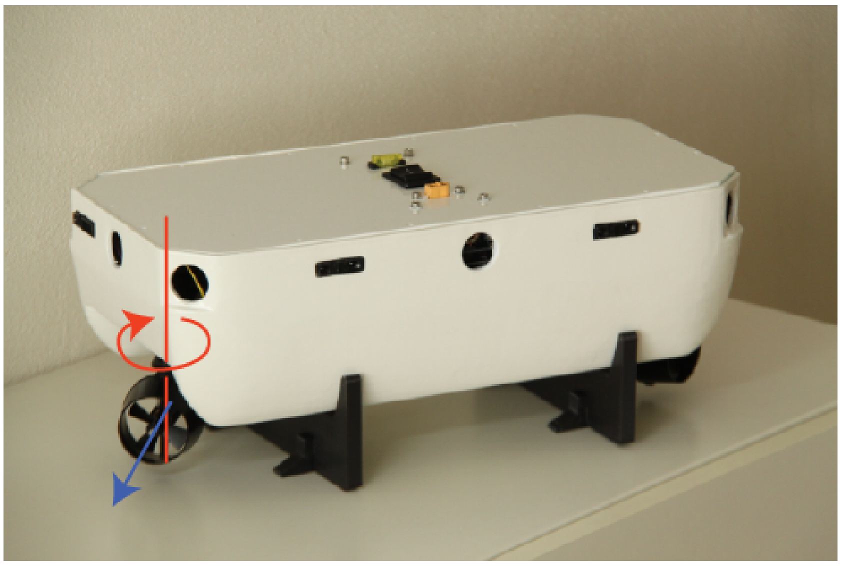


Figure 5.16: Degrees of freedom of one azimuth thruster. Each thruster (front and back) needs its direction (red) and propeller speed (blue) controlled.

All decisions for the low level control system are made on the on board microprocessor. This controller responds to actuator reference commands via USB serial port connection. Connectivity of the low level control system with other elements in the network is done through a Raspberry-Pi, as illustrated in figure 5.17. The Raspberry-pi is the connecting element to the ROS network (via wifi) and the Arduino (via USB serial protocol). Other solutions exist that provide similar connectivity, yet a Raspberry-pi has some processing power which allows distribution of some, or all control tasks in future projects.

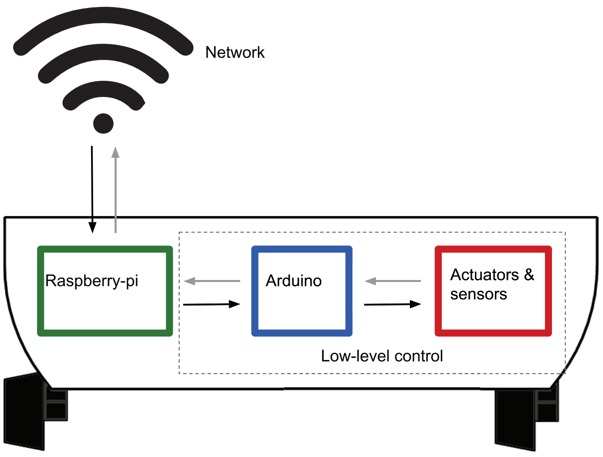


Figure 5.17: The interaction of the Delfia's low level control system with on board computer (Raspberry-pi) and the vessel network.

The propeller is powered with an electrical 12V DC engine, while a sensor signal on speed is created by means of an optical encoder. Figure 5.18 shows the propeller drivetrain that can be found above each thruster, and figure 5.19 illustrates the feedback loop that controls propeller velocity.

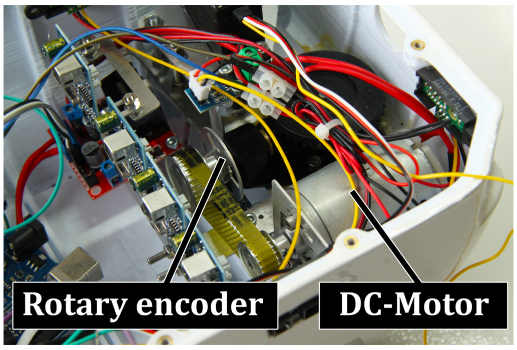


Figure 5.18: The propeller drivetrain. The rotation in the axle of the DC-motor is transferred to the propeller via a belt and gears.

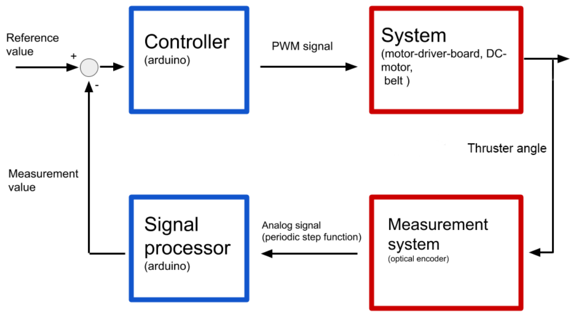


Figure 5.19: The feedback loop of the low level propeller-speed control system

Each thruster angle is actuated by a "Parallax Feedback 360° High-Speed Servo". Thruster and servo are connected via a rubber cog belt (figure 5.20). The servo needs a 50Hz PWM signal as input and uses a hall effect sensor to provide angular position feedback as a 910Hz PWM signal. The feedback loop that controls the thruster angles is as shown in figure 5.21.

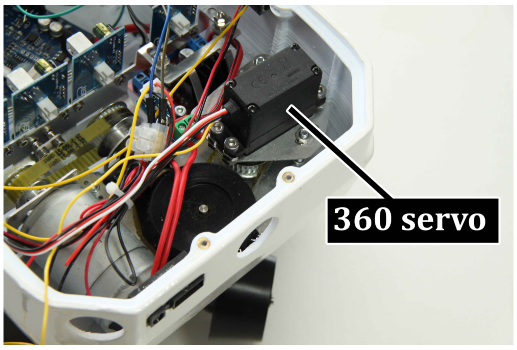


Figure 5.20: The propeller drivetrain. The rotation in the axle of the DC-motor is transferred to the propeller via a belt and gears.

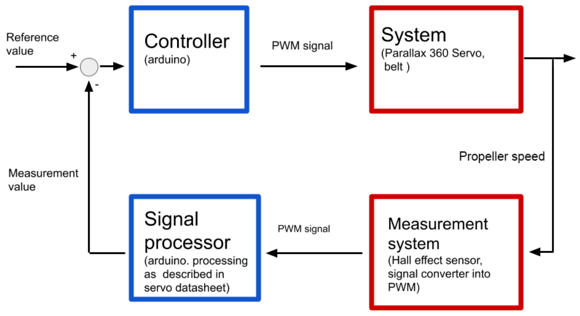


Figure 5.21: The feedback loop of the low level thruster angle control system

Fundamentals of the Delfia-1* low level control system have been developed during a research project

Boogmans [5], by the same author as this work. Development of hardware and software is described therein, and parameter tuning is done by means of evaluating reference step responses. Characteristic control system behavior is shown in figures 5.22 and 5.23 for propeller-speed and thruster-angle respectively. Note how the propeller speed signal contains significant fluctuations, of which the majority is considered sensor noise. An optimum trade-off point of signal filtering was found between smoothness and increased latency. High frequency notes in this signal (order of 10^3hz) that translate to engine control effort will not result in noticable differences of vessel dynamics response, as the vessel responds in much lower frequencies (order of 10^0 hz).

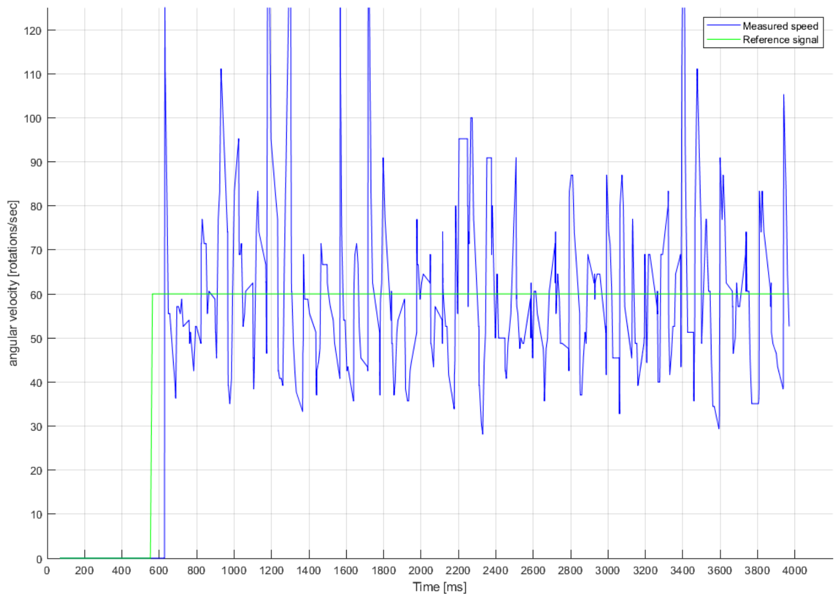


Figure 5.22: Measured signal of a characteristic step-response of the propeller-speed control system. Considerable high frequency notes can be observed, although it oscillates rather well around the reference input.

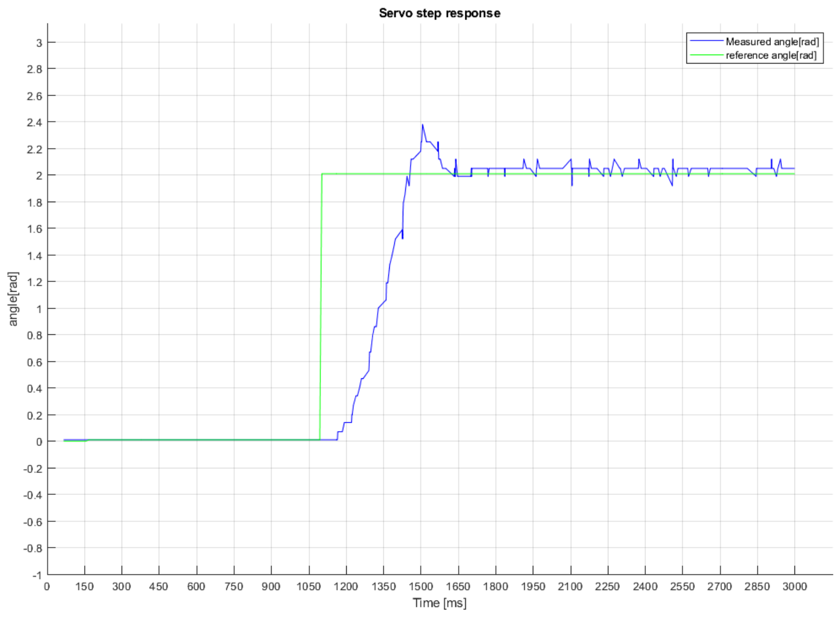


Figure 5.23: Measured signal of a characteristic step-response of the thruster-angle control system.

It is worth noting that the slope of the servo step response, as shown in figure 5.23, proved quite consistent with more agressive controller gains. Higher proportional or integral gain caused more overshoot, or even instability, without the benefit of a quicker response time. This is considered simply due to the maximum rotation speed of the servo.

Actuators that connect the modules have been developed and implemented as a set of active magnets, or solenoids, shown in figure 5.24 and 5.25. Models are available in various lifting capacities, and a $10N$ model that operates at $12V$ was estimated to provide enough force for vessels to remain connected in reasonably expected scenarios. It is powered by the $12V$ micro powergrid from the on-board battery. Higher powered magnets draw more power, and this variant seemed a good balance at a $1.0W$ power consumption. Mounts for the connectors were developed and fabricated by a 3D printer to fit in existing slots in the Delfia's hull.

At first, connections were implemented where a connection was made between two vessels with connectors of opposite polarity. Magnet connector polarity can be reversed by reversing the potential on coil connectors. This can be automated with, for instance, an H-bridge. Tests proved connector force to be insufficient between connecting solenoids. Attracting forces to a piece of iron did give the desired results, thus connector counterparts were lathed to size. Available 3D printers proved to have tolerances sufficient such that connector parts, both magnet and piece of iron, were made to stay in the mouts by using a press.

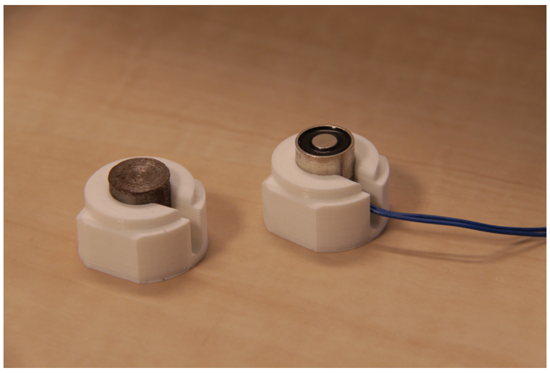


Figure 5.24: An active (right) and passive (left) magnetic connector in a (white) 3D-printed hull mount.

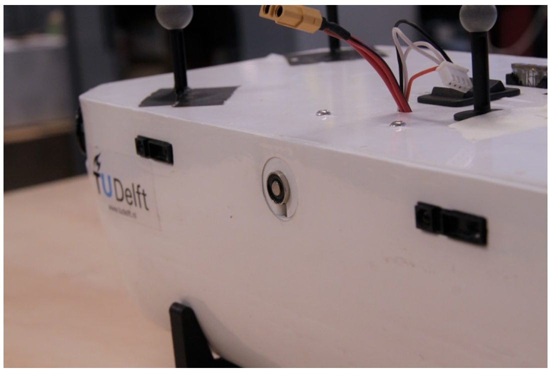


Figure 5.25: Active magnet mounted in the hull of a Delfia-1* module to connect modules into a rigid platform.

5.2.2. Network setup

The backbone of the communication system is deployment of ROS over an internet network. Vessels connect over WiFi, while shore facilities connect to the network router through ethernet cables. ROS provides various standardized messagetypes to improve interoperability and modularity. Various conventions that are developed throughout this project are intended to serve as a framework for further development and reasearch in the facilities of Researchlab Autonomous Shipping. The proposed standardization includes:

- ROS message types for control related signals
- Naming convention for control related ROS-topics
- IP adress reservations for vessels and other components

Table 5.2 shows the proposed conventions for communication over ROS in RAS facilities.

Topic content	Messagetype	Naming convention
Vessel position and orientation	$geometry\_msgs/PoseStamped$	$/vrpn\_client\_node/ < vesselname > /pose$
Vessel Actuation	$std\_msgs/Float64MultiArray$	$/actuation < vesselname >$
Controller reference (for surface plane dynamic positioning systems)	$geometry\_msgs/Pose2D$	$/reference < controllername >$

Table 5.2: Proposed conventions for communication over ROS in facilities of Researchlab Autonomous Shipping.

The module state estimation system, consisting of a set of camera's and a device that interprets the image data, streams its poses of each module on the dedicated topics. The fleet control system uses this information to make decisions for a set of actuators, and broadcast it. Actuators throughout the fleet respond to such updates, closing the control loop. It is worth noting that the shown division of tasks leaves the system rather modular. Systems performing control and state estimation can be replaced, without the need of adapting other components given that these communication protocols are followed. The location and hardware on which a task is performed also becomes flexible. Furthermore, tasks can be divided, distributed, or centralized at the convenience of the designer, without the need of drastic system changes. Localization has been centralized, as an on-shore motion tracking system provides higher accuracy with respect to alternatives that perform localization on board (e.g. leidar, radar, GPS, IMU, camera image localization). Control decisions are centralized, as an on-shore computer that has access to computational power many times higher than can be feasibly realized on board of the model scale vessels.

The trend in this minimalistic approach is to not perform tasks on a ship if it is not nessecary to do so. This does have some effects to be considered. Firstly, the system becomes reliant on network performance and availability. Advancements in telecommunication industry and technology led to options for increased connectivity between devices. To name only some examples; rural areas are commonly covered with a 4g internet network, and devices creating local (wireless) networks have become widespread and affordable. Secondly, simplicity of modules benefits scaling to a higher amount of vessels. Less components will be present that can cause malfunction or requirements for maintenance.

To connect a module to the network, an on-boar Raspberry-pi provides the link between ROS, over wifi, and the arduino, over serial USB protocol. A standardized operating system image of Ubuntu-mate has been developed to be cloned on all Raspberry-pi's on the Delfia fleet. This has been done with the goal of needing to configure as little as possible to the Raspberry-pi's settings after cloning. Python scripts running on the device uses a single parameter (vessel number) to operate properly and subscribe to the correct ROS topics following naming convention as shown in table 5.2. Rapidly setting up on-board Raspberry-pi's is valued so highly, as future changes that need to be applied on a large number of devices will become more laboursome.

5.2.3. Control Software

Control decisions are made on a single device, where scripts are developed using Matlab as programming language. This software interacts with the network only through ROS, which means that it could have been developed in other languages as well thanks to the interoperability of ROS. This choice comes down to a matter of the developer's preference, and other options such as Python can be suitable as well.

In an effort to develop a matlab fleet-control-framework that is understandable, scalable and re-usable in future projects, an object-oriented structure is formed. Main functionality of the framework that forms the key elements of the feedback control loop are described further in this section. Sourcecode can be found in appendix C. Classes that make up the framework can be shortly described as follows:

- A **Vessel** superclass provides basic traits and methods that are relvant for managing a single ship in 3 degrees of freedom on the surface plane. Various vessel types, such as the Delfia, Tito-Neri and Grey-Seabax types from the RAS-fleet, can inherit common characteristics and methods from this class. Some examples of key parameters are: state (positions and velocities), mass, vessel width and length.

- The **Delfia** object class, subclass of Vessel, contains information and methods unique to the Delfia-1* robot. It is distinguished by various aspects such as: actuator setup with two azimuth thrusters, actuator-model, and methods that aid the standardized interaction through the ROS network.

- The **Platform Controller** object class stores information and provides methods required on a platform level. This object's main function is running the feedback control loop for the platform consisting of multiple vessels. To do so, it contains and uses methods for platform-model estimation, control-system adaptation, platform-state estimation, control effort generation, control effort allocation and communication over ROS.

- The **Fleet Manager** object class acts as an overarching entity that dictates tasks of controllers in a phase-wise fashion. It provides control objectives for the platform controllers to use as a reference, and allocates ownership of a module to the platform controller to which a module is assembled.

A control loop revolves primarily around a Platform Controller (referred to as 'controller' in this section), performing tasks of which the design is described in section 5.1. Modules, represented as the Delfia class can be connected to- or disconnected from a platform trough the $attacgBody()$ and $detachBody()$ functions. This automatically triggers protocols that adjust functionality of the control loop to it's new configuration. Such functions are:

- Estimating the platform's centre of gravity
- Forming an estimate of the platform's dynamical model
- Adjusting gains of PID controllers that generate control effort
- Adjusting parameters affecting distribution of control effort between actuators.

The control loop runs event-driven, responding on position updates of the first connected module. As the optical motion tracking system publishes a module's pose, callbacks in the respected Delfia object update the position, and the platform initates it's control loop iteration which runs the following key steps

- The platform starts by estimating it's current state from the updated module position and the known configuration of that module with respect to the platform.
- The new state is fed alongside the reference to the parralel PID controllers. These three conrollers output desired control effort for all dimentions. This thus results in two desired forces along the platform's local x and y axis, and a desired resulting moment around z.
- The desired control effort is then allocated between the actuators of all connected modules. Allocation occurs by individually allocating each force and moment, which is then combined into a solution that satisfies all control efforts simultaneously.
- Finally, allocated control efforts (which are forces for each thruster in x and y direction) are translated to actuator commands (thruster angle and propeller speed). These commands are then published on ROS-topics dedicated to actuation for each respective module, such that they are executed by the physical vessel.

5.2.4. Modifications throughout development

The overall fleet control system underwent many iterative changes of which some key considerations are discussed here. First, the process of developing control gain tuning is discussed. Secondly, changing approaches for succesful assembly are discussed. Varying responses from systems with different settings have shown behaviors that are interesting to analyze. Off course not all responses show behavior that is particularly interesting, so only a selection is discussed.

As mentioned in section 5.1, control parameter tuning is done according to a particular reference configuration. The chosen reference configuration is a single module body. Step responses of movement in the three degrees of freedom (x,y,yaw) were evaluated to aid decision making for optimizing control behavior. The key measures of performance that are optimized throughout iterations are rise-time, settlingtime and overshoot. Amplitude of step inputs are based on a characteristic motion for a platform performing a logistical task. Translational movement is assumed to be a short distance, in the order of several times the length of the platform. This lead to the choice of amplitude in x and y reference of 1.0 and 0.5 meter. A characteristic rotation is set to a 90 degree turn.

PID control gains are developed with the following approach [39]

- Add proportional control to improve the rise time
- Add a derivative control to reduce the overshoot
- Add an integral control to reduce the steady-state error
- Iteratively adjust each gain to improve the overall response

Effects of control gains on system performance indicators can generally be described as in table 5.3.

Table 5.3: Effects of independent P, I, and D tuning Ang et al. [1]

Closed-Loop Response	**Rise Time**	**Overshoot**	**Settling Time**	**Steady-State Error**	**Stability**
Increasing Kp	Decrease	Increase	Small Increase	Decrease	Degrade
Increasing Ki	Small Decrease	Increase	Increase	Large Decrease	Degrade
Increasing Kd	Small Decrease	Decrease	Decrease	Minor Change	Improve

Initial proportional gains were chosen based on characteristic initial error, maximum control effort and desired fraction of utilization of maximum control effort at the step time. Initial error is assumed equal to the step amplitude, and maximum control effort for single vessel operation could be calculated from the known thruster setup. Table 5.4 shows units and order of magnitude of forces, errors and rise times.

Table 5.4: Maximum control effort of a single Delfia-1* vessel

System parameter	Order of magnitude	Unit
Max. control effort: translation - Force	fmax = 0.45	[N]
Max. control effort: rotation - Torque	mmax = 0.06	[Nm]
Step amplitude: translation	1.0	[m]
Step amplitude: rotation	$\frac{1}{2}\pi$	[rad]
Rise time: translation (amp = 1.0 m)	6.0	[s]
Rise time: rotation (amp = $\frac{1}{2}\pi$ rad)	2.0	[s]

Step responses for single vessel operation has been collected from three simultaneous, but independently operating modules. Figure 5.26 illustrates how such step responses were executed. The step in reference occurs along the x_n direction, after the system settled to it's initial reference. A 1.0 m translational step occurs in x_n around $t \approx 0s$. A pi/2 rad rotational step occurs at $t \approx 110s$. A 0.5m translational step occurs in x_n around $t \approx 210s$. The translational steps are both in global x coordinate, yet they represent 'forward' and 'sideways' motion as the platform rotated in between. All modules respond fairly identical, as can be observed from the similarities of vessel response signals in x and Ψ dimentions. It is worth noting how some coupling between degrees of freedom is visible, as a step in one dimention result in some, albeit small, distortions in the others.

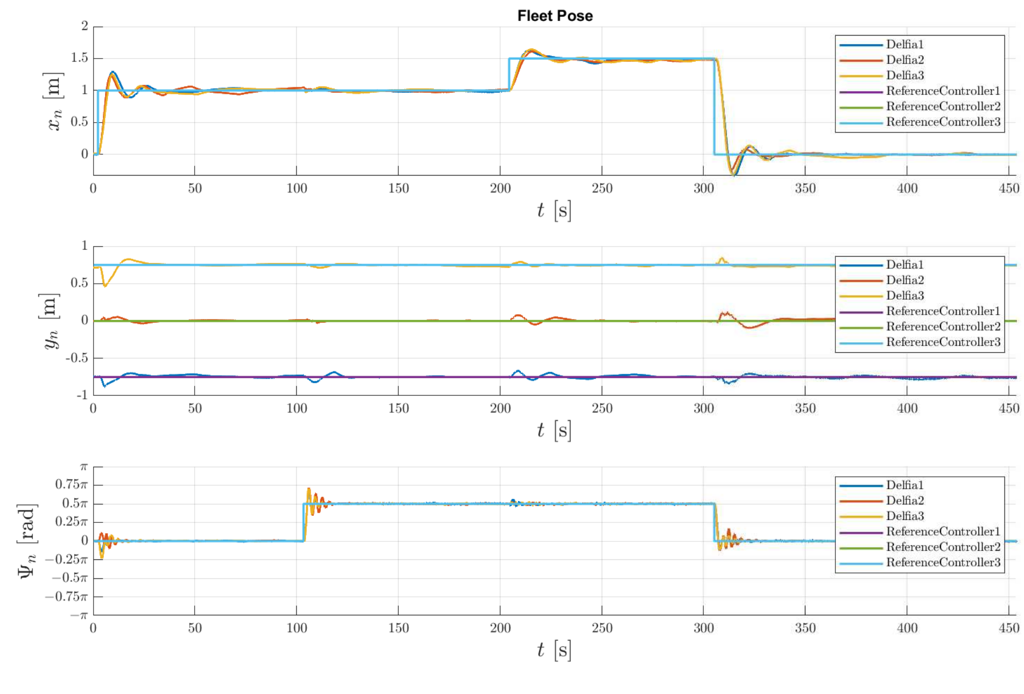


Figure 5.26: 3 separate vessels doing the similar step responses, but at an y-offset, such that they are besides oneanother. Three different step responses are done in this test representing forward, rotational and sideways motion.

Throughout the process of iteratively improving control performance by adjusting gains, various insights were gained. A steady state error for all step responses was neglectably small without utilizing any integral control. This can be explained by the fact that the test setup was in a lab setting with little to no disturbances such as wind, current or waves. Thus there was no nessecity of setting any integral gain. Dampening that was naturally present on the vessels proved to be quite significant without addition of derivative control. This is obviously thanks to hydrodynamic dampening. Enough dampening was considered to be naturally present in translation, and addition of more dampening terms only decreased performance. However, rotational motion did benefit from derivative control, as way more oscillation could be noticed in the responses. Magnitude of derivative gain for rotational motion formed as a balance between rise-time and settling-time.

Hence the control gain tuning process formed control gains from the reference configuration. Following the control-gain scaling approach as described in section 5.1.4, the configuration-independent control gains were set as follows

Dimention	Proportional gain	Integrator gain	Derivative gain
translation x and y	1.0 $[m^{-1}]$	0 $[s * m^{-1}]$	0 $[s^{-1}m^{-1}]$
rotation Ψ	0.4456 $[rad^{-1}]$	0 $[s * rad^{-1}]$	0.15 $[s^{-1}rad^{-1}]$

Table 5.5: Configuration independent control gains that resulted from iterative system response evaluation and tuning.

The initial assembly protocol has been slightly adjusted throughout initial experiments. Early tests on platform assembly proved unable to perform within a reasonable timeframe. This was caused due to the fact that the magnet connectors did not reliably come into the area of acceptance upon which they would connect. Magnets seemed an attractive solution to connecting, especially as there was not a required approach direction during connecting, but vessels simply had to be adequately close to oneanother. Practically, modules would indeed be positioned in assembly orientation, yet oscillate a little, and not properly coming into contact. The area of acceptance proved too small for timely assembly with the current approach. The difficulties during connecting are visible in one test in particular, for which the module references and responses are shown in figure 5.27. During this test the assembly protocol ran as described in section 5.1.6 and illustrated in figure 5.11.

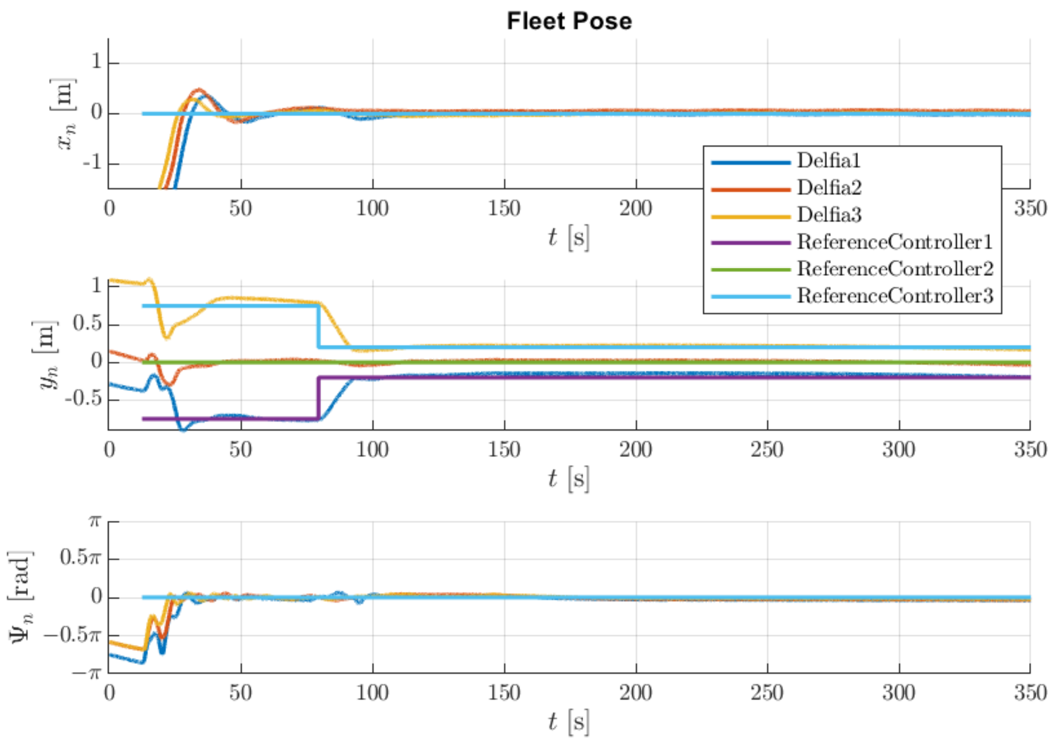


Figure 5.27: System responses during early connecting protocol. Notice how x and yaw reference remains constant. Most notably is the change in y reference at $t = 80s$. Assembly preparation and initial line-up occurs until this moment, after which the reference changes such that the vessels move to connection position.

From approximately $t = 100$ one would expect modules to be approximately lined up such that magnet connectors could properly function. Successful connection can be judged in a more quantitative manner by expressing relative motion between modules. Succesful connections between modules should display zero or near zero relative motion. Figure 5.28 5.29 show relative position between modules. For this, the body fixed frame of the middle vessel (module 2) is used, in which position of the connecting vessels (1 & 3) are expressed (see figure 4.3 for an illustration on how to interpret expressing location of a module in another vessel's body fixed coordinate system).

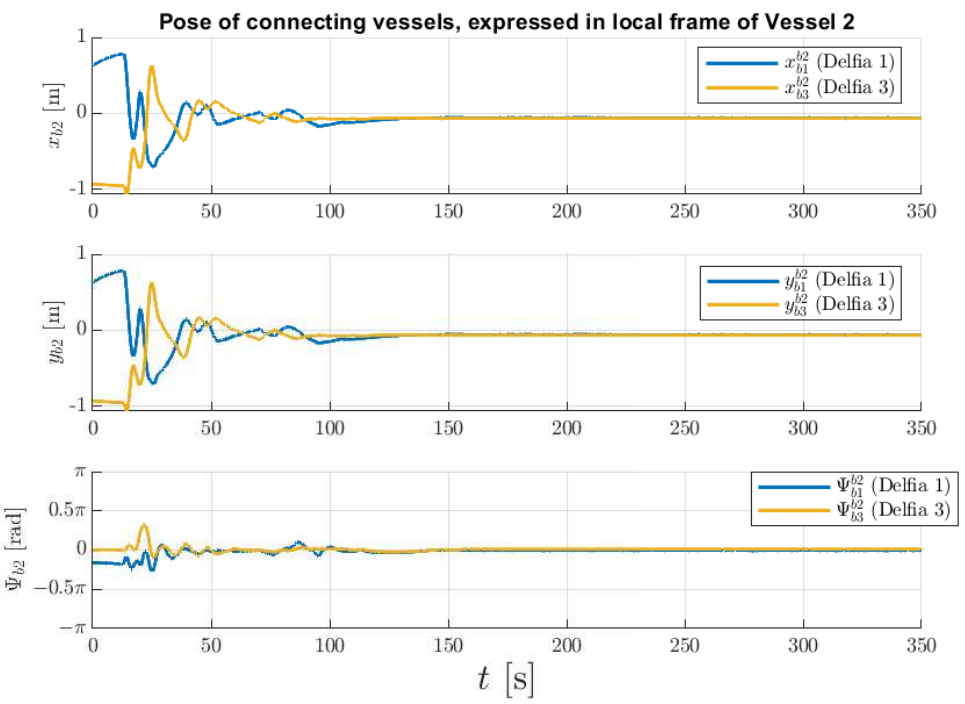


Figure 5.28: Relative motion of two modules connecting to module nr. 2, while running early assembly protocols.

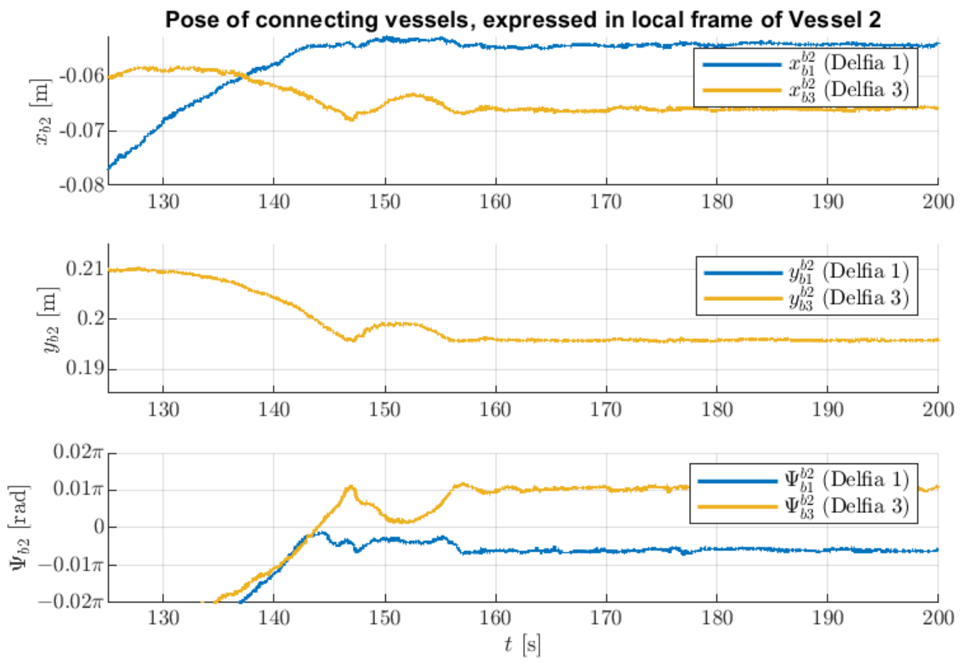


Figure 5.29: Relative motion of two modules connecting to module nr. 2, while running early assembly protocols. Relative motion of a module stops in all three degrees of freedom, more or less at the same time, indicating succesful connection.

Although connection was succesfully achieved, it required a period in the order of 70 seconds, which was considered rather long, even for a proof of concept. It was observed that modules were positioned near the connection site, but with lacking stimulants to get connectors into area of acceptance. Figure 5.30 illustrates how modules float around the connection site, while they were unconstrained in three degrees of motion, resulting in discrepancy in x y and yaw positioning. Small errors in the definition of the body-fixed frame of the modules obtained from the optical tracking system, can be seen in figure 5.29 and 5.30, as the former shows nonzero x and Ψ values after connecting and the latter shows a seeming overlap of vessels. Relative motion in three degrees of freedom are present, which causes the modules to be misaligned, making succesful connecting less probable. Seeming overlap of module 1 & 2 is caused by small errors in the definition of modules' origins. The definitions of the module's origins with respect to the optical trackers were slightly adjusted to decrease the magnitude of these errors for further tests.

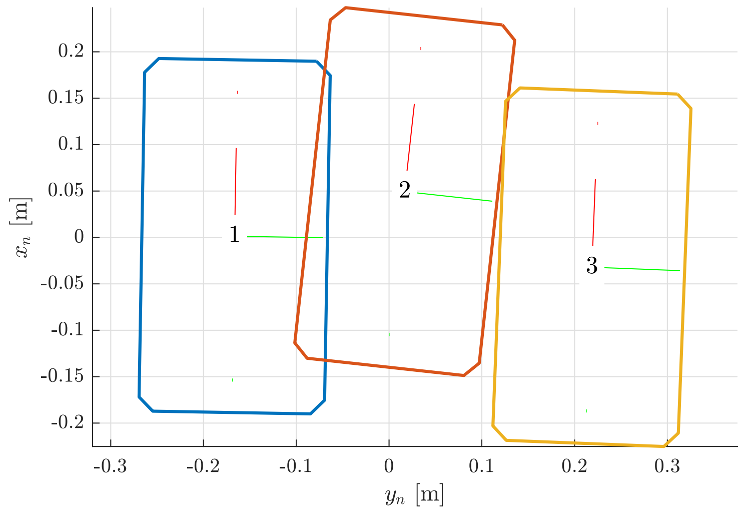


Figure 5.30: Measured system state at $t = 130s$ during early assembly test shown in figure 5.28 and 5.29.

The problem of being insufficiently able to position modules within connector area of acceptance was solved by using normal forces on the modules' hulls to align surfaces. This is a common trick used by regular, human operated ships. A characteristic example is a ferry that pushes itself to a dock, such that it does not rotate or move, allowing safe transfer of passengers. A small normal force between vessels ensures that they keep making contact, and avoid rotation. As modules made contact, only one motion remained, which was translation along the contact surface. With only one degree of (relative) motion connections could be made timely. Figure 5.33 shows how normal forces have been created by setting contoller reference somewhat within the rigid body of a neighbouring module. Figure 5.28 shows relative motion between modules of an assembling system that apply this approach. Notice that relative rotation and motion perpendicular to the contact surface are constrained first. Sliding motion along contact surface later stops as connectors snap into place.

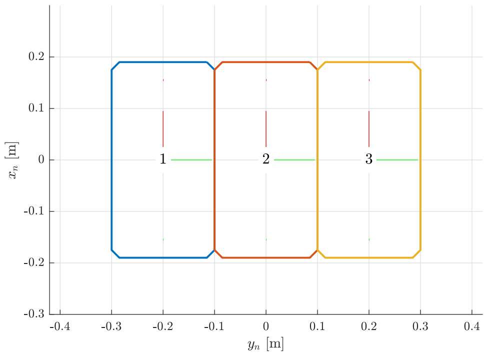


Figure 5.31: Initial reference positions of modules for assembly

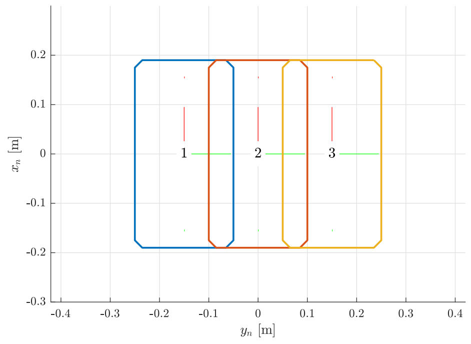


Figure 5.32: Adjusted references for positioning of modules while connecting, to create some normal forces between hulls. The shown references are impossible for the system to reach, as vessels are constrained by their hulls which can not overlap.

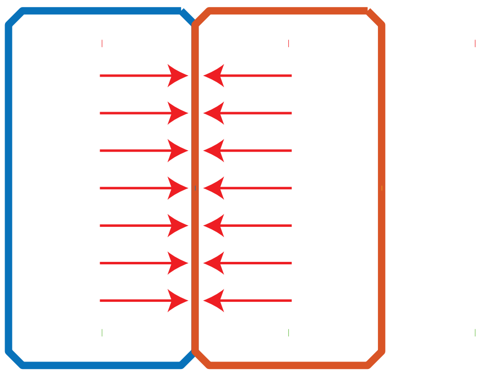


Figure 5.33: Normal forces that result from overlapping references during assembly.

The normal forces between modules were kept to a minimum, as they purely function to remove motion between vessels perpendicular to the connection surface (y), and rotation. Remaining motion was by sliding along the connecting surface of the hull (x direction), where friction did not seem to play a significant role with a smooth surface and low normal forces.

This chapter explained design choices that characterize the developed self-assembling and cooperative fleet control system. At first, the multi-robot control approach with variable topology is explained after which the design of each subsystem is discussed. Implementation is thereafter explained, showing the realization of the fleet control system. The next chapter will assess performance of the developed framework, to see whether or not the system performs according to the presented design criteria and goals.

6 Evaluation

This chapter evaluates the performance of the developed fleet control system. Evaluation is done in two steps, namely in terms of assembly and motion control behavior.

Evaluation starts with a focus on the task of automated assembly. After that, motion control performance is evaluated for two platform configurations; a single vessel scenario and a 3x1 assembled platform. Results of system behavior are presented and performance is quantified. System evaluation concludes by comparing system behavior with the design criteria discussed in section 5.1

6.1. Assembly

This section discusses the performance of the system's ability of automated self assembly. Evaluation starts in a more general sense and ends in a quantitative manner. A variety of system responses performing assembly have been collected throughout this project from system iterations. The results that are discussed in this section are from the most final version of the platform assembly proceidure, of which the overall design is discussed in section 5.1.6, with minor implementational changes as discussed in section 5.2.4.

Performance of reconfiguration will be quantified by evaluating the change in relative pose between neighbouring modules in all considered degrees of freedom. Pose of vessel j with respect to vessel i expressed in the body fixed coordinate system of vessel i is expressed as:

$$\eta_j^i = \begin{bmatrix} \mathbf{p}_j^i \\ \Theta_{ij} \end{bmatrix} = \begin{bmatrix} x_j^i \\ y_j^i \\ \Psi_j^i \end{bmatrix} \tag{6.1}$$

The overall system response during assembly is shown in figure 6.1, which shows poses of the modules in the degrees of freedom of the water surface. Figure 6.1 shows that the reference heading (Ψ) and x position remains constant during the assembly protocol. Response in y direction shows pre-assembly line up at $t <$ 295, after which the modules approach and finally connect.

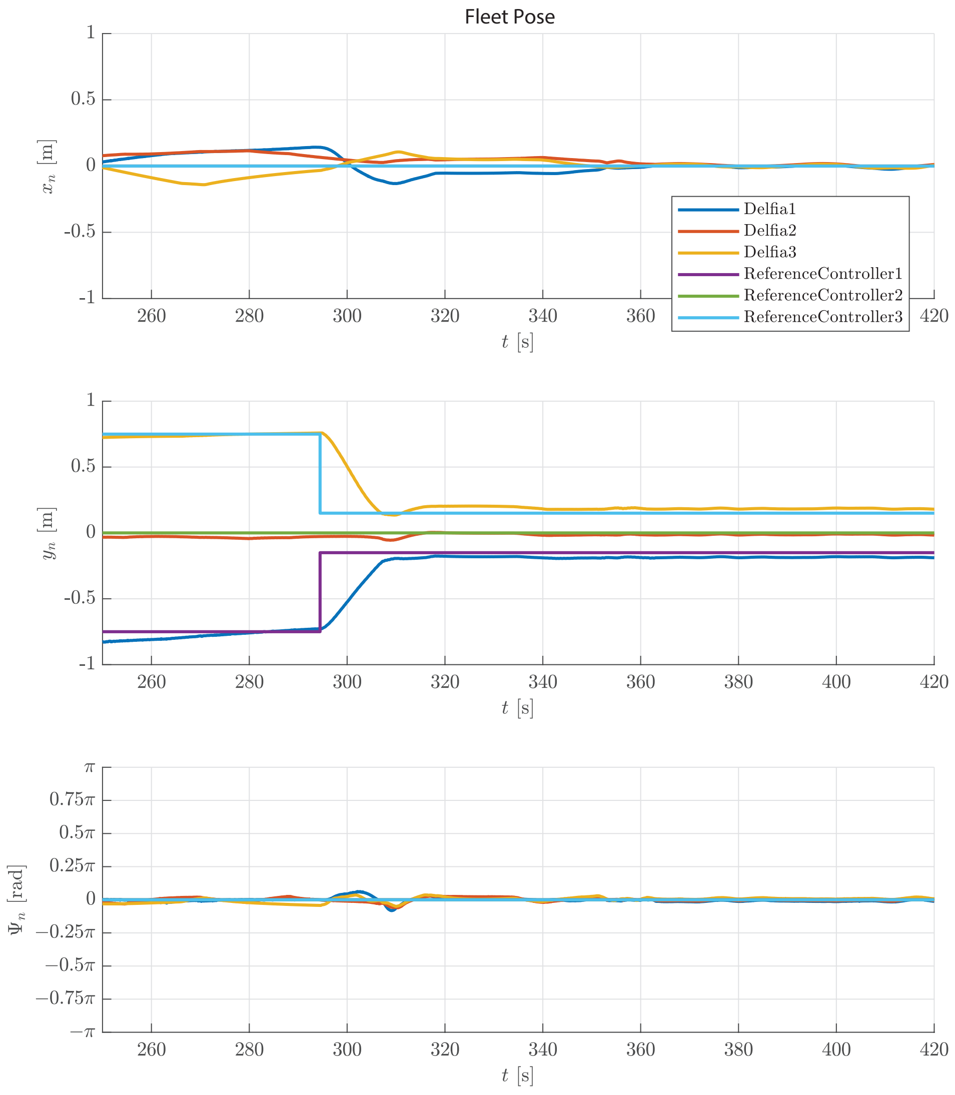


Figure 6.1: Module position plotted versus time, during the platform assembly stage. Vessels are initially lining up side-by-side, until they approach due to changin y reference values.

Assembly succes or failure was easily detectable by eye, but can be supported quantitatively by evaluating relative motion. The two modules attempted assembly simultaneously which is shown in figure 6.2. Simultaneous plotting of relative pose of both module 1 & 3 requires a rather large scale in y direction which doesn't show a lot of detail. Therefor the relative pose of module 1 & 3 are also individually plotted in figure 6.3 and 6.4 respectively.

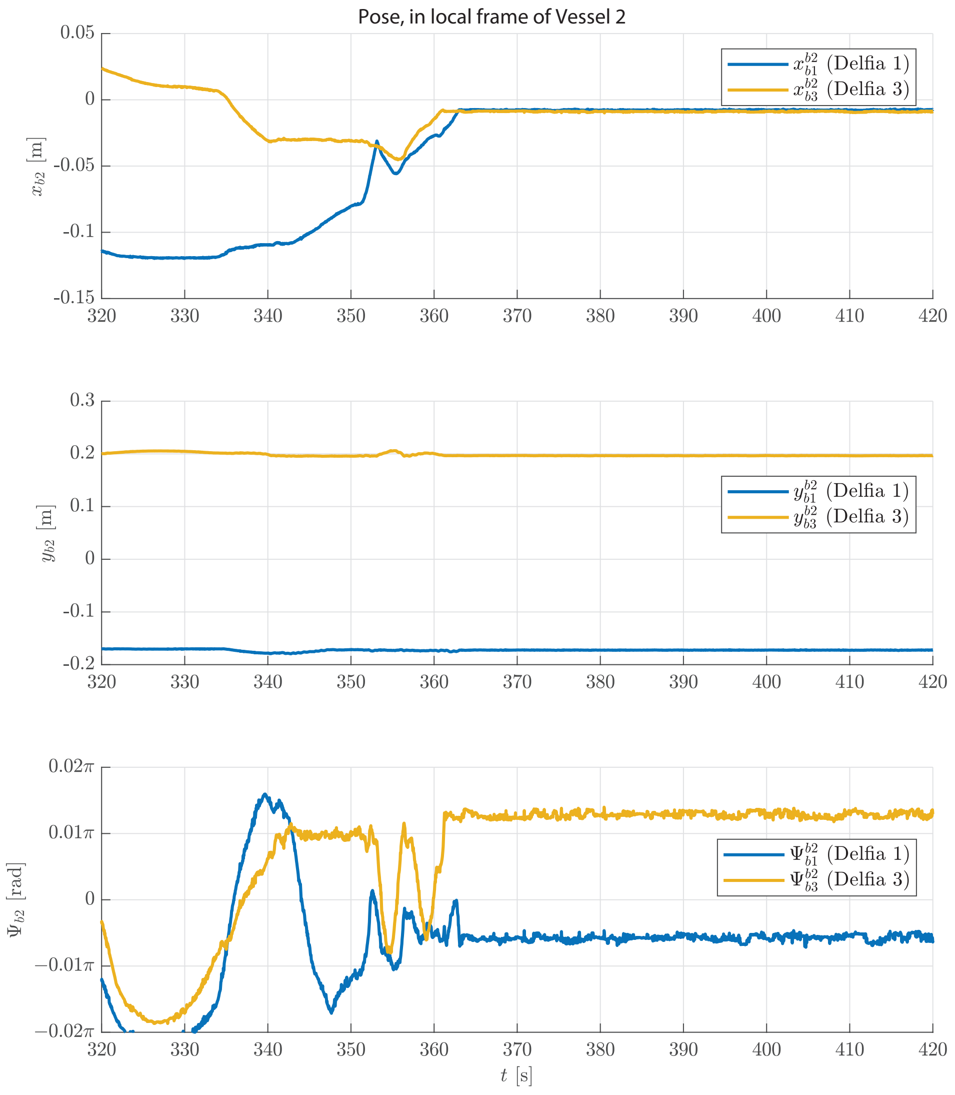


Figure 6.2: Relative motion of assembling vessel system

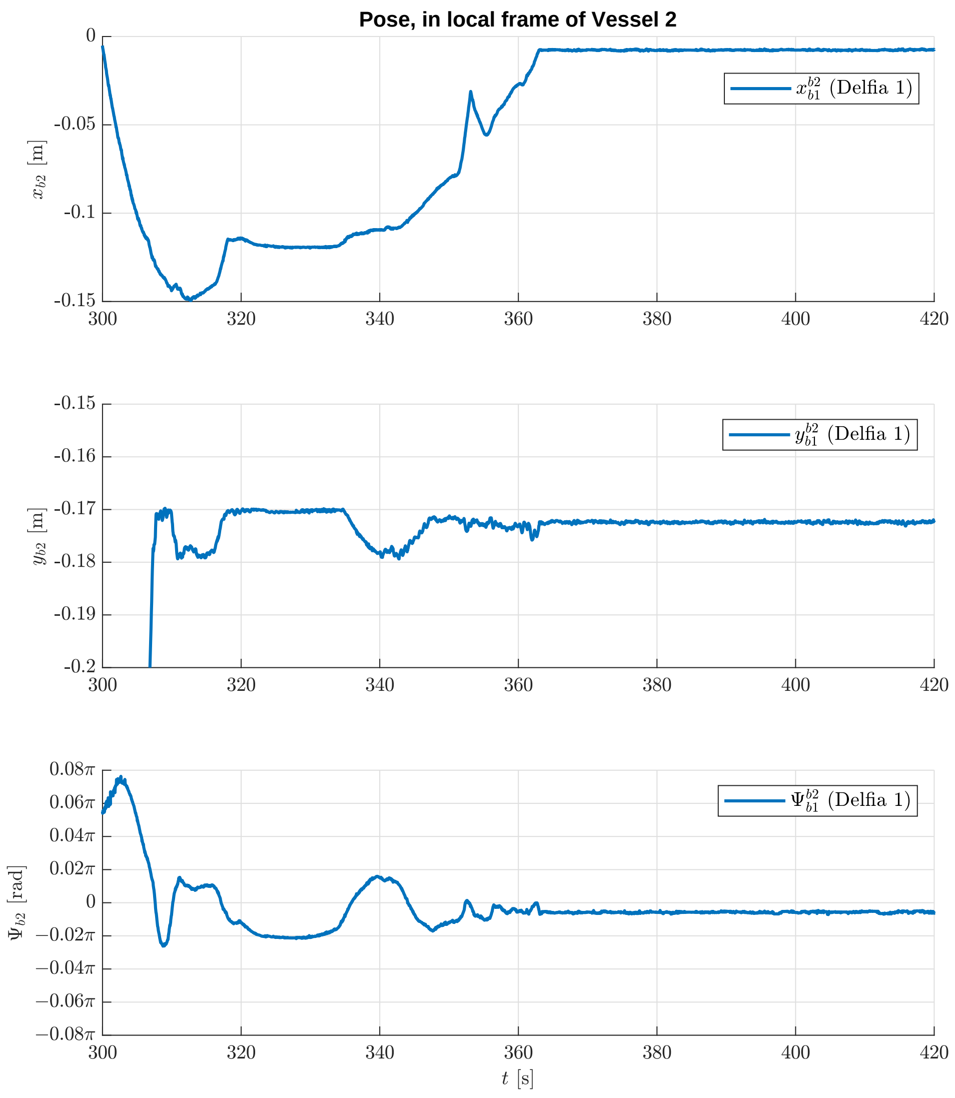


Figure 6.3: Relative motion of module 1 expressed in body fixed frame of module 2.

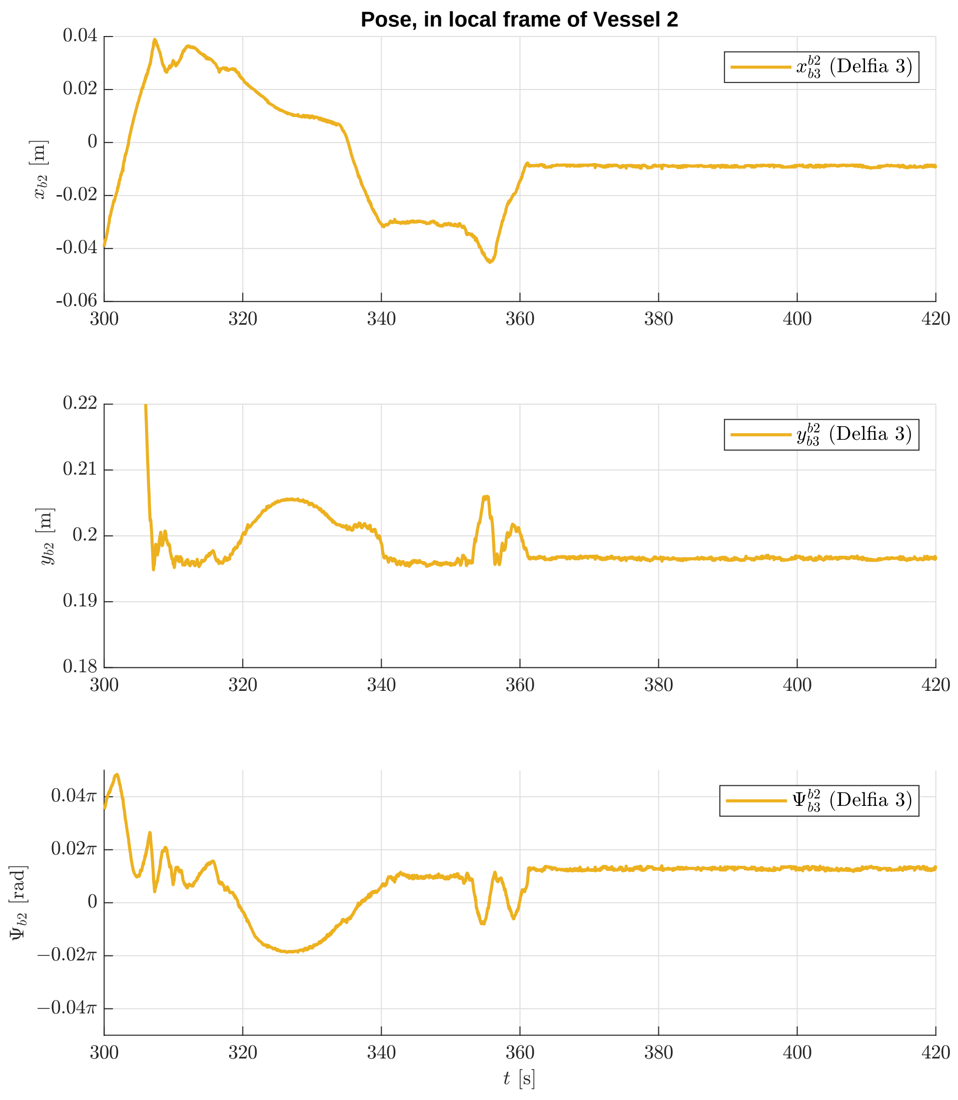


Figure 6.4: Relative motion of module 3 expressed in body fixed frame of module 2.

Some notes on the module's relative positions during assembly are considered worth discussing, particularly on the interpretation of the responses. The signals showing relative positions in figure 6.3 and 6.4 show some forms of plateauing behavior, which is particularly interesting. The slope of these signals becoming a plateau means that relative speed has suddenly become near zero. This is considered to be due to physical contact between hulls. This can be a bump, soft contact, or a connector snapping two modules together. Relative pose of both connecting modules show some similarities. Pose in various DOFs show plateaus where the speed suddenly becomes near zero, which are most clear in y direction for both modules. Furhermore, a clear moment can be observed on which relative motion becomes near zero for not one, but all degrees of motion, indicating a succesful connection.

Figure 6.5 to 6.7 illustrate how the relative motion was interpreted for module 1 at two different timestamps. At $t = 308s$ rapid descelleration, is visible in figure 6.5 in y direction due to hull contact, while figure 6.6 shows that vessel 1 is still misaligned. Some motion and aligning occurs until relative motion suddenly halts, around $t = 364s$. This shows the connector performing to restrain relative motion in configured state, shown in figure 6.7.

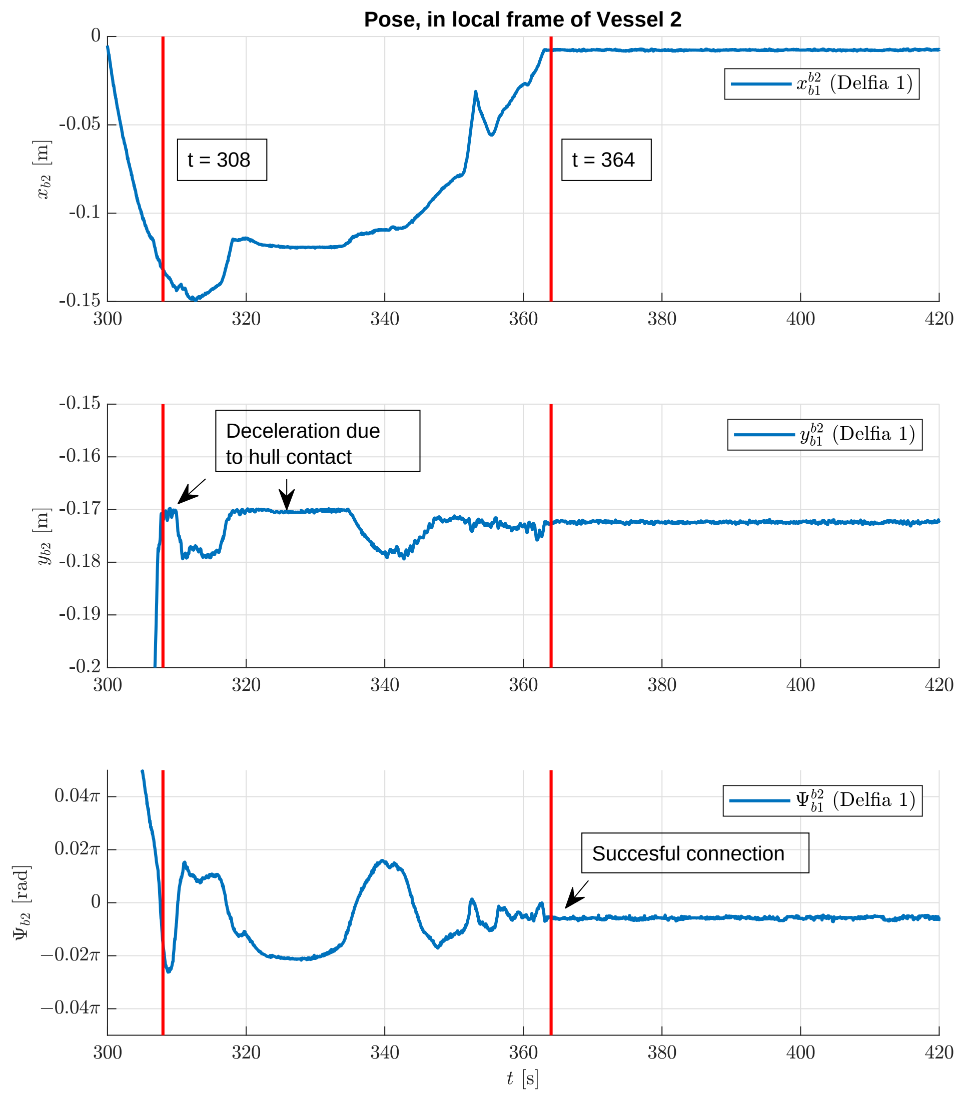


Figure 6.5: EvaluationConnectTwoInstancesAllDofs

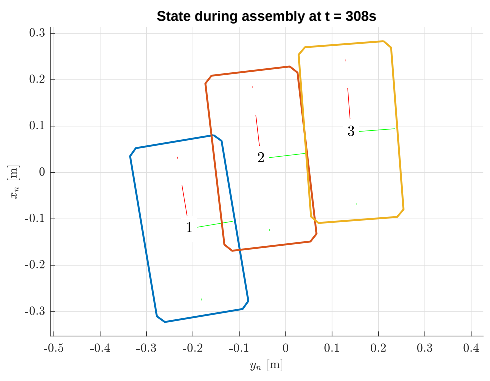


Figure 6.6: Fleet pose shown in a top-down-view at t = 308s while modules are aligning during assembly.

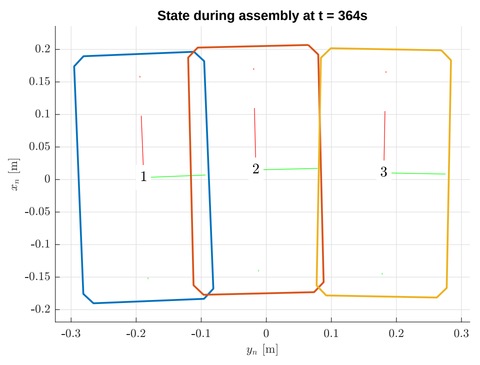


Figure 6.7: Fleet pose shown in a top-down-view at t = 364s, as motion between modules became near zero.

To further quantify the remaining perceived motion after assembly of module 1, the dataset is evaluated from the moment of connecting (approximately at $t = 364$) and 10 seconds thereafter.

		Average	Minimum	Maximum	maxAmpl	Variance
x	[m]	-0.0076628	-0.0082233	-0.0071825	0.0010408	4.3663e-08
y	[m]	-0.17236	-0.17288	-0.17198	0.00089951	2.6465e-08
Ψ	[rad]	-0.018446	-0.02136	-0.014622	0.006738	1.0167e-06

Table 6.1: Signal analysis of relative motion of module 1 with respect to body-fixed frame of module 2 for $364 < t < 374$.

6.2. Platform Motion Control

To evaluate the motion control performance of the developed framework, vessel system will be tasked to follow a time varying reference signal with step changes. Performance quantification is done by expressing rise-time and settling-time and overshoot (see fig 6.8)of step responses to reference position of the platform in all considered degrees of freedom. Thoughout experiments steps in the platform's reference pose are given in one degree of freedom at a time.

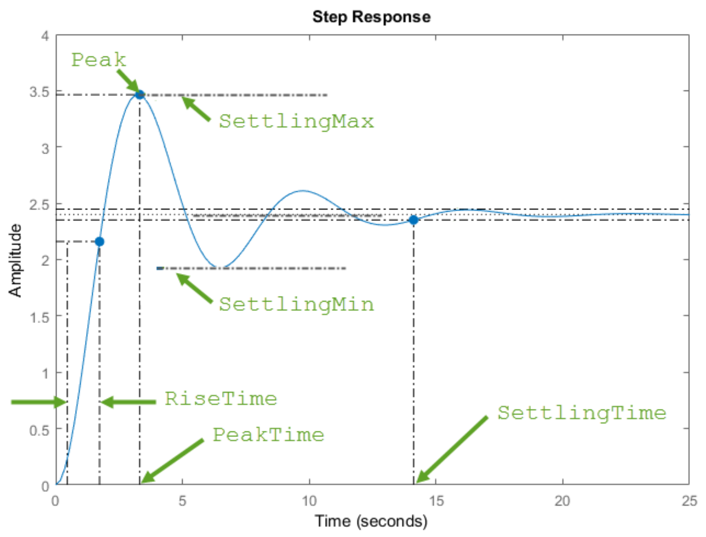


Figure 6.8: Interpretation of typical quantities illustrated that are used to characterize a second order system [29].

Platform motion is evaluated for two scenario's; single vessel operation and 3x1 lattice platform configuration, as shown in figure 6.9 and 6.10. Note that control parameter tuning occured in single vessel operation. Any other arbitrary configuration, including the tested 3x1 platform, is required to behave satisfactory due to the configuration-adaptability of the control system.

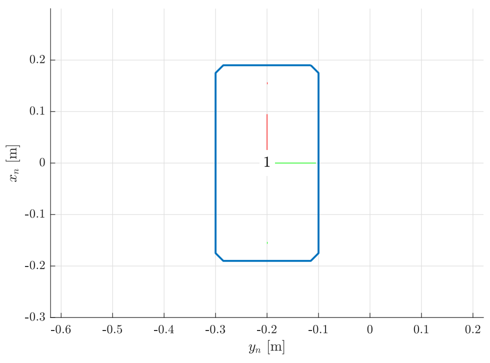


Figure 6.9: A single Delfia configuration

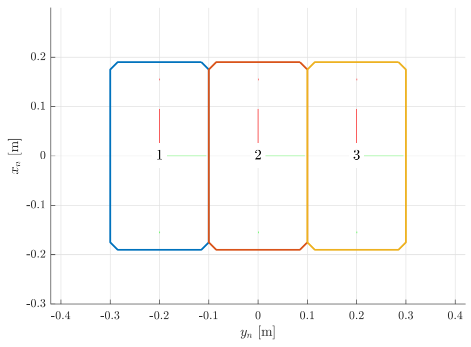


Figure 6.10: A 3x1 platform configuration, which represented assembled platform state troughout experiments.

Both scenarios get similar tasks of short displacements as a step input on the reference pose. These step responses are then evaluated to quantify this performance in post processing. Figure 6.11 shows three step responses for single vessel controlled by the system. Figure 6.12 shows responses of a 3x1 configured platform. Responses of both configurations that are plotted have quantitative analysis parameters shown in tables 6.2 to 6.7.

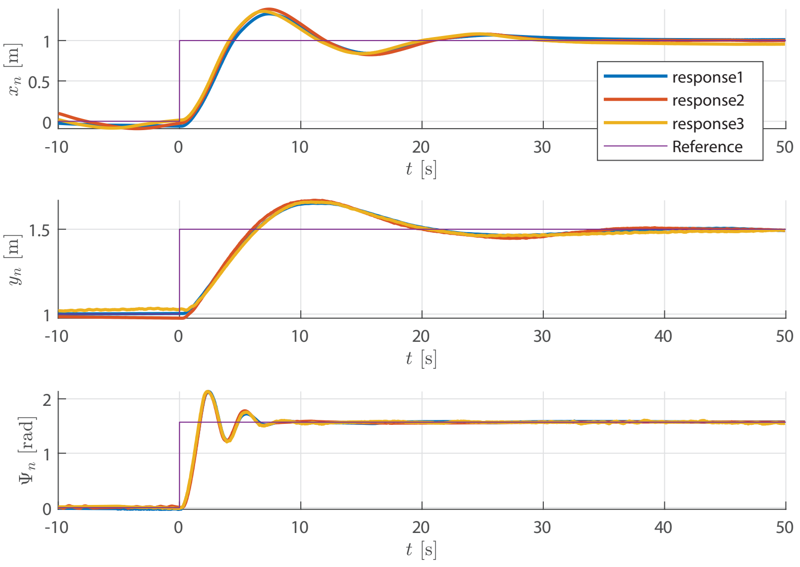


Figure 6.11: Step responses of single Delfia configuration. Responses are gathered by changing reference of a single degree of motion (x y and yaw) at a time. Three datasets are shown, which show that the responses are rather constant.

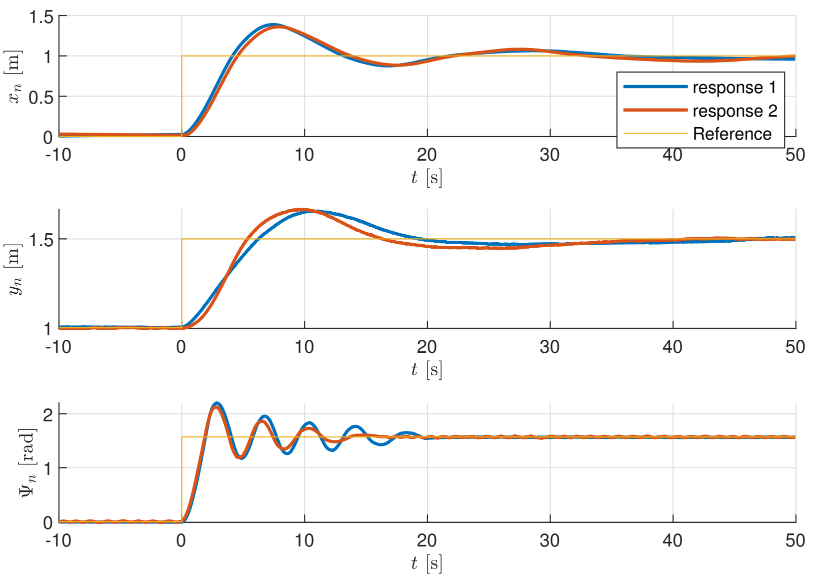


Figure 6.12: Step responses of a 3x1 lattice configuration. Responses are gathered by changing reference of a single degree of motion (x y and yaw) at a time. Two step responses are shown for each degree of motion.

Step responses are evaluated quantitatively in terms of rise-time, overshoot and settling-time, calculated by Matlab's 'stepinfo' function [29]. Rise time is defined as the time between a 10% and 90% rise of the signal. which are shown for all degrees of motion of both single and assembled configuration in tables 6.2 to 6.7. The configured settling-time criteria is met if the signal is dampened within a factor of 0.05 of the steady state value.

	RiseTime [s]	SettlingTime [s]	Overshoot [%]
response1	2.803	27.7096	33.1361
response2	2.3159	27.0722	38.8513
response3	2.6438	26.6143	36.229
Average	2.5876	27.132	36.0721

Table 6.2: Single vessel x-direction step response evaluation

	RiseTime [s]	SettlingTime [s]	Overshoot [%]
response1	2.9263	31.2773	38.7962
response2	2.9202	45.2133	36.0778
Average	2.9232	38.2453	37.437

Table 6.3: Assembled 3x1 platform x-direction step response evaluation

	RiseTime [s]	SettlingTime [s]	Overshoot [%]
response1	4.4688	32.2703	30.6531
response2	4.0913	31.7191	34.0618
response3	4.3417	35.8629	32.147
Average	4.3006	33.2841	32.2873

Table 6.4: Single vessel y-direction step response evaluation

	RiseTime [s]	SettlingTime [s]	Overshoot [%]
response1	4.5348	32.9588	31.2256
response2	3.3676	32.6829	33.3475
Average	3.9512	32.8209	32.2865

Table 6.5: Assembled 3x1 platform y-direction step response evaluation

	RiseTime [s]	SettlingTime [s]	Overshoot [%]
response1	1.0576	6.1474	35.8941
response2	0.97707	6.105	35.6279
response3	0.99748	6.0969	35.9651
Average	1.0107	6.1164	35.829

Table 6.6: Single vessel Ψ-direction step response evaluation

	RiseTime [s]	SettlingTime [s]	Overshoot [%]
response1	1.2894	18.2124	40.2248
response2	1.3382	12.8114	35.3374
Average	1.3138	15.5119	37.7811

Table 6.7: Assembled 3x1 platform Ψ-direction step response evaluation

Tables 6.2 to 6.7 show quantitative indicators of performance of the system in two configurations. The graphs and tables show that the designed system is able to control various configurations. Step-response performance could be optimized more, but that is not the point. More important is learning from the scalability of this motion control system, as its novelty comes from using information about the platform's shape, size, and configuration. Thus, responses of single and assembled configuration will be compared.

Recall from section 5.1.4 that configuration dependent platform motion control was designed with using an approximate time scale. This timescale took into account varying inertia and available control effort, with respect to the reference configuration (where reference configuration was implemented as a single vessel).

Taking a look at the appoximate timescale for linear motion, it was found that the available thrust and inertia equally scaled linear to the amount of modules, since a homogeneous fleet is used. Thus maximum platform thrust scales as

$$F_{platform,max} \approx n_{modules} * F_{single,max} \tag{6.2}$$

where $F_{platform,max}$ is maximum platform thrust when it fully utilizes all azimuth thrusters, $F_{single,max}$ is the maximum thrust of a single Delfia-1* module. Similarly, platform mass scales as

$$m_{platform} \approx n_{modules} * m_{single} \tag{6.3}$$

Computing approximate time scale of linear platform motion using equation 5.16 yields

$$C_{x,y} = \frac{F_c \, I_{ref}}{F_{ref} \, I_c} = 1 \tag{6.4}$$

which is quite intuitive, as the maximum thrust to mass ratio remained constant. Angular control effort and angular-inertia were appoximated to not scale similar. Values of estimated inertias and control effort are shown in table 6.8, which yield an approximate time constant for rotation of

$$C_{\Psi} = 1.2980 \tag{6.5}$$

	Inertia	Maximum input
Single vessel: Translation	$4.1275kg$	$0.4320N$
Single vessel: Rotation	$0.1410kg*m^2$	$0.0670N*m$
3x1 Configuration: Translation	$12.3825kg$	$1.2960N$
3x1 Configuration: Rotation	$0.7091kg*m^2$	$0.2596N*m$

Table 6.8: Approximate scaling of reference and assembled configuration

Correctness of using the assumption of the used timescale can be evaluated by comparing response times of reference and assembled configuration. This will show to what extent the system maintains similar behavior over changing configurations, which will give indicators how it might behave for other configurations.

	RiseTime $[s]$	SettlingTime $[s]$	Overshoot $[\%]$	PeakTime $[s]$
Single module	2.5876	27.132	36.0721	7.2288
3x1 conf.	2.9232	38.2453	37.437	7.6737
Ratio	1.1297	1.4096	1.0378	1.0615

Table 6.9: Comparison between step responses of reference (single vessel) and adapted (3x1) configuration of linear motion in x direction

	RiseTime [s]	SettlingTime [s]	Overshoot [%]	PeakTime [s]
Single vessel configuration	4.3006	33.2841	32.2873	11.0471
3x1 configuration	3.9512	32.8209	32.2865	10.4886
Ratio	0.91876	0.98608	0.99998	0.94944

Table 6.10: Comparison between step responses of reference (single vessel) and adapted (3x1) configuration of linear motion in y direction

Dimention: yaw	RiseTime [s]	SettlingTime [s]	Overshoot [%]	PeakTime [s]
Single vessel configuration	1.0107	6.1164	35.829	2.3432
3x1 configuration	1.3138	15.5119	37.7811	2.8033
Ratio	1.2999	2.5361	1.0545	1.1964

Table 6.11: Comparison between step responses of reference (single vessel) and adapted (3x1) configuration of rotational motion in yaw direction

To evaluate control system adaptation performance the ratio between performance indicators of reference and configured state is used from table 6.9 to 6.11.

Overshoot seems to remain rather constant, varying 3.78%,0.002% and 5.45% for x,y and Ψ motion respectively.

The **rise-time** of linear motion somewhat behaves according to the approximate timeconstant of 1 (equation 6.4). The risetime ratios of 1.1297 and 0.91876 for x and y motion are quite near the approximated scaling of 1.0. Differences can be explained by the fact that the implemented control system does not use directional dependent mass or dampening for linear motion. Platform mass was approximated as constant in all directions for adapting PID gains, as the position controllers used global coordinates.

The rise-time of angular motion changed with a ratio of 1.299, which is rather close to the approximated time constant of $C_\Psi = 1.2980$ (6.5).

Settling-time varied significantly between some measurements, such as the in the x-response of the assembled configuration (table 6.3 settlingtimes of 31.2773 and 45.2133 s). This is peculiar as the two responses in x direction plotted in figure 6.12 are very much alike. The steady state criteria was reached half a period later, but with the long periods of oscillation (in the range of 20 seconds), this resulted in a significant difference in settling time. Applications that highly value the performance indicator of settling time are recommended to rethink appropriate settling criteria. Most other settling times seemed rather consistent throughout datasets, with the exception of rotation of the assembled platform. Two rotation responses of the assembled platform (figure 6.12, Ψ dimention) show identical behavior on the first period, yet seem to show some difference in oscillation dampening, which are in line with the different settling times (table 6.7).

Collaborative distribution of control effort was aimed to perform in a coordinated fashion. Figure 6.13 to 6.16 show various independent and combined modes of motion in 3x1 assembled configuration, illustrating how the control allocation protocol assigns tasks to actuators.

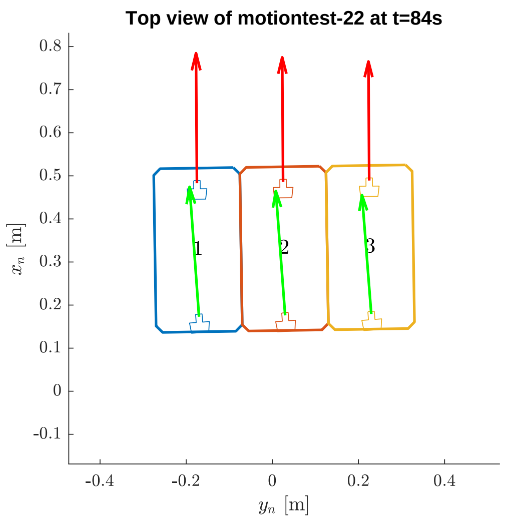


Figure 6.13: Control effort allocated to create (almost) pure force in x direction.

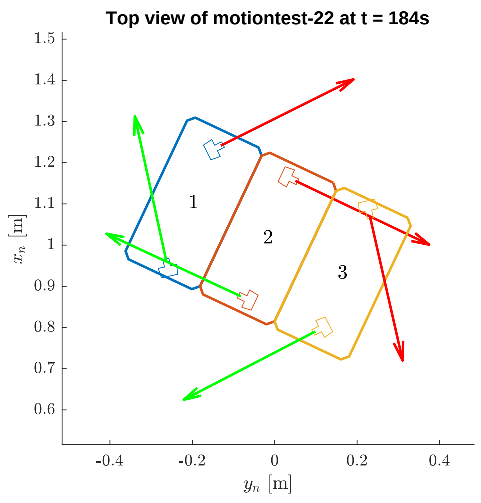


Figure 6.14: Control effort allocated to create pure torque about CM.

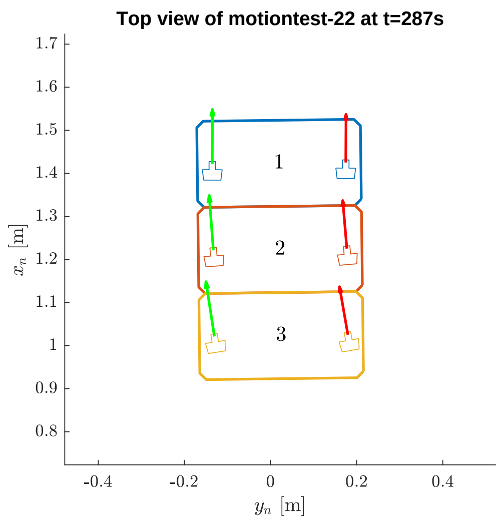


Figure 6.15: Control effort allocated to create (almost) pure force in body-fixed-y direction.

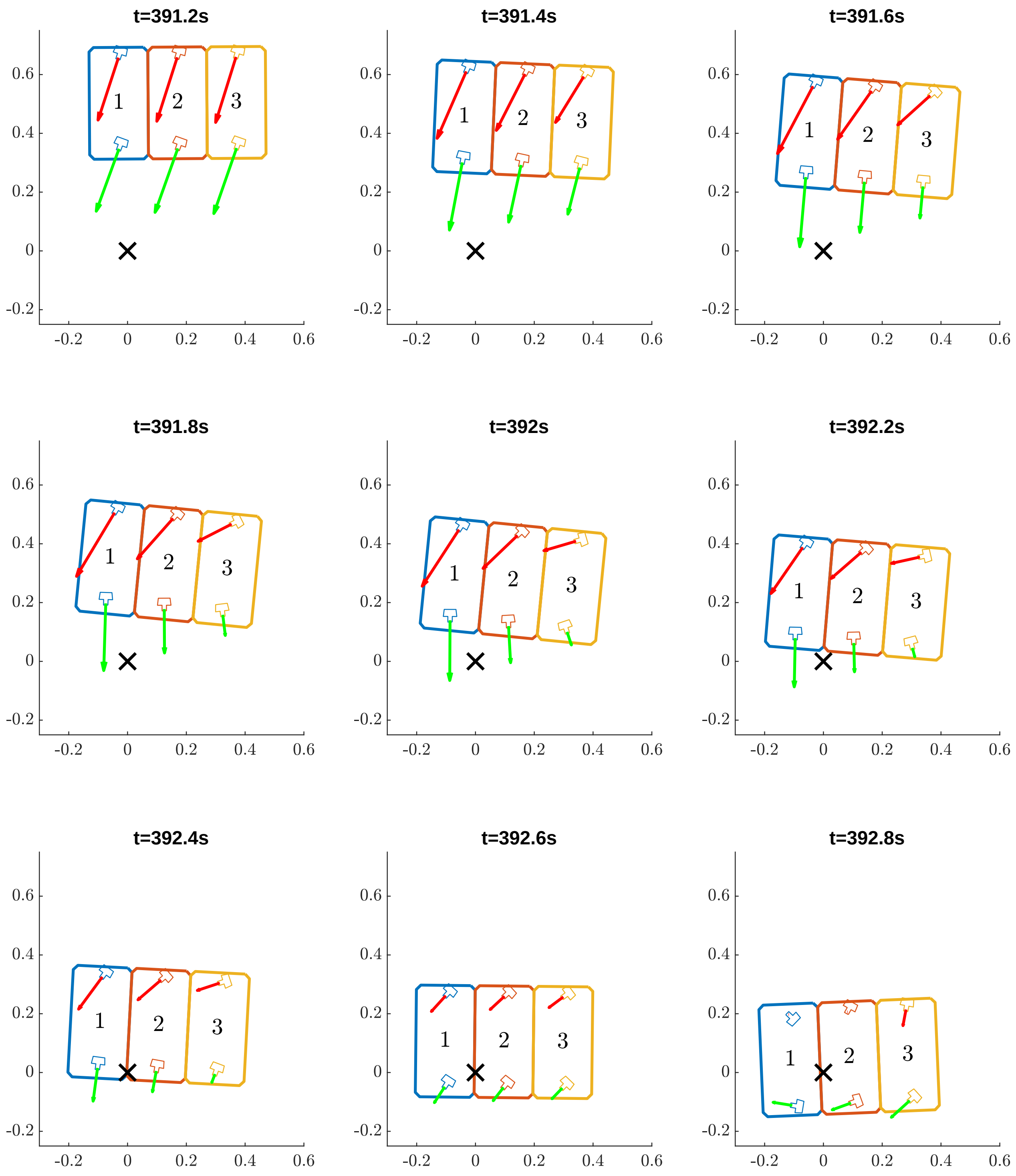


Figure 6.16: Various timeframes of motiontest-22 moving to a reference position (×) involving simultaneous rotation and translation. Arrows depict thruster forces, showing combined control effort (forces in various directions and torque) allocated and distributed over the platform's modules.

6.3. Comparing results with design criteria

Section 5.1 starts describing design criteria which are evaluated here.

Firstly, the system is required to perform automated vessel platform reconfiguration. This criteria is reached as has been observed throughout operation (by seeing magnet connectors snap into place), yet this is quantitatively supported by expressing relative module motion in section 6.1. Figure 6.5 illustrates a typical sudden stop in relative vessel motion in all degrees of freedom indicating succesful connection. Assembly of the final developed system behaved consistently throughout testing. Tab. 6.12 shows how relative motion after connection was quantified, being near zero, compliant with the set design criteria.

Motion	Maximum signal amplitude	Variance	Criteria	Compliant
x	0.0010408 m	4.3663e-08		✓
y	0.00089951 m	2.6465e-08	≈ 0	✓
Ψ	0.006738 rad	1.0167e-06		✓

Table 6.12: Key performance indicators regarding assembly and compliancy versus design criteria.

Secondly, the system needs to perform motion control in a collaborative manner, where multiple robots work together to achieve a single goal. Centralizing platform control to a single entity facilitated modules contributing to reaching one objective and allowed coordination strategies to increase actuator usage effectiveness. Figure 6.12 displays responses to reference step changes of a 3x1 lattice configurated platform, showing how the developed system controls to converging and stabilizing at reference state.

Configuration vs KPI	Risetime		Settlingtime		Overshoot	
	Criteria	Compliancy	Criteria	Compliancy	Criteria	Compliancy
Single module configuration	t_r <10s	100%	t_s<40s	100%	os<50%	100%
3x1 platform configuration		100%		83,3%		100%

Table 6.13: Key performance indicators regarding motion control of all degrees of motion.

The last hard design criteria requires the developed system to perform the two abovementioned behavioral criteria simultaneously. Reconfiguration and configuration dependent platform motion control operated within the same framework, where control approach changed in troughout operation as a response to platform assembly.

The designed framework theoretically supports motion control of any arbitrary configuration, as whas desired. Experiments were conducted on two quite general shapes that both showing satisfactory responses. The control approach adapts to unknown configurations by using scaling rules aimed to be representative over a wide variety of shapes and sizes. Although scaling rules have been designed relying on physics, used assumptions need to be re-evaluated for correctness of future configurations, particularly with more extreme shapes and scales.

The framework was developed aimed to represent a logistic solution that could be implemented in the near future. Off course full scale commercial adoption will bring some challenges that were not present in the experimental lab setup, such as disturbances due to wind and current. All system components are considered replacable for other solutions that might better fit more realistic scales, requirements and environments.

Replacing or improving components of the system is stimulated by making the control framework as modular as possible. Various subsystems can be further developed, which was expected as exploration of a novel combination of behaviors was the goal rather than optimization.

From this evaluation it is concluded that the system developed througout this project is compliant with the set design criteria.

This evaluation and findings throughout the process of system development are presented in the next chapter to conclude this project, mention points of discussion and present the author's view on future research directions.

7 Conclusion

This paper presents systematic development steps and design choices to realize a multivessel system able to configure vessels into interconnected platforms while performing perform configuration dependent collaborative control.

The developed framework was designed to reflect a scenario of an automated fleet system with the task to solve a logistical problem, by forming floating infrastructure in a predefined shape that can be loaded, which subsequantially needed to perform motion tasks. Implementation serves as proof of concept, which is evaluated as performing compliant to the set design criteria. Full scale commercial adoption will bring more challenges that were not present in the experimental lab setup, such as disturbances due to wind and current. Used design approach is considered applicable to other scales and environments as all system components are considered replacable for other solutions better serving requirements of the usecase.

The research approach relied on searching answers to the following main research question ans subquestions:

Main research question

- How can a fleet of modular surface platform vessels be controlled to achieve automated assembly and configuration-dependent platform control?

Sub-questions

1. What is the state of the art within automated vessel platforming systems?
2. What characteristics does a vessel system have that integrates automated assembly with configuration dependent control?
3. How can the dynamics of the multi-vessel system be represented?
4. How can a fleet control framework be developed that performs automated platform assembly and configuration adaptive platform control?
5. What is the performance of the developed system?

Literature survey describes major literature of projects related to vessel self-assembly and collaborative control approaches to answer the first subquestion. The review describes various works categorized as 'vessel platform self-assembly' and one project showing development of a 'collaborative and coordinated multi-vessel-platform control'. It seemed natural that modular vessel platform automation will encompass behavior of both categories in the near future to become of highest societal benefit. Yet no sources were found describing both behaviors integrated in a single system, leading to the gap of knowledge that this project adresses.

A more in depth look is given to the definition of multi-robot systems that perform 'self-assembly' and 'collaborative, coordinated control' to answer the second subquestion. This section significantly relied on literature works to map characteristics of both systems from different perspectives. Works with a general robotics perspective (versus marine robotics) were shown to have significantly more extensive descriptions of traits, design choices and categorizations, and thus were included within characterization efforts. It was shown that many approaches were possible to design the discussed systems, each with their own benefits and limitations with different probable usefulness for logistic purposes. Generalized system characteristics are distinct and narrowed down set of solutions that can be considered more feasible to perform in commercial applications, taking into account the design trend in other projects and their emergent characteristics.

A multi-vessel state description is proposed, together with a novel approach to predict dynamics of a combined waterborne structures to answer the third subquestion. Existing (single) vessel state descriptions are extended to formulate this multi-vessel state description aiming to enable consistent system notation of scenarios pertaining more than a single object (vessels, coordinate systems) that can be referred to. Furthermore, the concept and definition of a platform-coordinate system is introduced. This resulted in three distinct types of coordinate systems; vessels, assemblies/platforms and global (e.g. inertial) frames that define state of an object and coordinate system in which parameters of other objects (pose, velocities, forces) can be expressed. The proposed approach to predict dynamics of a combined waterborne structure utilizes superpositioning of module models into a combined platform model as alternative of parameter estimation experiments for variable configurations. Particular emphasis is given to taking into account platform shape (besides numbers) to formulate a combined model that inherits directional dependent terms, such as hydrodynamic added mass or directional dependent dampening. Connections between modules were assumed rigid to allow expression of module modules in a single point and orientation (such as the platform frame) analytical by means of coordinate system translation and rotation to formulate the combined expression. A key requirement of this approach is that individual module models sufficiently describe the behavior of a model also in very close proximity of other modules, or discrepancies due to module proximity are sufficiently negated by additional compensating terms. The proposed platform model can form a building block for approximated dynamics of the combined structure utilizing obtainable parameters (individual module models), which can be extended to include vessel proximity effects if needed.

An experimental fleet control system incorporating assembly and collaborative control automation has been developed to answer the fourth subquestion. A homogeneous set of Delfia-1* model scale robotic vessels formed the fleet of modules for the proof of concept in indoor lab environments. Major design decisions are as follows:

- The multi-robot network topology and hierarchy is centralized on a platform level, to support collaboration and a framework for coordination. This means that network topology changes according to changing configurations.
- The overall platform-level control approach is subdivided into modular subsystems. The tasks of these subsystems can be solved in various ways, of which the chosen solutions are explained and design choices motivated.
- The approach of controlling a multi vessel platform in a collaborative manner uses the proposed approximate platform model dependent on configuration size and shape. The platform model is formed by combining models of connected modules, which were assumed known, taking into account individual module dynamics, placement, and orientation.
- Desired control effort is generated with a set of parralel PID controllers that adapt to estimated platform parameters, such as configuration dependent maximum thrust and system inertia.

Chapter 6 answers the fifth and last subquestion by evaluating the performance of the final implemented system, starting with the ability of self-assembly, following with performance of the motion control system and it's ability to operate on any configuration.

It is concluded that the system developed througout this project is compliant with the set design criteria. Self assembling behavior was evaluated by expressing relative module motion as key performance indicators, which should be ideally zero if connected. This behavior was indeed observed, where the maximum motion after connecting in a 10 second period was within bounds of $1.0408e-3$ and $0.8995e-3$ meter for x and y motion and within $6.738e-3$ radians (≈ 0.3860 degrees) for rotation.

Main objectives of the motion control system were met, as the considered configurations converged to changing reference inputs in a reasonable time without reliance on configuration-specific methods. Performance of the fleet motion control system is evaluated from reference step responses on all considered degrees of motion (forward, sideways and rotating) of single vessel and assembled 3x1 platform configuration. The configuration dependent control system was aimed to effectively utilize available actuators while remaining in control of a changing dynamical system using the approximate platform model. The control to a single entity allowed coordination strategies to increase actuator usage effectiveness. Furthermore the controller should show acceptable adaptation to varying configurations.

As the control approach is intended to work on arbitrary platform configuration, it is evaluated for two configurations, namely single operaration and as a 3x1 lattice structure. Performance of the fleet motion control system is evaluated from reference step responses on all considered degrees of motion (forward, sideways and rotating) by expressing rise time, settling-time and overshoot as key performance indicators (with step amplitude as characteristic motion of 0.5m & 1m for translation and a 90°rotation). Behavior in terms of overshoot was compliant with the design criteria (<50%) for motion of both configurations along all dimentions averaging 34,7% for single and 35,8% for 3x1 configured assembly. Design goals for settlingtime (response within factor 0.05 of steady-state in $t_s < 40s$) were met through all single operation and in 83.3% of configured tests. Risetime behavior proved significantly within design criteria ($t_r < 10s$) in all directions and configurations, typically under four seconds.

The discussed criteria on behaviors of self-reconfiguration and collaborative control simultaneously within the same framework are considered achieved. It is concluded that a modular vessel platforming system can be equipped with features automating reconfiguration and collaborative and coordinated control, where the following major challenges are identified as follows

- Both individual behaviors can already be implemented in a wide variety of approaches, where integration of the two widens the design spectrum significantly more. Both behaviors can be considered complex by themselves, similarly increasing in a combined framework. This complexity toughens challenges of design, but also in facets realizing robustness of a multi-layered system.

- Earlier developed works showed iterated views on solutions to achieving the desired behaviors. It was found that a main factor for succesful integration is ensuring that both behaviors are interoperable. This particularly means that platform motion control systems need to be able to adapt in real time to configuration changes.

7.1. Discussion

All solutions for subsystems aimed to let the system perform in a lab setting with a fleet of Delfia-1* modules, yet it must be said that these design choices are not optimized or guaranteed to perform on use cases with a different fleet, scale, environment and goals. Replacing or improving components of the system is stimulated by making the control framework as modular as possible. Various subsystems can be further developed, which was expected as exploration of a novel combination of behaviors was the goal rather than optimization. That said, the reader is invited to use the approach and description of implementation for inspiration to develop waterborne multi-vessel collaborative systems.

The control approach uses the concept of an approximate platform-model to then influence motion control behavior similar to Park et al. [42], but the formation of this appropriate model differs. Their estimates of rigid body inertia scales by n and n^2 for translation (mass) and rotation (moment of inertia) respectively, yet this approach does not take platform shape into account. Their scaling rules make sense in some respects, but are simplistic in others.

There are two factors that are considered to limit estimated model accuracy. Firstly, the actual moment of inertia of a combined structure will differ depending on configuration shape, and not solely on numbers. Units at distance from the centre of mass provide far greater contributions to inertia than centered ones, similar as to how the moment of inertia of a flywheel and an equally weighted solid rod greatly differ, explained by the parralel axis theorem. Secondly, a vessel's hydrodynamic effects are directionally dependent, thus orientation and placement of a module with respect to the rest of the platform will affect hydrodynamic forces acting on it, thus taking this into account has the potential to find better platform representations.

Various models for hydrodynamic forces on a ship exist, often distincting accelleration and velocity dependent terms being modelled as "added mass" and dampening. Accelleration dependent contributions of hydrodynamic forces are commonly modelled as constant but not equal in all directions, yielding satisfactory accuracies in most conventional ship use cases Humphreys and Watkinson [28]. Is questionable to what extent such constant added mass terms remain an accurate model in the scenario of vessel assembly where modules operate in unconventionally close proximity. Yet given that (at least some part of) the hydrodynamic forces on a module are accurately represented by this constant directional dependent added mass, then the orientation and placement of that module should be taken into account when estimating contribution to the assembled structure.

Various concepts that are discussed in this thesis have the potential to be competitive problem solvers, yet feasibility needs to be always taken into account. Increased combined platform model accuracy (of a complex model estimator) is naturally desirable, yet one should ask what requirements of model estimators are. Perhaps a simple, less accurate (but satisfactory) approach can have outweighing benefits in terms of speed and cost effectiveness. Alternatives of predicting a dynamical platform model also exist. Parameter identification experiments can be conducted for all reasonably expectable configurations if they are not too numerous, or a controller can be designed to quickly learn from its responses to formulate a model once it encounters an unfamilliar configuration. The concept of reconfigurable robot systems may not be competitive, or only for a very small niche of logistic challenges. Seo et al. [45] notices that of its three main benefits of MMR systems, Versatility, robustness and low cost, only Versatility seems to commonly improve while robustness and cost are usually decreasing for revieuwed literature showing research prototypes. As development of MMR systems is still an active topic of research, we can hope to see this leading to MMR systems becoming more viable in the future. Positive system aspects can be further enhanced, while negative aspects are reduced or reduced to managable levels. This hopefully leads to a scenario where positive aspects far outweigh the negative, such that reconfigurable vessel systems become a commonly applied powerful logistical tool from a social, technical and economical vieuwpoint.

Throughout experiments only two configurations were tested, although the designed control system is aimed to function on any arbitrary configured platform. The results on the two configurations are positive, and indicate good hopes for performance on many other configurations, although this needs to be validated. The described approach on adapting to unknown configurations bears foundations which can be further explored to improve potential control over arbitrary configured dynamically shapeshifting vessel platforms.

7.2. Recommendations

Further investigation of arbitrary configured control approaches

The amount of works presenting approaches of controlling arbitrary configured modular marine structures is limited, as only Park et al. [42] was found adressing that particular challenge. Many design choices from the system developed in this work are similar to this earlier work, such as adoption of platform-level centralized control topology. Distinction between control effort generation and allocation was already common although this work applied it to a coordinated multi-robot structure. Park et al. [42] also presented the concept of using an approximate platform model estimator, which can be a useful tool supporting platform control performance if the estimation proves of sufficiently accurate. Various facets of this multi-robot collaborative system can be further investigated, as already initiated in section 3.2 of this work by searching for fundamental system characteristics and common approaches described more broad in general robotic literature.

Collaboration and coordination in a multi-robot structure is aims to increase some facet of the system's performance to increase overall system performance over the sum of individuals. Decentralized topologies could also support such behaviors while reducing single points of failure, although centralization has other benefits.

Other approaches into formulating an approximate platform model might prove more accurate. This work presents a novel platform model based on a combination of module models that make up the body in section 4.2 and appendix B. It assumes that the individual module models are to some extent representative operating in proximity of surrounding modules in a configuration, or that compensating factors can be formulated, which are both bold assumptions. Hydrodynamic effects on arbitrary shaped platforms will not likely be estimated perfectly anytime soon, but that is also not the goal. However, formulating rules of thumb or approximations that perform to a great extent to reasonably expectable scenarios would be of great benefit. For example, the concept of hydrodynamic added mass from Humphreys and Watkinson [28] is a rather rough simplification of reality, yet it is commonly used in marine control technology as it often provides a great trade-off between simplicity and representativeness. The platform model described in this paper conserves hydrodynamic effects of a module model that show directional dependence, such as hydrodynamic added mass, or dampening models depending on direction of motion. It is important to ask to what extent such models need to be accurate rather than a rough estimate. For future research it is thus suggested to explore benefits of platform model accuracy in use cases of modular marine robotic systems with varying objectives, scenarios and scales. Governing factors affecting vessel platform dynamics can be further investigated. Existing models can be assessed, improved, or new ones can be formed.

A framework of standardization, modularity, interoperability and solution sharing.

There are many problems that have been encountered during the design of the system's the proof of concept in this paper of which engineers will face similar problems in the future, and can waste effort finding suitable solutions for their problems that others already faced. A framework that allows sharing and integration of solutions to challenges will be key in global adoption of automated vessel systems. This would reduce constant re-invention of the wheel would instead allow developers to effectively add development upon existing. ROS is as an impactful standardizer and facilitator of robotic middleware adopted globally. For automated marine systems, an accepted sharing platform for solutions would greatly benefit development, standardization, module reusability interoperability and thus accellerate adoption of marine automation to benefit society. It is recommended to focus on development and adoption of such a shared platform in successive research. This research could investigate if this is best founded from existing middleware, such as a ROS-maritime-control branch, built upon other frameworks or stand-alone. Key functions and tasks can be further defined and categorized as modules and subfunctions reflecting general marine-control science and industry. Different stakeholders could effectively develop interoperable technical solutions with varying incentives. Sales could provide financial gains from developing competitive solutions to subsystems under some licenses, while contributions under an open source licence can create a snowball effect in developments.

Support creation of shared consistent definitions

To avoid misinterpretation as introduced in section 2.1, a clearly defined definition of autonomy in the context of shipping needs to be developed. The International Maritime Organisation is undertaking this. Yet now the interpretation of IMO's proposal is very different than how it is used in other sciences than engineering, which worries me. A strong definition of autonomy that is conceptually different than automation has the possibility to be a very useful term in development of marine technology. These standards developed by IMO are estimated to become leading definitions once the iterative changes come to an end. It is recommended that anyone using related terms keeps an eye on developments of terminology standardization of important organisations such as the IMO, and strive to contribute formulation if able.

Keep considering what to automate

The fact that we can automate something, doesn't mean we should. It is not by definition desirable to make vessels machine controlled. Some times troughout writing this paper the notion was encountered that full vessel system automation is the ultimate solution. However, humans can be great controllers of vessels, generally seen as versatile and having great ability of adaptation to a situation or environment. Machines can also do certain tasks better than humans, where raw calculation jobs can be seen as an extreme example. As control technology advances, division of tasks between humans and machines can slowly shift, as machines hopefully continue to become better at doing some tasks, allowing humans to use their time for other things. Judgement of performance is off course not to be simply based on quality of vessel operation (who can steer the vessel the best?), but also on various other aspects such as financial, social or safety. It is important to keep asking ourselves: "How would we benefit from automation in this situation?", and not automate for the sake of automation. Let humans do what they do good, and likewise for machines.

Bibliography

[1] Kiam Heong Ang, Gregory Chong, and Yun Li. Pid control system analysis, design, and technology. *IEEE transactions on control systems technology*, 13(4):559–576, 2005.

[2] Yara International ASA. Yara birkeland status november 2020, Accessed 2021, October 10. URL `https://www.yara.com/news-and-media/press-kits/yara-birkeland-press-kit/`.

[3] Hashem Ashrafiuon, Kenneth R Muske, and Lucas C McNinch. Review of nonlinear tracking and set-point control approaches for autonomous underactuated marine vehicles. In *Proceedings of the 2010 American Control Conference*, pages 5203–5211. IEEE, 2010.

[4] Baris Bidikli, Enver Tatlicioglu, and Erkan Zergeroglu. Robust dynamic positioning of surface vessels via multiple unidirectional tugboats. *Ocean Engineering*, 113:237–245, 2016.

[5] B Boogmans. Delfia-1* low level control performance optimization. Technical report, Delft University of Technology, 2020.

[6] D. Braganza, M. Feemster, and D. Dawson. Positioning of large surface vessels using multiple tugboats. In *2007 American Control Conference*, pages 912–917, 2007. doi: 10.1109/ACC.2007.4282954.

[7] L Chen. *Cooperative Multi-Vessel Systems for Waterborne Transport*. PhD thesis, Delft University of Technology, 2019.

[8] Linying Chen, Hans Hopman, and Rudy R Negenborn. Distributed model predictive control for cooperative floating object transport with multi-vessel systems. *Ocean Engineering*, 191:106515, 2019.

[9] Linying Chen, Rudy Negenborn, Yamin Huang, and Hans Hopman. Survey on cooperative control for waterborne transport. *IEEE Intelligent Transportation Systems Magazine*, 13(2):71–90, 2020.

[10] David Clarke. The foundations of steering and manoeuvring. In *Proc. of the IFAC Conference on Manoeuvering and Control Marine Crafts, Girona*, pages 10–25. School of Marine Science and Technology, University of Newcastle upon Tyne - UK, 2003.

[11] Jiri de Vos, Robert G Hekkenberg, and Osiris A Valdez Banda. The impact of autonomous ships on safety at sea–a statistical analysis. *Reliability Engineering & System Safety*, 210:107558, 2021.

[12] Zhe Du, Vasso Reppa, and Rudy R Negenborn. Cooperative control of autonomous tugs for ship towing. *IFAC-PapersOnLine*, 53(2):14470–14475, 2020.

[13] Zhe Du, Rudy R Negenborn, and Vasso Reppa. Cooperative multi-agent control for autonomous ship towing under environmental disturbances. *IEEE/CAA Journal of Automatica Sinica*, 8(8):1365–1379, 2021.

[14] Joel Esposito, Matthew Feemster, and Erik Smith. Cooperative manipulation on the water using a swarm of autonomous tugboats. In *2008 IEEE International Conference on Robotics and Automation*, pages 1501–1506. IEEE, 2008.

[15] Odd M Faltinsen. *Hydrodynamics of high-speed marine vehicles*. Cambridge university press, 2005.

[16] Matthew Feemster, Joel Esposito, and Jack Nicholson. Manipulation of large objects by swarms of autonomous marine vehicles. part 1-rotation. Technical report, Naval academy Annapolis MD dept of weapons and systems engineering, 2006.

[17] Matthew G Feemster and Joel M Esposito. Comprehensive framework for tracking control and thrust allocation for a highly overactuated autonomous surface vessel. *Journal of Field Robotics*, 28(1):80–100, 2011.

[18] Amsterdam Institute for Advanced Metropolitan Solutions. Roboat project website, Accessed March 2021. URL `https://roboat.org/`.

[19] International Organiszation for Standardization (ISO). Regulatory scoping exercise for the use of maritime autonomous surface ships (mass). maritime safety committee (msc). 103th session. agenda item 5. International Maritime Organization, 15 march 2021.

[20] T Fossen. Guidance and control of ocean vehicles. john willey & sons. *Inc., New York*, 1994.

[21] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control.* John Wiley & Sons, 2011.

[22] Thor I. Fossen and Ola-Erik Fjellstad. Nonlinear modelling of marine vehicles in 6 degrees of freedom. *Mathematical Modelling of Systems*, 1(1):17–27, 1995. doi: 10.1080/13873959508837004. URL `https://doi.org/10.1080/13873959508837004`.

[23] Banti Gheneti, Shinkyu Park, Ryan Kelly, Drew Meyers, Pietro Leoni, Carlo Ratti, and Daniela Rus. Trajectory planning for the shapeshifting of autonomous surface vessels. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 76–82, 2019. doi: 10.1109/MRS.2019.8901099.

[24] Roderich Groß, Michael Bonani, Francesco Mondada, and Marco Dorigo. Autonomous self-assembly in swarm-bots. *IEEE transactions on robotics*, 22(6):1115–1130, 2006.

[25] Roderich Groß, Elio Tuci, Marco Dorigo, Michael Bonani, and Francesco Mondada. Object transport by modular robots that self-assemble. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2558–2564. IEEE, 2006.

[26] Golnaz Habibi, William Xie, Mathew Jellins, and James McLurkin. Distributed path planning for collective transport using homogeneous multi-robot systems. In *Distributed Autonomous Robotic Systems*, pages 151–164. Springer, 2016.

[27] Jinyeong Heo, Junghoon Kim, Yongjin Kwon, et al. Analysis of design directions for unmanned surface vehicles (usvs). *Journal of Computer and Communications*, 5(07):92, 2017.

[28] DE Humphreys and KW Watkinson. Prediction of acceleration hydrodynamic coefficients for underwater vehicles from geometric parameters. Technical report, NAVAL COASTAL SYSTEMS LAB PANAMA CITY FL, 1978.

[29] The Mathworks Inc. Stepinfo function documentation, Accessed 2021, August 23. URL `https://www.mathworks.com/help/control/ref/lti.stepinfo.html`.

[30] Tor A Johansen and Thor I Fossen. Control allocation—a survey. *Automatica*, 49(5):1087–1103, 2013.

[31] Erkan Kayacan, Shinkyu Park, Carlo Ratti, and Daniela Rus. Learning-based nonlinear model predictive control of reconfigurable autonomous robotic boats: Roboats. In *IROS*, pages 8230–8237, 2019.

[32] Ryan Henderson Kelly. *Algorithms for planning and executing multi-roboat shapeshifting.* PhD thesis, Massachusetts Institute of Technology, 2019.

[33] G Kirchhoff. Über die bewegung eines rotationskörpers in einer flüssigkeit. *J. Reine Ang. Math*, 71:237–281, 1869.

[34] Aristotelis Komianos. The autonomous shipping era. operational, regulatory, and quality challenges. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 12(2), 2018.

[35] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja J Matarić. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

[36] Luis A Mateos, Wei Wang, Banti Gheneti, Fabio Duarte, Carlo Ratti, and Daniela Rus. Autonomous latching system for robotic boats. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7933–7939. IEEE, 2019.

[37] Shuhei Miyashita, Marco Kessler, and Max Lungarella. How morphology affects self-assembly in a stochastic modular robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 3533–3538. IEEE, 2008.

[38] Wa Nzengu, Jérôme Faivre, Ann-Sofie Pauwelyn, Victor Bolbot, Lars Andreas Lien Wennersberg, and Gerasimos Theotokatos. Regulatory framework analysis for the unmanned inland waterway vessel. *WMU Journal of Maritime Affairs*, pages 1–20, 2021.

[39] University of Michigan and Carnegie Mellon. Matlab & simulink control: Pid controller design, Accessed 2021, October 10. URL `https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID`.

[40] Ian O'hara, James Paulos, Jay Davey, Nick Eckenstein, Neel Doshi, Tarik Tosun, Jonathan Greco, Jungwon Seo, Matt Turpin, Vijay Kumar, et al. Self-assembly of a swarm of autonomous boats into floating structures. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1234–1240. IEEE, 2014.

[41] Raja Parasuraman, Thomas B Sheridan, and Christopher D Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000.

[42] Shinkyu Park, Erkan Kayacan, Carlo Ratti, and Daniela Rus. Coordinated control of a reconfigurable multi-vessel platform: Robust control approach. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4633–4639. IEEE, 2019.

[43] James Paulos, Nick Eckenstein, Tarik Tosun, Jungwon Seo, Jay Davey, Jonathan Greco, Vijay Kumar, and Mark Yim. Automated self-assembly of large maritime structures by a team of robotic boats. *IEEE Transactions on Automation Science and Engineering*, 12(3):958–968, 2015.

[44] Svein I Sagatun and Thor I Fossen. Lagrangian formulation of underwater vehicles' dynamics. In *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1029–1034. IEEE, 1991.

[45] Jungwon Seo, Jamie Paik, and Mark Yim. Modular reconfigurable robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:63–88, 2019.

[46] Erik T Smith, Matthew G Feemster, and Joel M Esposito. Swarn manipulation of an unactuated surface vessel. In *2007 Thirty-Ninth Southeastern Symposium on System Theory*, pages 16–20. IEEE, 2007.

[47] Tim Smithers. Autonomy in robots and other agents. *Brain and cognition*, 34(1):88–106, 1997.

[48] TheSocietyofNavalArchitectureandMarineEngineers SNAME. Nomenclature for treating the motion of a submerged body through a fluid. *The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin No*, pages 1–5, 1950.

[49] Elio Tuci, Muhanad HM Alkilabi, and Otar Akanyeti. Cooperative object transport in multi-robot systems: A review of the state-of-the-art. *Frontiers in Robotics and AI*, 5:59, 2018.

[50] Marialena Vagia, Aksel A. Transeth, and Sigurd A. Fjerdingen. A literature review on the levels of automation during the years. what are the different taxonomies that have been proposed? *Applied Ergonomics*, 53:190–202, 2016. ISSN 0003-6870. doi: https://doi.org/10.1016/j.apergo.2015.09.013. URL `https://www.sciencedirect.com/science/article/pii/S0003687015300855`.

[51] Wei Wang, Luis A. Mateos, Shinkyu Park, Pietro Leoni, Banti Gheneti, Fabio Duarte, Carlo Ratti, and Daniela Rus. Design, modeling, and nonlinear model predictive tracking control of a novel autonomous surface vehicle. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6189–6196, 2018. doi: 10.1109/ICRA.2018.8460632.

[52] Wei Wang, Tixiao Shan, Pietro Leoni, David Fernández-Gutiérrez, Drew Meyers, Carlo Ratti, and Daniela Rus. Roboat ii: A novel autonomous surface vessel for urban environments. *arXiv preprint arXiv:2007.10220*, 2020.

[53] Hongxing Wei, Ning Li, Miao Liu, and Jindong Tan. A novel autonomous self-assembly distributed swarm flying robot. *Chinese Journal of Aeronautics*, 26(3):791–800, 2013.

[54] Norbert Wiener. Cybernetics: Or control and communication in the animal and the machine. 1948.

[55] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

A
Scientific report

Vessel-platform automation: Integrating self assembly with configuration dependent control strategies

Bart Boogmans, Vittorio Garofano, Yusong Pang, Rudy Negenborn

November 2, 2021

Abstract

Modular waterborn stuctures, built from a set of smaller connected units, have shown recent developments to automate assembly and collaborative motion control. Future automated fleet control systems are envisioned to show formation of combined structures together with collaborative motion control approaches within a single framework to minimize human interaction over both operations.

A literature survey explored the state of the art on automated formation and control of modular waterborn structures, where the design spectrum has been broadened by characterizing approaches for more general 'reconfigurable' and 'collaborative' robotic systems. From this, a proof of concept has been developed of a modular fleet system performing automated assembly and collaborative motion control of an arbitrary configured structure, taking into account changing dynamics and network topology.

The developed framework proved able to assemble a set of single operating modules into a 3x1 lattice platform structure, with motion control in single vessel and assembled state adapting to the configuration at the time.

An approach on combining two systems (automated assembly & collaborative control) is presented throughout this paper supplemented with various insightsto aid further developments

Index Terms– Automation, Modular Vessel-Platform, Collaboration, Coordination, Motion-Control, Reconfiguration

1 Introduction

Advancing vehicle automation technologies allow, new or improved logistical applications in the maritime sector. Modular vessel systems, consisting of assembled waterborn units can compete to perform a niche of tasks where variable shape, rapid deployment and versitality are key. Structures assembled on water surface into platforms have seen explorative studies towards automation to reduce reliance on operators and potentially outperform them in some aspects. Automation of modular vessel-platforms could aid formation and motion of structures to, for instance carry arbitrary shaped objects or form ad-hoc infrastructure such as temporary bridges. Using the strength of modular, reusable components combined with low reliance on operators due to automation can form competitive solutions operating effectively with low resource consumption to a niche of logistical challenges.

O'hara et al. [8] and Paulos et al. [10] show development of an automated fleet of modules performing individual module motion and platform reconfiguration. The fleet is homogeneous where modules are identical and shaped rectangular at model scale with dimentions in the ratio of standard shipping containers. Modules connect via mechanical hook and rope actuators into a repeating lattice architecture [15]. O'hara et al. [8] approaches reconfiguration deterministically by distincting (1) generation of desired configuration, (2) selecting reconfiguration sequence, (3) positioning a module and (4) perform docking sequence.

Wang et al. [13] present a line of modular fleet systems named Roboat, where Amsterdam institute for Advanced Metropolitan Solutions (AMS) and Massachusetts Institute of Technology (MIT) are major contributors. The module design is based to form a framework upon which further tests can be performed for transportation and self-assembly to floating infrastructures, where model scale experiments pave the road towards full sized implementation [14]. Roboat vessels are used as a use-case by works that focus on various facets of modular vessel platform automation.

Park et al. [9] utilize a platform-centralized approach to control motion ofan arbitrary sized vessel-platform of Roboat modules. Control effort is generated by a PI controller running on one vessel that is elected as coordinator, which subsequently allocates effort between actuators on the modules. This centralized platform control system can be classified in terms of Guidance, Navigation and Control (GNC) [1], where the platform motion-controller is fed reference pose (position and orientation) as input from a planning (guidance) layer and performs a state estimation (navigation). An approximate platform model is estimated based on the number of connected modules to adapt the control-effort generation subsystem to arbitrary configurations.

Mateos et al. [7] shows development of a latching system consisting of male/female ball/cone components for assembling Roboat vessels, discusing latching hardware and experimental results. Gheneti et al. [3] and Kelly [5] present trajectory planning algorythms for reconfiguration of modular surface structure components. The proposed core logic of the shapeshifting algorythm is finding the largest overlap between current and desired configuration. Gheneti et al. [3] show approaches and experimental results of platform shapeshifting, distincting the whole reconfiguration problem into (1) task planning, (2) trajectory planning and (3) trajectory tracking.

Literature survey showed no information on a modular fleet system incorporating automated reconfiguration (Fig. 1) with collaborative configuration dependent control strategies (Fig. 2) in a single framework, thus consequences of integrating the two behaviors in a single system appear unmapped. It would be useful for developers to know what system requirements, characteristics and constraints emerge from such integration, as this information can be used to make well informed, effective, scalable, interoperable and long term beneficial design choices. Gathering and documenting approaches and experiences on integration can pave the road towards effective implementations to fully benefit from automated structure assembly while motion control of the combined platforms are enhanced by effective collaborative approaches.

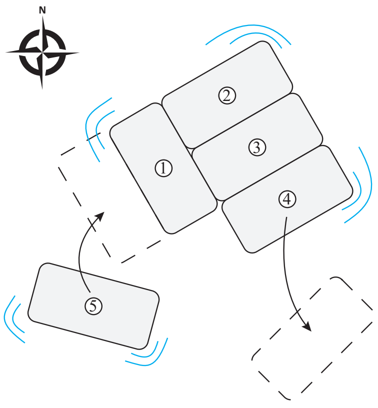


Figure 1: Automatic reconfiguration of a vessel platform. Module (1)-(4) are connected. (4) disassembles. (5) approaches for assembly.

A framework is proposed to control a multi-robot fleet simultaneously performing automated assembly into platforms and collaborative, coordinated motion control using Delfia-1* model scale vessels (Fig. 3). The framework should support arbitrary configurations, as this creates versatility to a wide set of tasks for modular vessel platforms

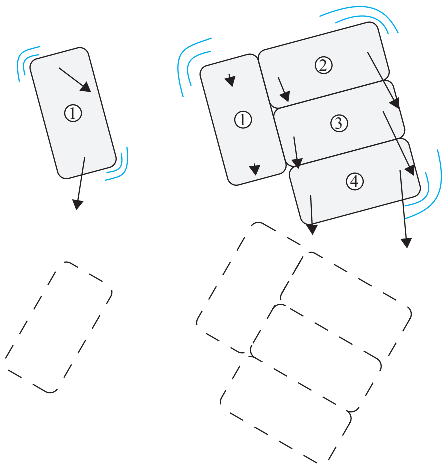


Figure 2: Multi-robot vessel platform motion control collaboration and coordination shown on two configurations. Arrows illustrate forces generated by thrusters.

as a key element supporting commercial competitivity. Solutions are designed to be modular, such that subsystems are conveniently swapped out others (perhaps improved and better performing), stimulating further improvements and reusability of work. The goal of this design is not to optimize an existing system, but to explore a novel combination of behavioral concepts that is expected to be of interest in the near future.

Guidance (including reconfiguration planning), navigation and motion control tasks are distributed to components of a multi-agent network. Motion control structure is centralized per platform, where network-topology varies throughout operation as configurations change. Control effort generation and allocation are distinct subsystems, where the former uses a proportional, integral and differential (PID) control approach adapting to predicted platform dynamics. Platform-models are estimated by combining models of individual modules while taking into account relative placement and orientation. This approach aims to provide a framework for predicting with reasonable accuracy over arbitrary shapes as a quick, cheap and scalable solution with respect to parameter estimation experiments on all configurations.

Design, implementation and evaluation of the developed system is discussed throughout this paper, where section 2 explains control structure, subsystem division and variable network-topology with high level perspective. Section 3 zooms down to discuss design and implementation of each subsystem. Performance of the overall system is evaluated in section 4, after which section 5 concludes.

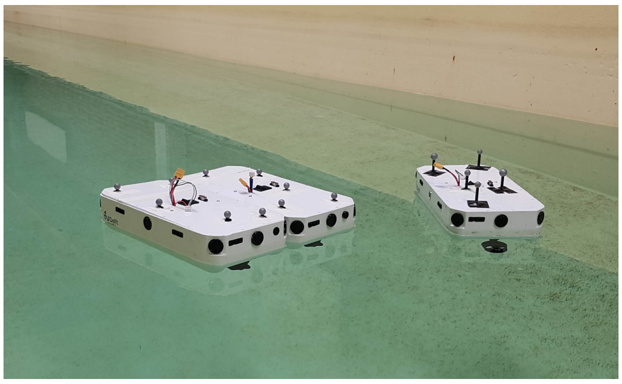


Figure 3: Three Delfia-1* modules from Researchlab Autonomous Shipping (RAS) Delft in the towing tank of TUDelft department of Maritime and Transport Technology (MTT)

2 Multi-Vessel Control Structure

Goal of the developed system is showing simple automation of both reconfiguration and collaborative platform motion control. Distinction of functions according to GNC scheme logic is used to categorize subfunctions. The extra function of assembly planning is integrated withing the guidance layer.

2.1 Network Topology

The control structure operates in three layers, depicted in figure 4. The concept of a 'platform' and a corresponding centralized platform-controller are key to the functioning of this system.

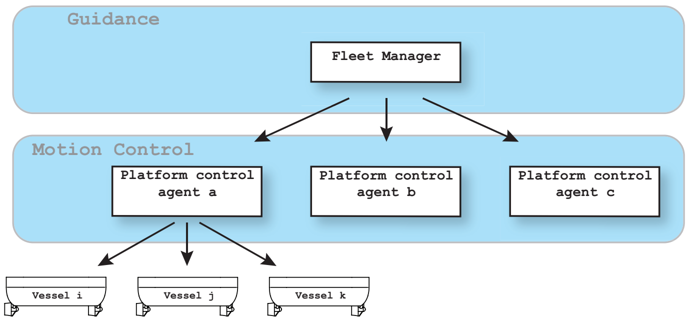


Figure 4: Hierarchical network topology, where platform control agents each manage a unique set of modules.

A guidance layer coordinates motion planning and the assembly processes, which is implemented as a single fleet-manager entity operating as a rather simplistic state machine cycling trough phases due to system triggers or operator input. Other setups for guidance layer funcionality (such as proposed by [8] [10] [3] [5]) could be applied to the approach of integrating collaborative motion control with self-assembly as presented in this paper.

Tasks generated by the fleet manager are passed to Platform-controllers, each managing motion control of a set of connected modules in a centralized fashion. This platform controller uses a reference state and a platform-state estimation to generate actuator responses for all modules which are sent to the corresponding modules. The platform controlling agent can be embodied by a computer anywhere on the network (on a module, on shore or distributed) with sufficient computational power and the network reliability.

The amount of modules that an platform-control-agent manages varies over time as vessels attach or disconnect. Thus all functions of the platform controller need to be able to handle a variable amount of modules and configurations. Changes in configuration affect control structure topology, as module ownership is transferred between platform controlling agents during reconfiguration (for example, the network topology shown in Fig 4 would have ownership of disassembling vessel k move from platform controller a to b).

Communication between agents is facilitated through WiFi and the Robotic Operating System (ROS) as middleware in an interoperable and modular way thanks to its publisher-subscriber protocols, which allows a variable amount of agents to listen (subscribe) and send (publish) to various organised datastreams (topics).

2.2 Multi-agent Control approach

State of vessels are described in line with conventions of SNAME [12] as shown in table 1, and illustrated in figure 5.

DOF	Positions, Orientations	Velocities	Forces, Moments
Surge	x	u	X
Sway	y	v	Y
Heave	z	w	Z
Roll	ϕ	p	K
Pitch	θ	q	M
Yaw	ψ	r	N

Table 1: SNAME vessel state notation

Vessel motion in simplified to three degrees of freedom on the surface plane (x,y and yaw), in which a platform coordinate system $\{p\}$ is defined that has constant placement relative to connected modules (Fig. 6).

Control objective is generated by a guidance system as time varying position with preprogrammed protocols to provide platform motion control layer with a reference state. Platform reference state is noted as

$$x_{ref} = \eta_{p,ref} = \begin{bmatrix} \mathbf{p}_p^n \\ \Theta_{np} \end{bmatrix} = \begin{bmatrix} x_p^n \\ y_p^n \\ \Psi_p^n \end{bmatrix} \tag{1}$$

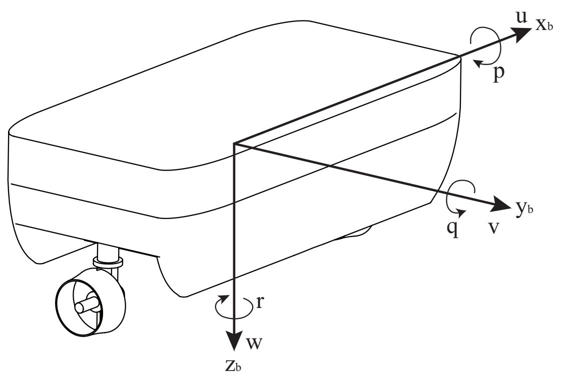


Figure 5: Six degrees of motion of a Delfia vessel expressed with respect to the body-fixed coordinate system $\{b\}$

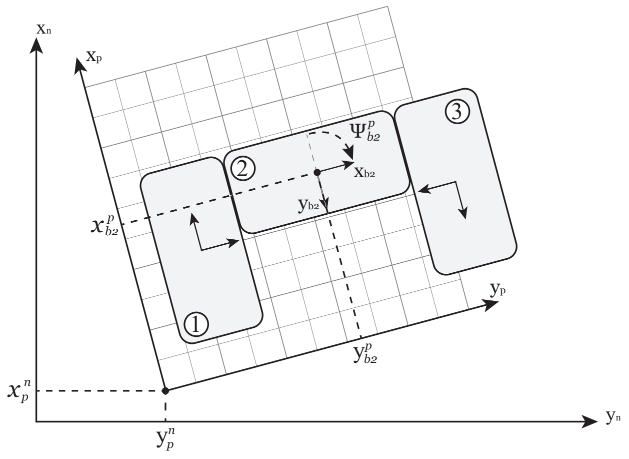


Figure 6: An assembly of three connected vessels, illustrating interpretation of global ($\{n\}$), platform ($\{p\}$) and module ($\{b\}$) coordinate systems.

where η_p, ref is a vector describing vessel pose, including positions in vector $\mathbf{p}^n_p$ and angles in vector Θ_{np} of which elements are shown for three DOF surface motion. Platform motion control is based on state feedback, with the control loop illustrated in Fig. 7. Platform state is estimated by trasposing measured position of modules according to module placement within the platform coordinate system. Control effort generation and allocation are distinct where the former uses an estimated model to adapt actuator behavior to maintain performance while the system dynamics change.

Used modules are rectangular Delfia-1* vessels, equipped with two rotating azimuth thrusters, forming a homogeneous fleet. Delfia-1* modules have two axes of symmetry in hull shape, weight distribution and thruster placement. The vessel's body fixed coordinate system origin is defined in the middle, where planes of symmetry coincide.

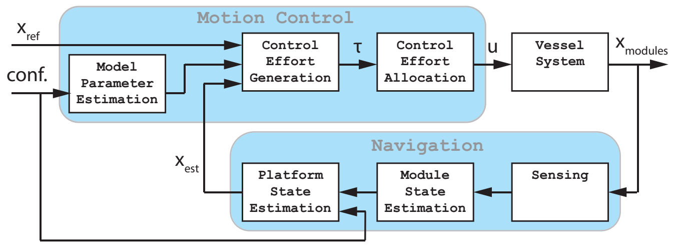

Motion Control
x_ref
conf.
Model Parameter Estimation
Control Effort Generation
τ
Control Effort Allocation
u
Vessel System
x_modules
Navigation
x_est
Platform State Estimation
Module State Estimation
Sensing

Figure 7: A schematic showing the platform motion control feedback loop.

3 Subsystems

System components as described in Sec. 2 and illustrated in Fig 4 and 7 each had a solution picked as explained here.

3.1 Guidance & Navigation

Modules assemble in lattice formation, using active magnets to remain configured. Task execution, including motion guidance and assembly operates in a static preprogrammed phasewise fashion. Automated configuration of a 3x1 configured platform is illustrated in Fig. 8, where connection phases consist of (1) initial line-up, (2) movement to connection site, (3) evaluation of succesful assembly by operator, (4) change of control topology to match new configuration and (5) adjust platform control behavior to new configuration.

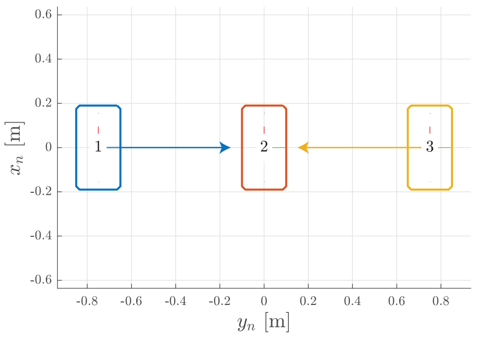


Figure 8: Simple platform assembly protocol from top vieuw. Vessel 1 and 3 are lined up to approach connecting to vessel 2 on either side. Once connectors are within range, the magnets snap into place, fixing relative motion.

Early test yielded modules floating around the connection site unconstrained in three degrees of motion, resulting in unpractically long connection times. This was solved by using normal forces on the modules' hulls to align surfaces, being a common trick used by human controlled ships (such as docking ferries). Fig. 9 shows how references of approaching modules overlapping with docking station (in this case another module) yield a small normal force between vessels, ensuring that they keep making contact, and avoid rotation. Reconfigured platforms are controlled by a

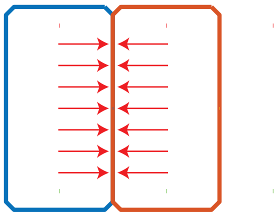


Figure 9: Normal forces that result from overlapping references during assembly.

single platform-controller with new configuration, network topology and control approach to reach motion-control objectives of logistical tasks as given by the guidance layer.

The navigation system aims to estimate the platform state, which is actually but a concept to represent a collection of modules and is not directly measurable. It is however estimated by using a feedback signal of module positions aquired over through an on-shore optical tracking system, and module's known, constant placement within the body. An OptiTrack$^{\mathrm{TM}}$ optical motion tracking system uses ceiling mounted cameras and infra red reflectors (grey balls visible on top of Delfias in Fig. 3), to localize modules. Consider an update of the position and orientation of a module as

$$\eta^n_{b/n} = \begin{bmatrix} \mathbf{p}^n_{b/n} \\ \Psi_{b/n} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^n_{b/n} \\ \mathbf{y}^n_{b/n} \\ \Psi_{b/n} \end{bmatrix} \tag{2}$$

where $\eta^n_{b/n}$ is the pose of body (module) b with respect to the origin of inertial frame $\{n\}$, expressed in $\{n\}$. The placement of all modules (defined by the position and orientation of module's body fixed coordinate system $\{b\}$) within the assembly is known and can be described with respect to $\{p\}$ expressed in $\{p\}$ as

$$\eta^p_{b/p} = \begin{bmatrix} \mathbf{p}^p_b \\ \Theta_{pb} \end{bmatrix} \tag{3}$$

altough, instead of euler angles Θ_{pb}, the rotation matrix $\mathbf{R}^p_b$ from coordinate system $\{b\}$ to $\{p\}$ can also be used to express relative orientation. Interpretation of pose expressed in platform frame is illustrated in Fig. 6. Measured module pose is transformed to platform pose by subsequent rotation and translation. For the three considered degrees of surface plane motion the platform state becomes

$$\eta^n_{p/n} = \begin{bmatrix} \mathbf{p}^n_{p/n} \\ \Psi_{p/n} \end{bmatrix} \tag{4}$$

where

$$\Psi_{p/n} = \Psi_{b/n} - \Psi_{b/p} \tag{5}$$

and

$$\mathbf{p}^n_{p/n} = \mathbf{p}^n_{b/n} - \mathbf{p}^n_{b/p} = \mathbf{p}^n_{b/n} - \mathbf{R}(\Psi_{p/n})\mathbf{p}^p_{b/p} \tag{6}$$

3.2 Model parameter estimation

The control approach uses the concept of an approximate platform-model to then influence motion control behavior similar to Park et al. [9], yet also taking into account platform shape besides number of connected modules. Module-connectivity and dynamical models of individual modules are known, and are used to form a single platform model assuming rigid connections. Use-cases where vessel assemblies are formed in many varying configurations can make experimental model parameterization infeasible for all reasonably forseeable configurations, where predicient of a dynamical model by combining models of modules can provide a quick, cheap and scalable solution.

A dynamic model of the platform is formed by expressing all models of the modules in identical generalized coordinates that describe platform state. By describing module motion and forces in coordinate system and origin of $\{p\}$, resultants can be summed. It is shown how terms from multiple module models can be grouped for convenient expression, and how subsequently the combined structure's centre of mass can be found.

Models of modules are expressed in platform frame origin by (1) translating the expressions to o_p as a reference point and (2) rotating the expressions to match coordinate system $\{p\}$. This approach works also for models of which the origonal inertial matrix is not defined in the CG, and/or for models that have directional dependent mass (e.g., hydrodynamic added mass).

Generalized positions and velocities of the platform are described respectively as

$$\eta^n_{p/n} = \begin{bmatrix} \mathbf{p}^n_p \\ \Theta_{np} \end{bmatrix} \tag{7}$$

$$\nu^p_{p/n} = \begin{bmatrix} \mathbf{v}^p_{p/n} \\ \omega^p_{p/n} \end{bmatrix} \tag{8}$$

The assembly is considered rigid, thus there is no motion between modules, making relative velocity of a module expressed in rotating frame $\{p\}$ zero (if derivative is taken in rotating, non-inertial frame$\{b\}$ or $\{p\}$)

$$\nu^p_{bi/p} = \nu^p_{bi/bj} = \begin{bmatrix} \mathbf{v}^p_{bi/p} \\ \omega^p_{bi/p} \end{bmatrix} = 0 \tag{9}$$

and

$$\frac{d}{dt}\mathbf{R}^p_b = 0 \tag{10}$$

Velocities and generalized forces can be converted to vector notation as

$$\nu^b_{b/n} = \begin{bmatrix} \mathbf{v}^b_{b/n} \\ \omega^b_{b/n} \end{bmatrix} = \mathbf{J}^b_p \mathbf{H}(\mathbf{p}^p_{b/p}) \nu^p_{p/n} \tag{11}$$

$$\begin{aligned} \tau^p_p = \begin{bmatrix} \mathbf{f}^p_p \\ \mathbf{m}^p_p \end{bmatrix} &= \begin{bmatrix} \mathbf{R}^p_b \mathbf{f}^b_b \\ \mathbf{R}^p_b(\mathbf{m}^b_b + \mathbf{p}^b_{p/b} \times \mathbf{f}^b_b) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}^p_b & 0 \\ 0 & \mathbf{R}^p_b \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{S}(\mathbf{p}^b_{p/b}) & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}^b_b \\ \mathbf{m}^b_b \end{bmatrix} \\ &= \mathbf{J}^p_b \mathbf{H}^\top(\mathbf{p}^b_{p/b}) \tau^b_b \end{aligned} \tag{12}$$

where coordinate system transformation between rotated frames $\{p\}$ and $\{b\}$ is done by operator

$$\mathbf{J}^p_b = \begin{bmatrix} \mathbf{R}^p_b & 0 \\ 0 & \mathbf{R}^p_b \end{bmatrix}, \quad \mathbf{J}^{p\top}_b = \begin{bmatrix} \mathbf{R}^p_b & 0 \\ 0 & \mathbf{R}^p_b \end{bmatrix} \tag{13}$$

and translation of forces is represented by operator [1]

$$\begin{aligned} \mathbf{H}^\top(\mathbf{p}^b_{p/b}) &= \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{S}(\mathbf{p}^b_{p/b}) & \mathbf{I} \end{bmatrix} \\ \mathbf{H}(\mathbf{p}^b_{p/b}) &= \begin{bmatrix} \mathbf{I} & -\mathbf{S}(\mathbf{p}^b_{p/b}) \\ 0 & \mathbf{I} \end{bmatrix} \end{aligned} \tag{14}$$

and $\mathbf{S}(\mathbf{x})$ is the 3x3 cross product matrix operator. Motion of an individual rigid body can be described in $\{b\}$ as [1]

$$\mathbf{M}\dot{\nu}^b_{b/n} + \mathbf{C}(\nu^b_{b/n})\nu^b_{b/n} = \tau_{res} \tag{15}$$

where $\mathbf{M}$ represent inertia of the rigid body and constant hydrodynamic added mass, $\mathbf{C}(\nu^b_{b/n})\nu^b_{b/n}$ represent coriolis and centripetal forces and τ_{res} are the resultant forces on the body.

Coriolis and centripetal forces arise due to the rotation of $\{b\}$ with respect to the inertial frame, and are fully determined by the inertial matrix. Fossen [1] shows how an energy approach, using Kirchhoff's equations is a convenient way to find the coriolis and centripetal matrix. If kinetic energy of vessel and added mass is written in quadratic form Kirchhoff [6]

$$\mathbf{T} = \frac{1}{2} \nu^{b\ \top}_{b/n} \mathbf{M}^b \nu^b_{b/n} \tag{16}$$

where inertial matrix and velocities are described in $\{b\}$, and inertial matrix $\mathbf{M}^b$ represent rigid body inertia and hydrodinamic added mass. Substituting Eq. 11 gives

$$\mathbf{T} = \frac{1}{2} \nu^{p\ \top}_{p/n} \mathbf{H}^\top(\mathbf{p}^p_{b/p}) \mathbf{J}^{b\ \top}_p \mathbf{M}^b \mathbf{J}^b_p \mathbf{H}(\mathbf{p}^p_{b/p}) \nu^p_{p/n} \tag{17}$$

Which can be rewritten as

$$\mathbf{T} = \frac{1}{2} \nu^{p\ \top}_{p/n} \mathbf{M}^p \nu^p_{p/n} \tag{18}$$

where the inertial matrix of a module is expressed in platform coordinates as

$$\mathbf{M}^p = \mathbf{H}^\top(\mathbf{p}^p_{b/p}) \mathbf{J}^{b\ \top}_p \mathbf{M}^b \mathbf{J}^b_p \mathbf{H}(\mathbf{p}^p_{b/p}) \tag{19}$$

Equation 18 can be substituted in Kirchhoff's vector equations [6]

$$\frac{d}{dt}\left[\tfrac{\partial \mathbf{T}}{\partial \nu_1}\right] + \mathbf{S}(\nu_2)\frac{\partial \mathbf{T}}{\partial \nu_1} = \tau_1 \tag{20}$$

$$\frac{d}{dt}\left[\tfrac{\partial \mathbf{T}}{\partial \nu_2}\right] + \mathbf{S}(\nu_2)\frac{\partial \mathbf{T}}{\partial \nu_2} + \mathbf{S}(\nu_1)\frac{\partial \mathbf{T}}{\partial \nu_1} = \tau_2 \tag{21}$$

where $\nu_1 = \mathbf{v}^p_{p/n}$, $\nu_2 = \omega^p_{p/n}$, $\tau_1 = \mathbf{f}^p_p$ and $\tau_2 = \mathbf{m}^p_p$ to obtain the equations of motion of a module expressed in platform coordinates. Notice that the expression of inertial matrix in Eq. 19 is constant, due to rigid body assumptions. This allows formation of the (tranlated and rotated) dynamical model in a 'normal' fashion, such as shown in Fossen [1], where terms that are not dependent on accelleration, but on velocity are grouped to form the coriolis-centripetal matrix, which can be represented in many forms. Various works describe options parameterizations such as skew-symmetric Sagatun and Fossen [11] or velocity independent Fossen and Fjellstad [2], which can be chosen to best suit a project.

Forces on modules can be summed to find resultant force on the complete platform, given that they are expressed in the same point and coordinate system. If n connected modules generate a generalized force τ_{bi} expressed in the same point and coordinate system, the forces can be added to find the total force for the entire platform as

$$\tau^p_p = \sum_{i=1}^{n} \tau^p_{bi} \tag{22}$$

Which can allow convenient reformulation of a total model. Grouping of terms allows expression of 'platform inertia', 'total dampening' or 'total control-effort', to name only some. For instance, expressing inertia of various modules in a the same platform coordinates allows us to express total platform-inertia

$$\begin{aligned} \mathbf{M}^p_p &= \sum_{i=1}^{n} \mathbf{M}^p_{bi} \\ &= \sum_{i=1}^{n} \mathbf{H}^\top(\mathbf{p}^p_{bi/p}) \mathbf{J}^{bi\ \top}_p \mathbf{M}^{bi}_{bi} \mathbf{J}^{bi}_p \mathbf{H}(\mathbf{p}^p_{bi/p}) \end{aligned} \tag{23}$$

where $\mathbf{M}^p_p$ is the platform inertial matrix, $\mathbf{M}^p_{bi}$ is the inertial matrix of module i expressed in $\{p\}$. This expression can conveniently be used to compute terms regarding coriolis and centripetal forces for the complete platform in one go as one would do with a single vessel, instead of obtaining it by summating the coriolis and centripetal components of all modules.

Other forces from individual modules, such as to dampening, can be integrated to a platform model similar to accelleration dependent coefficients as shown, by expressing in the same point and coordinate system.

From the expression of total platform inertial tensor as Eq. 23 we can find the centre of gravity of the platform. Recall that the centre of gravity is the point of a rigid object, if a (gravitational) force is applied, this force creates no resultant torque, and thus no angular accelleration. This effectively means that if we find the position of platform centre of gravity $\mathbf{p}_g$, and express our platform model in that point (similar as in Eq. 19), no coupling between rotation and translation should exist in the inertial matrix. Expressing the inertial matrix in CG_p can be done by:

$$\mathbf{M}_p^{CG} = \mathbf{H}^\top(\mathbf{p}_{g/p}^p)\mathbf{M}_p^p\mathbf{H}(\mathbf{p}_{g/p}^p) \tag{24}$$

No coupling in $\mathbf{M}_p^p$ between rotation and translation means that the off-diagonal quadrants are zero, thus if

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \tag{25}$$

then

$$\mathbf{M}_{12}^{CG} = \mathbf{M}_{21}^{CG} = 0_{3x3} \tag{26}$$

evaluating the resulting upper right quadrant of Eq. 24 and 26 gives

$$\mathbf{M}_{CG,12} = \mathbf{M}_{p,12} - \mathbf{M}_{p,11}\mathbf{S}(\mathbf{p}_g^p) = 0_{3x3} \tag{27}$$

$$\mathbf{S}(\mathbf{p}_g^p) = \mathbf{M}_{p,11}{}^{-1}\mathbf{M}_{p,12} \tag{28}$$

which allows us to easily extract the center of mass for the combined structure by substituting known inertial parameters and solving for $\mathbf{p}_g^p$.

It can be debated whether the summation of individual ship dynamic models sufficiently represents overall stucture dynamics. Humphreys and Watkinson [4] stated (about individually operating vessels) that forces and moments represented by acceleration hydrodynamic coefficients can, to a very great extent, be modeled as potential flow phenomena. Yet platforms have vessels operating in very close proximity, which might significantly affect the boundary layer size and shapes. This can affect terms representing added mass and dampening both positively and negatively.

3.3 Control effort generation

The control block is based on a Proportional-Integrator-Differential (PID) controller, designed to scale to estimated model parameters. Three parralel controllers are used to control each individual degree of freedom, as illustrated in 10

Control gains of the control-effort-generation block are designed to scale such that, once tuned for a single configuration, show similar behavior for any other configuration or platform size. For this, the following configuration dependent parameters are used.

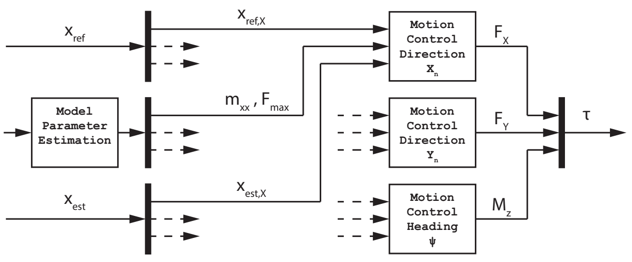

X_{ref}
$X_{ref,X}$
Model Parameter Estimation
m_{xx} , F_{max}
X_{est}
$X_{est,X}$
Motion Control Direction X_n
Motion Control Direction Y_n
Motion Control Heading ψ
F_X
F_Y
M_z
τ

Figure 10: Parralel platform motion controller setup showing signal flow to motion control block for x_n direction, parsing reference (X_{ref}), model parameters and state (X_{est}) to generate desired control effort (τ)

The centre of mass of the platform is found using Eq. 28. It can then be substituted in Eq. 24 to express the equations of motion in that particular point. For three degrees of freedom in the surface plane this becomes shaped as

$$\mathbf{M}_p^{CG} = \begin{bmatrix} m_{xx} & m_{xy} & 0 \\ m_{yx} & m_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{29}$$

where

$$\mathbf{M}_{11} = \begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix} \qquad \mathbf{M}_{12} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{M}_{12} = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad \mathbf{M}_{22} = I_{zz}$$

Which shows the estimated moment of inertia I_{zz} in the bottomright corner. If modules have hydrodynamic added mass being modelled as a constant, direction-dependent constant, the off-diagonal elements m_{xy} and m_{yx} may be nonzero. This can also result that masses in xx and yy direction may be unequal, which can feel rather counter intuitive, as this is never the case with normal rigid body motion. The cause of this still originates from the origonal form of module inertia matrix, as this inherited by using such models.

The controllers responsible for linear motion will use an estimation of the mass of the platform. Rotating the inertial matrix can be done such that the off diagonal elements m_{xy} and m_{yx} become zero. The magnitude of the diagonal elements can be easily found, as they are the eigenvalues of M_{11}. The average of the eigenvalues is used as the estimated omni-directional mass for adapting controller behavior. For linear motion in 2 degrees of freedom (x and y) this becomes

$$m_p \approx \frac{1}{2} \sum Eig(\mathbf{M}_{11}) \tag{30}$$

As the fleet utilizes rotatable azimuth thrusters, the maximum force is generated by the propellers on full power in a single direction. The maximum force that the platform can generate can be found by summation of maximum thruster force of all thrusters

$$\mathbf{f}_{p,max} = \sum_{i=1}^{n_{thr}} f_{i,max} \tag{31}$$

where n_{thr} refers to the total amount of thrusters, and $f_{i,max}$ is the maximum force that the ith propeller can supply. The homogeneous fleet has two identical propellers per module, such that.

$$\mathbf{f}_{p,max} = 2 * n * f_{prop,max} \tag{32}$$

Maximum torque is generated when all thrusters supply maximum force in the direction perpendicular to a vector between CG and the thruster. For a single vessel, this becomes

$$\mathbf{m}_{i,max} = |\mathbf{r}| f_{i,max} = |\mathbf{p}^p_{CG/p} - \mathbf{p}^p_{thr,i/p}| f_{i,max} \tag{33}$$

where $\mathbf{p}^p_{CG/p}$ is the position vector of the platform CG and $\mathbf{p}^p_{thr,i/p}$ is the position of the ith thruster. The latter is usually given in local frame of a module, but can be converted to platform coordinates by matrix rotation and a translation as:

$$\mathbf{p}^p_{thr,i/p} = \mathbf{R}^p_{bj} \mathbf{p}^{bj}_{thr,i/bj} + \mathbf{p}^p_{bj/p} \tag{34}$$

where $\mathbf{p}^{bj}_{thr,i/bj}$ is the position of thruster i, mounted on module j, expressed in the body fixed coordinate system of module j. $\mathbf{p}^p_{bj/p}$ is the position of module j expressed in platform coordinate system. Summation of Eq. 33 over all modules yields total maximum torque generated by actuators for a given configuration as

$$\mathbf{m}_{p,max} = \sum_{i=1}^{2n} \mathbf{m}_{i,max} \tag{35}$$

It should be noted that the Eq. 32 and 35 show absolute maxima, which need full participation of all actuators to be reached. These maxima can not be obtained in different degrees of motion simultaneously, as outputs will be saturated. To avoid unpredictable control effort generation, actuator operation near output saturation is avoided during implementation.

The control gains of the three parralel PID controllers are tuned to a single reference configuration. As the configuration changes the control gains will adapt to the newly estimated dynamics. The control gain scaling is based on the assumption that a configuration will have a response comparable to the reference, but in a different time scale. Typical errors that are fed into PID gains will thus be of comparable magnitude. System responses of current and reference configurations are thus aimed to be approximately comparable such that

$$\eta_c(t) \approx \eta_{ref}(C * t) \tag{36}$$

If control forces are a dominant factor to the response time of a step input on the system, a characteristic accelleration of a configuration can be defined as

$$a = \frac{Force}{Inertia} \tag{37}$$

This is used to estimate the time scaling factor is estimated as the ratio of maximum accelleration of a configuration with respect to the reference configuration.

$$C_c = \frac{a_c}{a_{ref}} = \frac{F_c \, I_{ref}}{F_{ref} \, I_c} \tag{38}$$

where a_c and a_{ref} are the characteristic accellerations of the current and reference configuration respectively.

All gains is designed to scale to the maximum obtainable control effort. The base value of contribution is determined in the gain tuning process of the reference configuration. The eventual output of any proportional integral and derivative gains is multiplied by the maximum control effort in that dimention.

Output of the control effort from proportional gain scales to the magnitude of the error, which assumed comparable in all configurations. This could result in, for example, a proportional gain that is to contribute 60% of the maximum obtainable control effort at an error of $e = 1.0$. The gain would become

$$K_p = K_{p,base} * \tau_{max} = 0.6 * \tau_{max} \tag{39}$$

such that the control effort contributed by the proportional block becomes

$$\tau_{i,prop} = e * K_p = 0.6 * \tau_{max} \tag{40}$$

Integral control is however affected by the time in which the system responds. A system that responds slower ($C_c < 1$) than the reference configuration will encounter additional integrator buildup. Time factor C_c compensates for change of integrator output due to response time by adjusting integral gain as

$$K_i = C_c \, K_{p,base} \, \tau_{max} \tag{41}$$

such that integral control output becomes

$$\tau_{i,int} = K_i \int_0^t e \, dt = C_c \, K_{p,base} \, \tau_{max} \int_0^t edt \tag{42}$$

Derivative control output is also affected by time, but scales inversely to time factor C_c with respect to integral control. Imagine, for example, a mass (*cough* *cough* vessel) that approaches the reference state, which would make the time derivative of the error negative. Derivative control would attempt to slow the mass down as it approaches it's desired state to avoid overshoot. An object, such as a container vessel, with low maximum control forces relative to the large mass would have to use take this speed more serious than highly manouverable vessels. Derivative adapts to a configuration as

$$K_i = \frac{K_{d,base}\ \tau_{max}}{C_c} \tag{43}$$

Base values of controller gains ($K_{p,base}$,$K_{i,base}$,$K_{b,base}$) will be set while revieuwing responses of the system in reference configuration. A PID controller can be manually tuned, or with the help of many tools such as automated PID tuning software. Linear motion in x and y direction will have identical control settings, as dependency on the orientation of reference frame $\{n\}$ is considered undesirable.

3.4 Control effort allocation

Control effort, as shown in the previous section, needs to be allocated onto the platform's actuators. The amount of thrusters available varies per configuration. Also placement and orientation of actuators can differ. A platform needs to be able to sufficiently control it's motion in all reasonably forseeable configurations. As the amount of different configurations is rather large, a general solution is used, that can solve the control effort allocation problem for all possible configurations. The designed control effort allocation protocol relies on the following main principle: "The contribution of an actuator to a desired resulting force or moment is proportional to its ability to contribute relative to that of the combined set of actuators."

This principle manifests particularly in rotational motion, as the ability of a thruster to generate torque relies on its placement with respect to the centre of gravity. Linear motion turns out not to exhibit such dependencies, as all thrusters are equal in strength, and possible orientation. To compute actuator commands that satisfy the desired control effort, it is allocated in each degree of freedom individually and finally combined.

Collaborative distribution of control effort was aimed to perform in a coordinated fashion. Fig.11 shows various independent and combined modes of motion in 3x1 assembled configuration, illustrating how the control allocation protocol assigns tasks to actuators using the described method.

4 Results

4.1 Assembly

Performance of reconfiguration will be quantified by evaluating the change in relative pose between neighbouring modules in all considered degrees of freedom. Pose of vessel j with respect to vessel i expressed in the body fixed coordinate system of vessel i is expressed as:

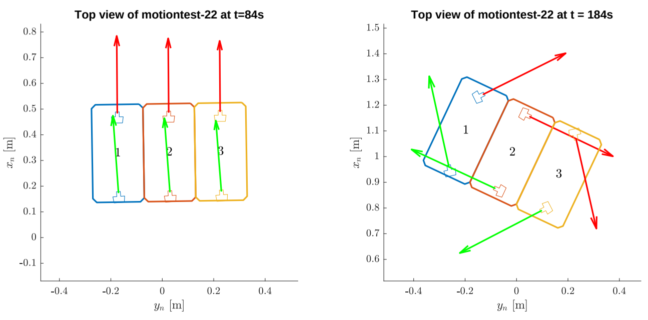


Figure 11: Control effort allocated to create (almost) pure force in x direction (left) and torque (right).

$$\eta^i_j = \begin{bmatrix} \mathbf{p}^i_j \\ \Theta_{ij} \end{bmatrix} = \begin{bmatrix} x^i_j \\ y^i_j \\ \Psi^i_j \end{bmatrix} \tag{44}$$

Assembly succes or failure was easily detectable by eye, but can be supported quantitatively by evaluating relative motion. The two modules attempted assembly simultaneously which is shown in Fig.12.

The signals showing relative positions in Fig.13 show forms of plateauing behavior, which is particularly interesting. The slope of these signals becoming a plateau means that relative speed has suddenly become near zero. This is considered to be due to physical contact between hulls. This can be a bump, soft contact, or a connector snapping two modules together. Relative pose of both connecting modules show some similarities. Pose in various DOFs show plateaus where the speed suddenly becomes near zero, which are most clear in y direction for both modules. Furhermore, a clear moment can be observed on which relative motion becomes near zero for not one, but all degrees of motion, indicating a succesful connection.

Figure 13 illustrate how the relative motion was interpreted for module 1 at two different timestamps. At $t = 308s$ rapid descelleration, is visible in y direction due to hull contact, while vessel 1 is still misaligned. Some motion and aligning occurs until relative motion suddenly halts, around $t = 364s$. This shows the connector performing to restrain relative motion in configured state, as all degrees of motion come to a halt instantaneous.

To further quantify the remaining perceived motion after assembly of module 1, the dataset is evaluated from the moment of connecting (approximately at $t = 364$) and 10 seconds thereafter.

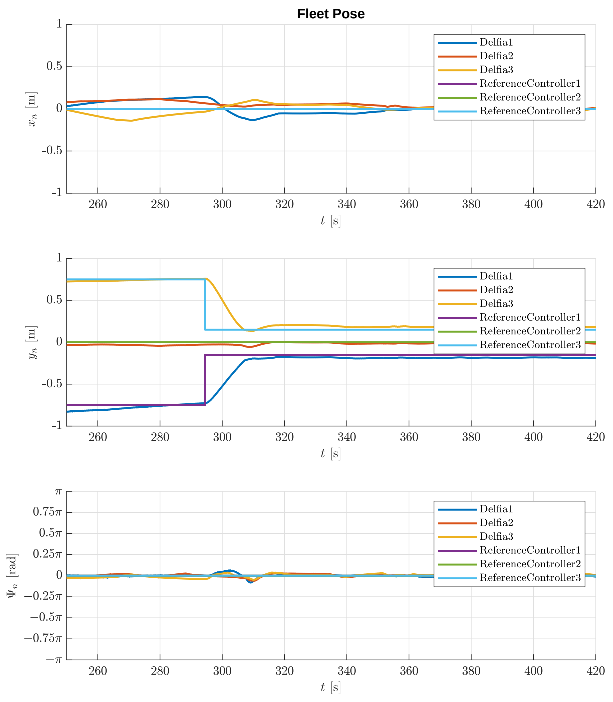


Figure 12: Module position plotted versus time, during the platform assembly stage. Vessels are initially lining up side-by-side, until they approach due to changin y reference values.

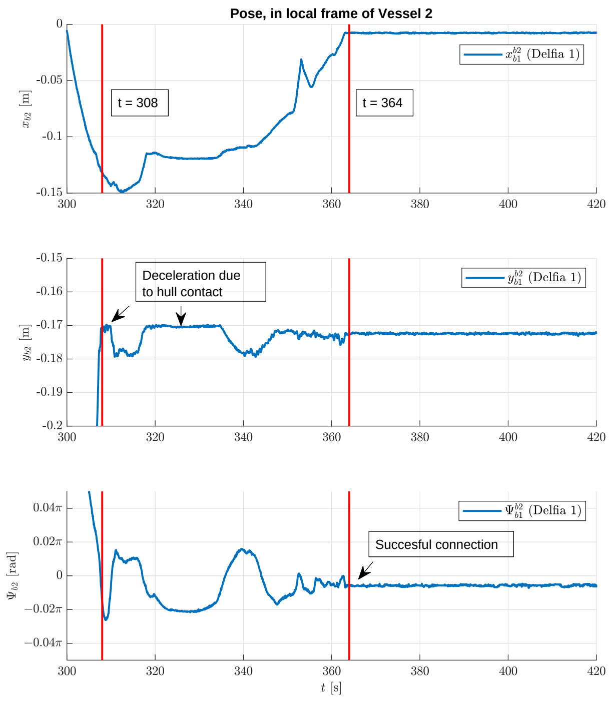


Figure 13: EvaluationConnectTwoInstancesAllDofs

		Average	Minimum	Maximum	maxAmpl	Variance
x	[m]	-0.0076628	-0.0082233	-0.0071825	0.0010408	4.3663e-08
y	[m]	-0.17236	-0.17288	-0.17198	0.00089951	2.6465e-08
Ψ	[rad]	-0.018446	-0.02136	-0.014622	0.006738	1.0167e-06

Table 2: Relative motion of module 1 with respect to body-fixed frame of module 2 for $364 < t < 374$.

4.2 Motion control

To evaluate the motion control performance of the developed framework, vessel system will be tasked to follow a time varying reference signal with step changes. Performance quantification is done by expressing rise-time and settling-time and overshoot of step responses to reference position of the platform in all considered degrees of freedom. Thoughout experiments steps in the platform's reference pose are given in one degree of freedom at a time.

Correctness of using the assumption of the used timescale can be evaluated by comparing response times of reference and assembled configuration. This will show to what extent the system maintains similar behavior over changing configurations, which will give indicators how it might behave for other configurations.

Firstly, the system is required to perform automated vessel platform reconfiguration. This criteria is reached as has been observed throughout operation (by seeing magnet connectors snap into place), yet this is quantitatively supported by expressing relative module motion. Fig.13 illustrates a typical sudden stop in relative vessel motion in all degrees of freedom indicating succesful connection. Assembly of the final developed system behaved consistently throughout testing. Table 2 shows that the sensed vessel motion after connecting is approximately zero.

Secondly, the system needs to perform motion control in a collaborative manner, where multiple robots work together to achieve a single goal. Centralizing platform control to a single entity facilitated modules contributing to reaching one objective and allowed coordination strategies to increase actuator usage effectiveness. Fig.15 displays responses to reference step changes of a 3x1 lattice configurated platform, showing how the developed system controls to converging and stabilizing at reference state.

The designed framework theoretically supports motion control of any arbitrary configuration, as whas desired. Experiments were conducted on two quite general shapes that both showing satisfactory responses. The control approach adapts to unknown configurations by using scaling rules aimed to be representative over a wide variety of shapes and sizes. Although scaling rules have been designed relying on physics, used assumptions need to be re-evaluated for

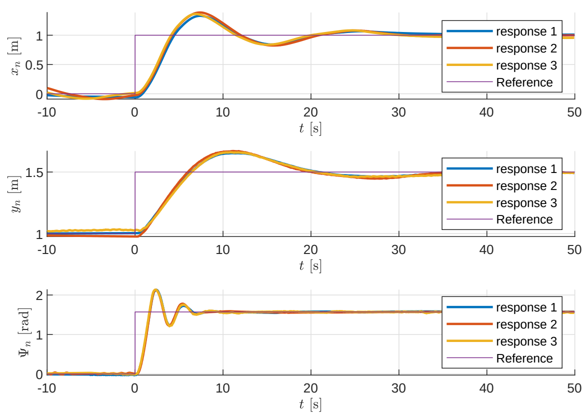


Figure 14: Step responses of single Delfia configuration. Responses are gathered by changing reference of a single degree of motion (x y and yaw) at a time. Three datasets are shown, which show that the responses are rather constant.

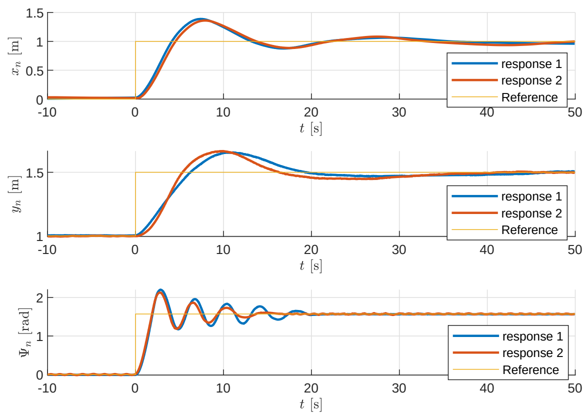


Figure 15: Step responses of a 3x1 lattice configuration. Responses are gathered by changing reference of a single degree of motion (x y and yaw) at a time. Two step responses are shown for each degree of motion.

	Inertia	Maximum input
Single module: Translation	$4.1275kg$	$0.4320N$
Single module: Rotation	$0.1410kgm^2$	$0.0670N*m$
3x1 Conf. Translation	$12.3825kg$	$1.2960N$
3x1 Conf. Rotation	$0.7091kgm^2$	$0.2596Nm$

Table 3: Approximate scaling of reference and assembled configuration

	Rise Time $[s]$	Settling Time $[s]$	Overshoot $[\%]$
Single module	2.5876	27.132	36.0721
3x1 conf.	2.9232	38.2453	37.437
Ratio	1.1297	1.4096	1.0378

Table 4: Comparison between step responses of reference (single vessel) and adapted (3x1) configuration of linear motion in x direction

correctness of future configurations, particularly with more extreme shapes and scales.

5 Conclusion

The discussed criteria on behaviors of self-reconfiguration and collaborative control simultaneously within the same framework are considered achieved. It is concluded that a modular vessel platforming system can be equipped with features automating reconfiguration and collaborative and coordinated control, where the following major challenges are identified as follows;

Firstly, both individual behaviors can already be implemented in a wide variety of approaches, where integration of the two widens the design spectrum significantly more. Both behaviors can be considered complex by themselves, similarly increasing in a combined framework. This complexity toughens challenges of design, but also in facets realizing robustness of a multi-layered system.

Secondly, rarlier developed works showed iterated views on solutions to achieving the desired behaviors. It was found that a main factor for succesful integration is ensuring that both behaviors are interoperable. This particularly means that platform motion control systems need to be able to adapt in real time to configuration changes.

5.1 Discussion

All solutions for subsystems aimed to let the system perform in a lab setting with a fleet of Delfia-1* modules, yet it must be said that these design choices are not optimized or guaranteed to perform on use cases with a different fleet, scale, environment and goals. Replacing or improving components of the system is stimulated by making the control framework as modular as possible. Various subsystems can be further developed, which was expected as exploration of a novel combination of behaviors was the goal rather than optimization. That said, the reader is invited to use the approach and description of implementation for inspiration to develop waterborne multi-vessel collaborative systems.

The approximated platform model relies on the assumptions that either module models remain representative, or that compensating terms can be found. Accelleration dependent contributions of hydrodynamic forces are commonly modelled as constant but inequal in all directions, yielding satisfactory accuracies in most conventional ship use cases [4]. Is questionable to what extent such constant added mass terms remain an accurate model in the scenario of vessel assembly where modules operate in unconventionally close proximity. If (at least some part of) the hydrodynamic forces on a module are accurately represented by this constant directional dependent added mass, then the orientation and placement of that module should be taken into account when estimating contribution to the assembled structure for which the proposed approach on estimating platform model is suitable.

5.2 Recommendations

Firstly, engineers will face similar problems in the future as encountered throughout this work. A framework that allows sharing and integration of solutions to challenges will be key in global development and adoption of automated vessel systems. This would reduce constant re-invention of the wheel would instead allow developers to effectively add development upon existing. ROS is as an impactful standardizer and facilitator of robotic middleware adopted globally. For automated marine systems, an accepted sharing platform for solutions would greatly benefit development, standardization, module reusability interoperability. It is recommended to focus on development and adoption of such a shared platform in successive research built on existing middleware such as a ROS, on others or stand-alone. Key functions and tasks can be further defined and categorized as modules and subfunctions reflecting general marine-control science and industry. Different stakeholders could effectively develop interoperable technical solutions with varying incentives.

Secondly, the amount of works presenting approaches of controlling arbitrary configured modular marine structures is limited, as only Park et al. [9] was found adressing that particular challenge. Many design choices from the system developed in this work are similar to this earlier work, such as adoption of platform-level centralized control topology. Distinction between control effort generation and allocation was already common although this work applied it to a coordinated multi-robot structure. Park et al. [9] also presented the concept of using an approximate platform model estimator, which can be a useful tool supporting platform control performance if the estimation proves of sufficiently accurate. Various facets of this multi-robot collaborative system can be further investigated by searching for fundamental characteristics and common approaches described more broad in general robotic literature.

Collaboration and coordination in a multi-robot structure is aims to increase some facet of the system's performance to increase overall system performance over the sum of individuals. Decentralized topologies could also support such behaviors while reducing single points of failure, although centralization has other benefits.

This work presents a novel platform model based on a combination of module models, assuming that the individual module models are to some extent representative operating in proximity of surrounding modules in a configuration, or that compensating factors can be formulated. Factors that affect accuracy of the combined model need to be investigated. Further developing rules of thumb or approximations performing to reasonably expectable scenarios would be benificial. This can function similar as the concept of hydrodynamic added mass from Humphreys and Watkinson [4], a simplification, yet commonly used in marine control technology providing a great trade-off between simplicity and representativeness. The platform model described in this paper conserves hydrodynamic effects of a module model that show directional dependence, such as hydrodynamic added mass, or dampening models depending on direction of motion. It is important to ask to what extent such models need to be accurate rather than a rough estimate. For future research it is thus suggested to explore benefits of platform model accuracy in use cases of modular marine robotic systems with varying objectives, scenarios and scales. Governing factors affecting vessel platform dynamics can be further investigated. Existing models can be assessed, improved, or new ones can be formed.

References

[1] Fossen, T. I. (2011). *Handbook of marine craft hydrodynamics and motion control.* John Wiley & Sons.

[2] Fossen, T. I. and Fjellstad, O.-E. (1995). Nonlinear modelling of marine vehicles in 6 degrees of freedom. *Mathematical Modelling of Systems*, 1(1):17–27.

[3] Gheneti, B., Park, S., Kelly, R., Meyers, D., Leoni, P., Ratti, C., and Rus, D. (2019). Trajectory planning for the shapeshifting of autonomous surface vessels. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 76–82.

[4] Humphreys, D. and Watkinson, K. (1978). Prediction of acceleration hydrodynamic coefficients for underwater vehicles from geometric parameters. Technical report, NAVAL COASTAL SYSTEMS LAB PANAMA CITY FL.

[5] Kelly, R. H. (2019). *Algorithms for planning and executing multi-roboat shapeshifting.* PhD thesis, Massachusetts Institute of Technology.

[6] Kirchhoff, G. (1869). Über die bewegung eines rotationskörpers in einer flüssigkeit. *J. Reine Ang. Math*, 71:237–281.

[7] Mateos, L. A., Wang, W., Gheneti, B., Duarte, F., Ratti, C., and Rus, D. (2019). Autonomous latching system for robotic boats. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7933–7939. IEEE.

[8] O'hara, I., Paulos, J., Davey, J., Eckenstein, N., Doshi, N., Tosun, T., Greco, J., Seo, J., Turpin, M., Kumar, V., et al. (2014). Self-assembly of a swarm of autonomous boats into floating structures. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1234–1240. IEEE.

[9] Park, S., Kayacan, E., Ratti, C., and Rus, D. (2019). Coordinated control of a reconfigurable multi-vessel platform: Robust control approach. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4633–4639. IEEE.

[10] Paulos, J., Eckenstein, N., Tosun, T., Seo, J., Davey, J., Greco, J., Kumar, V., and Yim, M. (2015). Automated self-assembly of large maritime structures by a team of robotic boats. *IEEE Transactions on Automation Science and Engineering*, 12(3):958–968.

[11] Sagatun, S. I. and Fossen, T. I. (1991). Lagrangian formulation of underwater vehicles' dynamics. In *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1029–1034. IEEE.

[12] SNAME, T. (1950). Nomenclature for treating the motion of a submerged body through a fluid. *The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin No*, pages 1–5.

[13] Wang, W., Mateos, L. A., Park, S., Leoni, P., Gheneti, B., Duarte, F., Ratti, C., and Rus, D. (2018). Design, modeling, and nonlinear model predictive tracking control of a novel autonomous surface vehicle. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6189–6196.

[14] Wang, W., Shan, T., Leoni, P., Fernández-Gutiérrez, D., Meyers, D., Ratti, C., and Rus, D. (2020). Roboat ii: A novel autonomous surface vessel for urban environments. *arXiv preprint arXiv:2007.10220*.

[15] Yim, M., Shen, W.-M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., and Chirikjian, G. S. (2007). Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52.

B An approach to deriving dynamics of a combined naval structure from components' models

This document describes an approach to form a dynamical model of a naval stucture (or 'assembly', 'platform') built up from a set of assembled modules (vessels). Dynamical models of individually operating modules are first expressed in the same body fixed reference point and coordinate system, which are then used to formulate a single model for the entire assembly.

$n = [1, 2, 3, ...]$ modules are fixed to eachother, such that they form a rigid assembly. There exists a body fixed (platform) reference frame $\{p\}$. Each module has a body fixed reference frame, refered to as $\{b1\}, \{b2\}, \{b3\}, ..., \{bi\}, ...\{bn\}$, where i refers to the i-th vessel, or simply $\{b\}$ if only a single module is considered. $\{n\}$ refers to the global frame which is assumed inertial.

Dynamical models exist for each individual vessel defined in the body fixed reference frame of the respective module, such as the example model given in equation B.34 where inertia is assumed constant, yet possibly direction-dependent due to hydrodynamic added mass.

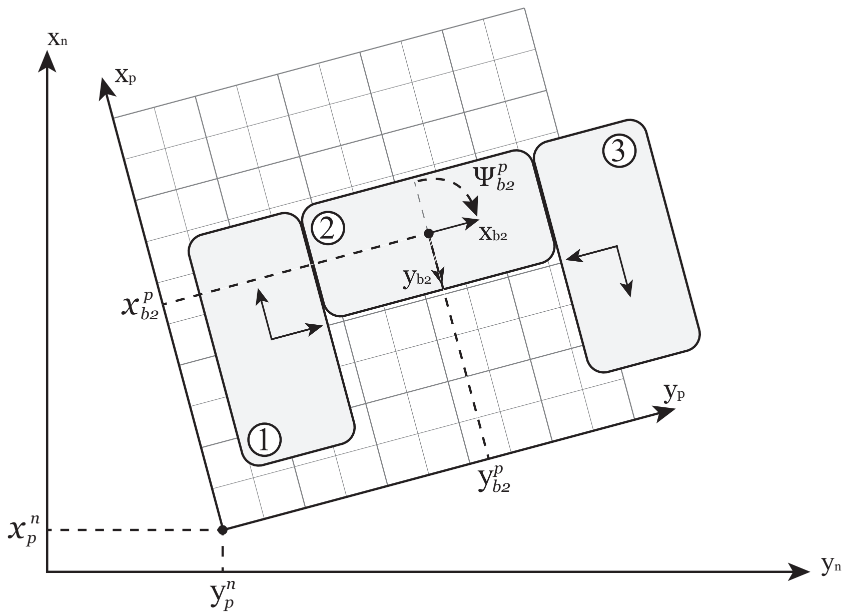


Figure B.1: An assembly of three connected vessels, of which an expression of the entire dynamical model is desired. Interpretation of global ($\{n\}$), platform ($\{p\}$) and module ($\{b\}$) coordinate systems is illustrated. For example, the position and orientation of module 2 expressed in $\{p\}$), and the position of $\{p\}$ expressed in $\{n\}$ is shown.

The approach presented here uses a known vessel model and (1) translates the expressions to another reference point and (2) rotates the expressions to match another coordinate system. This approach works also for models of which the origonal inertial matrix is not defined in the CG, and/or for models that have directional dependent mass (e.g., hydrodynamic added mass)

A dynamic model of an assembly (or platform) of modules is calculated by expressing all modules in the same point and coordinate system. It was shown how terms from multiple module models can be grouped for convenient expression, and how the centre of mass of the combined structure can be found.

We first start by describing module motion and forces in another point and frame of reference. For the purpose of platform dynamics estimation, this is done in position o_p expressed in platform coordinate system $\{p\}$. Then we follow up by combining models of a set of connected modules to express platform dynamics.

Generalized positions and velocities of the platform are described as, respectively

$$\eta^n_{p/n} = \begin{bmatrix} \mathbf{p}^n_p \\ \Theta_{np} \end{bmatrix} \tag{B.1}$$

$$\nu^p_{p/n} = \begin{bmatrix} \mathbf{v}^p_{p/n} \\ \omega^p_{p/n} \end{bmatrix} \tag{B.2}$$

The placement of all vessels (defined by the position and orientation of coordinate system $\{b\}$) within the assembly is known and can be described with respect to $\{p\}$ expressed in $\{p\}$ as

$$\eta^p_{b/p} = \begin{bmatrix} \mathbf{p}^p_b \\ \Theta_{pb} \end{bmatrix} \tag{B.3}$$

altough, instead of euler angles Θ_{pb} to express relative orientation, the rotation matrix from coordinate system $\{b\}$ to $\{p\}$ is more often used

$$\mathbf{R}^p_b \tag{B.4}$$

The assembly and the placement of the platform frame are considered rigid, thus there is no motion between them, such that relative velocity of a module expressed in rotating frame $\{p\}$ equals (if derivative is taken in rotating frame)

$$\nu^p_{bi/p} = \nu^p_{bi/bj} = \begin{bmatrix} \mathbf{v}^p_{bi/p} \\ \omega^p_{bi/p} \end{bmatrix} = 0 \tag{B.5}$$

and

$$\frac{d}{dt}\mathbf{R}^p_b = 0 \tag{B.6}$$

B.1. Translating and rotating expressions of motion

We would like to dectibe motion of the modules with respect to $\{n\}$ in terms of platform position and velocies $\nu^p_{p/n}$ and $\eta^p_{b/p}$. Thus, we are looking to describe the following motion in the local frame of a module i

Linear velocity of $\{b\}$ with respect to $\{n\}$ expressed in body fixed frame $\{b\}$	$\mathbf{v}^b_{b/n}$
Angular velocity of $\{b\}$ with respect to $\{n\}$ expressed in body fixed frame $\{b\}$	$\omega^b_{b/n}$
Linear accelleration of $\{b\}$ with respect to $\{n\}$ expressed in body fixed frame $\{b\}$	$\dot{\mathbf{v}}^b_{b/n}$
Angular accelleration of $\{b\}$ with respect to $\{n\}$ expressed in body fixed frame $\{b\}$	$\dot{\omega}^b_{b/n}$

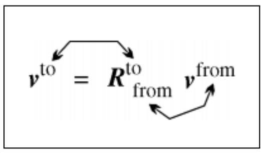
for which the following terms can be used to form the expression

Position of module $\{b\}$ w.r.t. $\{p\}$ expressed in $\{p\}$	$\mathbf{p}^p_{b/p}$
Orientation of $\{b\}$ w.r.t. $\{p\}$ as a rotation matrix	$\mathbf{R}^p_b$
Translational velocity of assembly w.r.t. $\{n\}$ expressed in $\{p\}$	$\mathbf{v}^p_{p/n}$
Angular velocity of assembly w.r.t. $\{n\}$ expressed in $\{p\}$	$\omega^p_{p/n}$
Translational accelleration of assembly w.r.t. $\{n\}$ expressed in $\{p\}$	$\dot{\mathbf{v}}^p_{p/n}$
Angular accelleration of assembly w.r.t. $\{n\}$ expressed in $\{p\}$	$\dot{\omega}^p_{p/n}$

A vector $\mathbf{v}$ expressed in coordinate system i can be expressed in coordinate system j by multiplication with the rotation matrix $\mathbf{R}$ between systems as

$$\mathbf{v}^j = \mathbf{R}^j_i \mathbf{v}^i \tag{B.7}$$

according to the notation convention shown in figure B.2



$\boldsymbol{v}^{\text{to}} = \boldsymbol{R}^{\text{to}}_{\text{from}}\, \boldsymbol{v}^{\text{from}}$

Figure B.2: Fossen [21]'s convention of notating a rotation matrix between coordinate systems.

B.1.1. Translation

A vector describing a position with respect to o_p, o_n and o_b relate as

$$\mathbf{p}_{b/n} = \mathbf{p}_{p/n} + \mathbf{p}_{b/p} \tag{B.8}$$

Expressing in $\{n\}$ gives

$$\mathbf{p}^n_{b/n} = \mathbf{p}^n_{p/n} + \mathbf{p}^n_{b/p} \tag{B.9}$$

where

$$\mathbf{p}^n_{b/p} = \mathbf{R}^n_p \mathbf{p}^p_{b/p} \tag{B.10}$$

differentiating equation B.9 gives

$$\mathbf{v}^n_{b/n} = \mathbf{v}^n_{p/n} + \mathbf{v}^n_{b/p} \tag{B.11}$$

where

$$\mathbf{v}^n_{p/n} = \mathbf{R}^n_p \mathbf{v}^n_{p/n} \tag{B.12}$$

and, using equation B.5 yields

$$\begin{aligned} \mathbf{v}^n_{b/p} &= \frac{d}{dt}(\mathbf{R}^n_p \mathbf{p}^p_{b/p}) \\ &= \frac{d}{dt}(\mathbf{R}^n_p)\mathbf{p}^p_{b/p} + \mathbf{R}^n_p \frac{d}{dt}(\mathbf{p}^p_{b/p}) \\ &= \mathbf{R}^n_p \omega^p_{p/n} \times \mathbf{p}^p_{b/p} \\ &= \mathbf{R}^n_p \mathbf{S}(\omega^p_{p/n})\mathbf{p}^p_{b/p} \end{aligned} \tag{B.13}$$

where $\mathbf{S}(\lambda)$ represents the cross product $\lambda\times$ of $\lambda = [\lambda_1, \lambda_2, \lambda_3]\top$ in the form of a skew symmetric matrix. Note how $\mathbf{v}^n_{b/p}$ can be nonzero even though $\{b\}$ is fixed to $\{p\}$, since $\{b\}$ and $\{p\}$ are not inertial but rotating frames.

Differentiation of equation B.11 yield

$$\dot{\mathbf{v}}^n_{b/n} = \frac{d}{dt}(\mathbf{v}^n_{p/n}) + \frac{d}{dt}(\mathbf{v}^n_{b/p}) \tag{B.14}$$

Where

$$\begin{aligned} \frac{d}{dt}\mathbf{v}^n_{p/n} &= \frac{d}{dt}(\mathbf{R}^n_p\mathbf{v}^p_{p/n}) \\ &= \frac{d}{dt}(\mathbf{R}^n_p)\mathbf{v}^p_{p/n} + \mathbf{R}^n_p\frac{d}{dt}(\mathbf{v}^p_{p/n}) \\ &= \mathbf{R}^n_p\mathbf{S}(\omega^p_{p/n})\mathbf{v}^p_{p/n} + \mathbf{R}^n_p\dot{\mathbf{v}}^p_{p/n} \end{aligned} \tag{B.15}$$

and (using equation B.5)

$$\begin{aligned} \frac{d}{dt}(\mathbf{v}^n_{bi/p} &= \frac{d}{dt}(\mathbf{R}^n_p\mathbf{S}(\omega^p_{p/n})\mathbf{p}^p_{b/p}) \\ = \frac{d}{dt}(\mathbf{R}^n_p)\mathbf{S}(\omega^p_{p/n})\mathbf{p}^p_{b/p} + \mathbf{R}^n_p\frac{d}{dt}(\mathbf{S}(\omega^p_{p/n}))\mathbf{p}^p_{b/p} &+ \mathbf{R}^n_p(\mathbf{S}(\omega^p_{p/n})\frac{d}{dt}(\mathbf{p}^p_{b/p}) \\ &= \mathbf{R}^n_p\mathbf{S}^2(\omega^p_{p/n})\mathbf{p}^p_{b/p} + \mathbf{R}^n_p(\mathbf{S}(\dot{\omega}^p_{p/n})\mathbf{p}^p_{b/p} \end{aligned} \tag{B.16}$$

Equation B.14, B.15 and B.16 yield

$$\dot{\mathbf{v}}^n_{b/n} = \mathbf{R}^n_p[\mathbf{S}(\omega^p_{p/n})\mathbf{v}^p_{p/n} + \dot{\mathbf{v}}^p_{p/n} + \mathbf{S}^2(\omega^p_{p/n})\mathbf{p}^p_{b/p} + \mathbf{S}(\dot{\omega}^p_{p/n})\mathbf{p}^p_{b/p}] \tag{B.17}$$

B.1.2. Rotation

Angular velocity of $\{b\}$ can be expressed in terms of angular velocity relative to the platform, and the angular velocity of the platform with respect to the inertial frame by

$$\omega_{bi/n} = \omega_{p/n} + \omega_{bi/p} \tag{B.18}$$

which can be expressed in inertial frame $\{n\}$ using equation B.5 as

$$\omega^n_{bi/n} = \mathbf{R}^n_p\omega^p_{p/n} \tag{B.19}$$

differentiation yields

$$\begin{aligned} \frac{d}{dt}\omega^n_{bi/n} &= \frac{d}{dt}(\mathbf{R}^n_p\omega^p_{p/n}) \\ &= \frac{d}{dt}(\mathbf{R}^n_p)\omega^p_{p/n} + \mathbf{R}^n_p\frac{d}{dt}(\omega^p_{p/n}) \\ &= \mathbf{R}^n_p\mathbf{S}(\omega^p_{p/n})\omega^p_{p/n} + \mathbf{R}^n_p\dot{\omega}^p_{p/n} \\ &= \mathbf{R}^n_p\dot{\omega}^p_{p/n} \end{aligned} \tag{B.20}$$

as a cross product of a vector with itself equals zero.

B.1.3. Motion of {b} expressed in terms of motion of {p}

Expressing equations B.11, B.17, B.19 and B.20 in $\{b\}$ gives

$$\mathbf{v}^b_{b/n} = R^b_p[\mathbf{v}^p_{p/n} + \mathbf{S}(\omega^p_{p/n})\mathbf{p}^p_{b/p}] \tag{B.21}$$

$$\dot{\mathbf{v}}^b_{b/n} = R^b_p[\mathbf{S}(\omega^p_{p/n})\mathbf{v}^p_{p/n} + \dot{\mathbf{v}}^p_{p/n} + \mathbf{S}^2(\omega^p_{p/n})\mathbf{p}^p_{b/p} + \mathbf{S}(\dot{\omega}^p_{p/n})\mathbf{p}^p_{b/p}] \tag{B.22}$$

$$\omega^b_{b/n} = \mathbf{R}^b_p\omega^p_{p/n} \tag{B.23}$$

$$\dot{\omega}^b_{b/n} = \mathbf{R}^b_p\dot{\omega}^p_{p/n} \tag{B.24}$$

B.2. Translating and rotating forces

A force vector described in reference point o_b expressed in $\{b\}$ consisting of linear forces and moments, described as

$$\tau_b^b = \begin{bmatrix} \mathbf{f}_b^b \\ \mathbf{m}_b^b \end{bmatrix} \tag{B.25}$$

Which is to be expressed in reference point o_p in coordinate system $\{p\}$. Describing forces in reference point o_p is easily done as linear accelleration of a rigid body is not affected by the point where a force is applied, thus

$$\mathbf{f}_p^b = \mathbf{f}_b^b \tag{B.26}$$

Resulting moment in o_p is affected by translation, as $\mathbf{f}_b^b$ can generate additional torque. This is described by the parralel axis theorem, which yields the expression for translated torque

$$\mathbf{m}_p^b = \mathbf{m}_b^b + \mathbf{p}_{p/b}^b \times \mathbf{f}_b^b \tag{B.27}$$

Rotating equation B.26 and B.27 to describe them in $\{p\}$ gives

$$\mathbf{f}_p^p = \mathbf{R}_b^p \mathbf{f}_b^b \tag{B.28}$$

$$\mathbf{m}_p^p = \mathbf{R}_b^p (\mathbf{m}_b^b + \mathbf{p}_{p/b}^b \times \mathbf{f}_b^b) \tag{B.29}$$

Which can be converted to vector notation as

$$\tau_p^p = \begin{bmatrix} \mathbf{f}_p^p \\ \mathbf{m}_p^p \end{bmatrix} = \begin{bmatrix} \mathbf{R}_b^p \mathbf{f}_b^b \\ \mathbf{R}_b^p (\mathbf{m}_b^b + \mathbf{p}_{p/b}^b \times \mathbf{f}_b^b) \end{bmatrix} = \begin{bmatrix} \mathbf{R}_b^p & 0 \\ 0 & \mathbf{R}_b^p \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{S}(\mathbf{p}_{p/b}^b) & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}_b^b \\ \mathbf{m}_b^b \end{bmatrix} = \mathbf{J}_b^p \mathbf{H}^\top (\mathbf{p}_{p/b}^b) \tau_b^b \tag{B.30}$$

where coordinate system transformation between rotated frames $\{p\}$ and $\{b\}$ is done by operator

$$\mathbf{J}_b^p = \begin{bmatrix} \mathbf{R}_b^p & 0 \\ 0 & \mathbf{R}_b^p \end{bmatrix}, \qquad \mathbf{J}_b^{p\top} = \begin{bmatrix} \mathbf{R}_b^p & 0 \\ 0 & \mathbf{R}_b^p \end{bmatrix} \tag{B.31}$$

and translation of forces is represented by operator (Fossen [21])

$$\mathbf{H}^\top (\mathbf{p}_{p/b}^b) = \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{S}(\mathbf{p}_{p/b}^b) & \mathbf{I} \end{bmatrix}, \qquad \mathbf{H}(\mathbf{p}_{p/b}^b) = \begin{bmatrix} \mathbf{I} & -\mathbf{S}(\mathbf{p}_{p/b}^b) \\ 0 & \mathbf{I} \end{bmatrix} \tag{B.32}$$

Equation B.30 can also be similarly applied from $\{p\}$ to $\{b\}$ such that

$$\tau_b^b = \mathbf{J}_p^b \mathbf{H}^\top (\mathbf{p}_{b/p}^p) \tau_p^p \tag{B.33}$$

B.3. Formulating a combined model

A model of an individual module can be described in $\{b\}$ as (Fossen [21])

$$\mathbf{M}\dot{\nu}_{b/n}^b + \mathbf{C}(\nu_{b/n}^b)\nu_{b/n}^b + \mathbf{D}\nu_{b/n}^b = \tau_{res} = \tau_b^b \tag{B.34}$$

where $\mathbf{M}$ represent inertia of the rigid body and constant hydrodynamic added mass. $\mathbf{C}(\nu_{b/n}^b)\nu_{b/n}^b$ represent coriolis and centripetal terms, and $\mathbf{D}\nu_{b/n}^b$ represent forces from linear viscous dampening. Motion in equation B.34 is defined in the origin o_b and coordinate system $\{b\}$ as

$$\nu_{b/n}^b = \begin{bmatrix} \mathbf{v}_{b/n}^b \\ \omega_{b/n}^b \end{bmatrix} \tag{B.35}$$

$$\dot{\nu}^b_{b/n} = \begin{bmatrix} \dot{\mathbf{v}}^b_{b/n} \\ \dot{\omega}^b_{b/n} \end{bmatrix} \tag{B.36}$$

Which can be expressed in terms of platform motion using equation (B.21 : B.24) as

$$\nu^b_{b/n} = \begin{bmatrix} R^b_p[\mathbf{v}^p_{p/n} + \mathbf{S}(\omega^p_{p/n})\mathbf{p}^p_{b/p}] \\ \mathbf{R}^b_p\omega^p_{p/n} \end{bmatrix} \tag{B.37}$$

$$\dot{\nu}^b_{b/n} = \begin{bmatrix} R^b_p[\mathbf{S}(\omega^p_{p/n})\mathbf{v}^p_{p/n} + \dot{\mathbf{v}}^p_{p/n} + \mathbf{S}^2(\omega^p_{p/n})\mathbf{p}^p_{b/p} + \mathbf{S}(\dot{\omega}^p_{p/n})\mathbf{p}^p_{b/p}] \\ \mathbf{R}^b_p\dot{\omega}^p_{p/n} \end{bmatrix} = \tag{B.38}$$

These can both be rewritten as:

$$\nu^b_{b/n} = \mathbf{J}^b_p\mathbf{H}(\mathbf{p}^p_{b/p})\nu^p_{p/n} \tag{B.39}$$

$$\dot{\nu}^b_{b/n} = \mathbf{J}^b_p\mathbf{H}(\mathbf{p}^p_{b/p})\dot{\nu}^p_{p/n} + \mathbf{J}^b_p \begin{bmatrix} \mathbf{S}(\omega^p_{p/n}) & 0 \\ 0 & \mathbf{S}(\omega^p_{p/n}) \end{bmatrix} \mathbf{H}(\mathbf{p}^p_{b/p})\nu^p_{p/n} \tag{B.40}$$

At this point, equation B.33, B.39 and B.40 can be substituted to module models such as equation B.34 to express it in o_p in coordinate system $\{p\}$. Notice how equation B.38 contains elements that are function of either velocity or accelleration. Sorting terms in the substituted expression yield expressions for inertia ($\mathbf{M}\dot{\nu}^p_{p/n}$), coriolis and centripetal forces ($\mathbf{C}(\nu^p_{p/n})\nu^p_{p/n}$) and other modelled terms such as dampening.

However, re-formulation of inertial, coriolis and centripetal forces might be more convenient using an energy approach with Kirchhoff's equations. If kinetic energy of vessel and added mass is written in quadratic form (Kirchhoff [33])

$$\mathbf{T} = \frac{1}{2}\nu^b_{b/n}{}^\top\mathbf{M}^b\nu^b_{b/n} \tag{B.41}$$

where inertial matrix and velocities are described in $\{b\}$, and inertial matrix $\mathbf{M}^b$ contains inertia of the rigid body and added hydrodinamic mass. Substituting B.39 gives

$$\mathbf{T} = \frac{1}{2}\nu^p_{p/n}{}^\top\mathbf{H}^\top(\mathbf{p}^p_{b/p})\mathbf{J}^b_p{}^\top\mathbf{M}^b\mathbf{J}^b_p\mathbf{H}(\mathbf{p}^p_{b/p})\nu^p_{p/n} \tag{B.42}$$

Which can be rewritten as

$$\mathbf{T} = \frac{1}{2}\nu^p_{p/n}{}^\top\mathbf{M}^p\nu^p_{p/n} \tag{B.43}$$

where the inertial matrix of a module is expressed in platform coordinates as

$$\mathbf{M}^p = \mathbf{H}^\top(\mathbf{p}^p_{b/p})\mathbf{J}^b_p{}^\top\mathbf{M}^b\mathbf{J}^b_p\mathbf{H}(\mathbf{p}^p_{b/p}) \tag{B.44}$$

Equation B.43 can be substituted in Kirchhoff's vector equations Kirchhoff [33]

$$\frac{d}{dt}\left[\frac{\partial\mathbf{T}}{\partial\nu_1}\right] + \mathbf{S}(\nu_2)\frac{\partial\mathbf{T}}{\partial\nu_1} = \tau_1 \tag{B.45}$$

$$\frac{d}{dt}\left[\frac{\partial\mathbf{T}}{\partial\nu_2}\right] + \mathbf{S}(\nu_2)\frac{\partial\mathbf{T}}{\partial\nu_2} + \mathbf{S}(\nu_1)\frac{\partial\mathbf{T}}{\partial\nu_1} = \tau_2 \tag{B.46}$$

where $\nu_1 = \mathbf{v}^p_{p/n}$, $\nu_2 = \omega^p_{p/n}$, $\tau_1 = \mathbf{f}^p_p$ and $\tau_2 = \mathbf{m}^p_p$ to obtain the equations of motion of a module expressed in platform coordinates. Notice that the expression of inertial matrix in equation B.44 is constant, due to rigid body assumptions. This allows formation of the (tranlated and rotated) dynamical model in a 'normal' fashion, such as shown in Fossen [21], where terms that are not dependent on accelleration, but on velocity are grouped to form the coriolis-centripetal matrix, which can be represented in many forms. Various works describe options parameterizations such as skew-symmetric **?**] or velocity independent Fossen and Fjellstad [22], which can be chosen to best suit a project.

The assumption of a rigid assembly allows summation of forces on modules in a platform, given that they are expressed in the same point and coordinate system. If n connected modules generate a generalized force τ_{bi} expressed in the same point and coordinate system, the forces can be added to find the total force for the entire platform as

$$\tau_p^p = \sum_{i=1}^{n} \tau_{bi}^p \tag{B.47}$$

Which can allow convenient reformulation of a total model by grouping certain terms. Grouping of terms allows expression of 'platform inertia', 'total dampening' or 'total control-effort', to name only some. For instance, expressing inertia of various modules in a the same platform coordinates allows us to express total platform-inertia

$$\mathbf{M}_{platform}^p = \sum_{i=1}^{n} \mathbf{M}_{bi}^p = \sum_{i=1}^{n} \mathbf{H}^\top(\mathbf{p}_{bi/p}^p)\mathbf{J}_p^{bi\top}\mathbf{M}_{bi}^{bi}\mathbf{J}_p^{bi}\mathbf{H}(\mathbf{p}_{bi/p}^p) \tag{B.48}$$

which can also conveniently be used to compute terms regarding coriolis and centripetal forces for the complete platform in one go, instead of obtaining it by summating the coriolis and centripetal matrices of all modules.

Other forces can be expressed in platform coordinates by substituting equation B.33 and B.37. For example, forces due to linear viscous dampening can be described as

$$\tau_{damp}^b = \mathbf{D}^b \nu_{b/n}^b \tag{B.49}$$

Substitution yields

$$\tau_{damp}^p = \mathbf{J}_p^{b\top}\mathbf{H}^\top(\mathbf{p}_{p/b}^b)\mathbf{D}^b\mathbf{J}_p^b\mathbf{H}(\mathbf{p}_{b/p}^p)\nu_{p/n}^p \tag{B.50}$$

Similar forces acting on the platform, such as dampening, can be grouped.

$$\tau_{damp,total}^p = \sum_{i=1}^{n} \tau_{damp,i}^p \tag{B.51}$$

In the case of linear dampening, these terms result in a constant dampening matrix.

$$\tau_{damp,total}^p = \mathbf{D}_{total}\nu_{p/n}^p \tag{B.52}$$

where

$$\mathbf{D}_{platform} = \sum_{i=1}^{n} \mathbf{J}_p^{bi\top}\mathbf{H}^\top(\mathbf{p}_{p/bi}^{bi})\mathbf{D}^{bi}\mathbf{J}_p^{bi}\mathbf{H}(\mathbf{p}_{bi/p}^p) \tag{B.53}$$

Combining equation B.48, an accompanying expression for coriolis and centripetal forces, and other forces in platform frame yield the expression of overall platform dynamics. This becomes with, for example linear dampening from equation B.52

$$\mathbf{M}_{platform}^p\dot{\nu}_{p/n}^p + \mathbf{C}_{platform}(\nu_{p/n}^p)\nu_{p/n}^p + \mathbf{D}_{platform}\nu_{p/n}^p = \tau_{res}^p \tag{B.54}$$

From the expression of total platform inertial tensor as equation B.48 we can find the centre of gravity of the platform. Recall that the centre of gravity is the point of a rigid object, if a (gravitational) force is applied, this force creates no resultant torque, and thus no angular accelleration. This effectively means that if we find the position of platform centre of gravity $\mathbf{p}_g$, and express our platform model in that point (similar as in equation B.44), no coupling between rotation and translation should exist in the inertial matrix. Expressing the inertial matrix in CG_p can be done by:

$$\mathbf{M}_p^{CG} = \mathbf{H}^\top(\mathbf{p}_{g/p}^p)\mathbf{M}_p^p\mathbf{H}(\mathbf{p}_{g/p}^p) \tag{B.55}$$

No coupling in $\mathbf{M}_{p,CG}^p$ between rotation and translation means that the off diagonal quadrants are zero, thus if

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \tag{B.56}$$

then

$$\mathbf{M}_{12}^{CG} = \mathbf{M}_{21}^{CG} = 0_{3x3} \tag{B.57}$$

evaluating the resulting upper right quadrant of equation B.55 and equation B.57 gives

$$\mathbf{M}_{CG,12} = \mathbf{M}_{p,12} - \mathbf{M}_{p,11}\mathbf{S}(\mathbf{p}_g^p) = 0_{3x3} \tag{B.58}$$

$$\mathbf{S}(\mathbf{p}_g^p) = \mathbf{M}_{p,11}{}^{-1}\mathbf{M}_{p,12} \tag{B.59}$$

which allows us to easily extract the center of mass for the combined structure by substituting known inertial parameters and solving for $\mathbf{p}_g^p$.

B.4. Notes on representativeness of this approach

The approach presented in this paper relies on some assumptions that should be considered

- The combined structure is considered a rigid body. This correctness of this assumption is however affected by assembly size, shape, forces acting on the structure, and connector stiffness. Larger, especially slender structures, or with low connector-stiffness, need to be evaluated whether a rigid body model is appropriate. Alternatively, a multi-body approach can be used, where motion between modules is not neglected. This rigid body approach can however still be used to make an approximation of overall platform dynamics.
- Models of modules are assumed to sufficiently describe the behavior of a model in very close proximity of other modules, or discrepancies due to module proximity are sufficiently negated by additional compensating terms. Constant terms that represent hydrodynamic added mass can be both positively and negatively affected when modules assemble. Additional water can be trapped between vessels to effectively increase added mass, while tight and smooth connection surfaces between modules can decrease the volume of the boundary layer represented by added mass terms. Similarly, other terms of module models, such as dampening or actuator models, should be assessed whether they still sufficiently represent the respective module if operating in close proximity to others. Effects related to close proximity of modules that appear predictable to some extent can be integrated in this approach to increase accuracy of the approximate platform model.

Use-cases where vessel assemblies are formed in many varying configurations can make experimental model parameterization infeasible for all reasonably forseeable configurations. Experimental model parameterization of a configuration will, however, include effects due to the proximity of modules to relatively high accurracy. Predicting a dynamical model by combining models of modules as shown in this document can provide a quick, cheap and scalable solution at reduced accuracy with respect to performing parameter estimation experiments. Accurracy of the combined model depends heavily on the extent that module models are representable in operating range of other modules, or availability of means to estimate additional terms that predict and compensate effects of vessel proximity.

C Implemented Algorithms

For questions regarding sourcode content or access the author wholeheartedly invites the reader to take up contact.

C.1. Matlab Class: Delfia

```
classdef Delfia < Vessel & handle
%Delfia creates a vessel object representing RAS' Delfia-1* model.
    %   The Delfia object encompasses major variables, parameters, functions,
    %   and components that help various tasks related to control automation.
    %
    %   Purpose of this class is including, but not limited to:
    %   - Storing vessel specific data in a structured fashion
    %   - Support connectivity over ROS networks
    %   - Provide Delfia-1* model specific actuator control functions
    %   - Provide display functions
    %
    %   d1 = Delfia(x,y,yaw,'delfiaName')

    % (c) Bart Boogmans 2020/2021 bartboogmans@hotmail.com,
    % You may use, distribute and modify this code under given that clear
    % credit is given to the author's work.
    %
    % Developed in cooperation with the Researchlab Autonomous Shipping
    % Delft & Department Maritime and Transport Technology of faculty 3mE, TU Delft.
    % https://rasdelft.nl/nl/
    %
    % Changelog
    % Last update 29/08/2021: commenting : Bart Boogmans

    properties
        % Actuator states
        thrAngle                        % [radians]     (1x2)   Angles of thrusters
        thrForce                        % [N]           (1x2)   Estimated propulsion force of a thrus
        thrSpd                          % [rounds/s]    (1x2)   Reference speed of thrusters

        % Communication
        actuation_publisher             % [ROS publisher]       Object for sending data to ROS topics
        actuation_subscriber            % [ROS subscriber]      Object for subscribing to ROS topics
        location_publisher
        location_subscriber
    % Multi-vessel platforming related variables
    platf_eta                   % [m,m,radian]  (1x3)   Pose of this vessel in platform

    % Parameters
    r_thr                       % [m]           (2x2)   Location of thrusters in local f
    RPSmin                      % [rounds/s]    (1x1)   Minimum speed of azimuth propell
    RPSmax                      % [rounds/s]    (1x1)   Maximum speed of azimuth propell
    thrFmax                     % [N]           (1x1)   Maximum force of single azimuth
    I_z                         % [kg*m^2]      (1x1)   Estimated moment of inertia arou

    % Display
    plotThr                     % [Bool]        (1x1)   Boolean true if the thruster is
    plotF                       % [Bool]        (1x1)   Boolean true if thruster forces
    forceColor                  % [rgb colormap](2x3)   Color of the force vectors from
    refColor                    % [rgb colormap](1x3)   Color of the reference
    thrOutline                  % [double]      (2xi)   Array with data on the thruster
    forceScale                  % [double]      (1x1)   Scaling factor for force arrow d

    % Other
    nr                          % [integer]     (1x1)   Optional identifier to distingui
end

methods
    % Constructor
    function obj = Delfia(x_,y_,yaw_,name_,~)
        obj = obj@Vessel();
        if nargin > 2
            obj.x(1:3) = [x_;y_;yaw_];
        end
        if nargin == 4
            obj.name = name_;
        end

        % Set parameters specific to Delfia-1* models
        obj.RPSmin = 8;
        obj.RPSmax = 100;
        obj.w = 0.20;
        obj.l = 0.38;
        obj.m = 4;
        obj.r_thr = [-0.155,0;0.155,0];
        [obj.M,~,~,~] = Delfia_Model([0;0;0;0;0;0]);
        obj.thrFmax = 0.2160;          % [N] per thruster at ~120rps. Error is expected t

        % Set initial state for control related variables
        obj.thrAngle = [0,0];
        obj.thrForce = [0,0];
        obj.thrSpd = [0,0];
        obj.platf_eta = [0;0;0];

        % Initiate display settings
        cf = 0.015; % decorative chamfer
        obj.outline = [   obj.l/2      ,obj.l/2-cf   ,-obj.l/2+cf    ,-obj.l/2       ,-c
                                obj.w/2-cf   ,obj.w/2      ,obj.w/2        ,obj.w/2-cf
        obj.thrOutline =   [   0.006,       -0.013,      -0.013,      -0.037,      -0.037,
                                     -0.0075,     -0.0075,     -0.023,      -0.020,      0.02
            obj.plotThr = true;
            obj.plotF = true;
            obj.forceColor = [0,1,0 ; 1,0,0];
            obj.forceScale = 2.0;
        end

        % Destructor
        function delete(obj)
            if ~isempty(obj.actuation_publisher)
                disp(join(['Deleting ',obj.name,' .actuation_publisher ',obj.actuation_publisher.
                delete(obj.actuation_publisher)
            end
            if ~isempty(obj.actuation_subscriber)
                disp(join(['Deleting ',obj.name,' .actuation_subscriber ',obj.actuation_subscribe
                delete(obj.actuation_subscriber)
            end
            if ~isempty(obj.location_publisher)
                disp(join(['Deleting ',obj.name,' .actuation_publisher ',obj.location_publisher.T
                delete(obj.location_publisher)
            end
            if ~isempty(obj.location_subscriber)
                disp(join(['Deleting ',obj.name,' .actuation_publisher ',obj.location_subscriber.
                delete(obj.location_subscriber)
            end
        end

        %% Interaction with ROS
        function ros_sub_pose(obj,message)
            % Processes an update on ROS pose topic
            obj.x(1) = message.Pose.Position.X; % Horizontal, in line with the tank (positive = a
            obj.x(2) = message.Pose.Position.Y; % Horizontal, perpendicular to length of tank (pc
            eul = euler(quaternion([message.Pose.Orientation.W,message.Pose.Orientation.X,message
            obj.x(3) = eul(1); % Range [pi to +pi]
        end

        function ros_sub_actuation(obj,message)
            % Processes an update on ROS actuation topic
            obj.thrSpd = [message.Data(1),message.Data(2)];
            obj.thrAngle = [message.Data(3),message.Data(4)];
            obj.calcThrF;
        end

        function ros_pub_pose(obj,time)
            % Publish object pose on ROS
            msg = rosmessage(obj.location_publisher);
            msg.Pose.Position.X = obj.x(1);
            msg.Pose.Position.Y = obj.x(2);

            quat = eul2quat([obj.x(3),0,0]);
            msg.Pose.Orientation.W = quat(1);
            msg.Pose.Orientation.X = quat(2);
            msg.Pose.Orientation.Y = quat(3);
            msg.Pose.Orientation.Z = quat(4);

            msg.Header.Stamp.Sec = floor(time);
            msg.Header.Stamp.Nsec = floor((time - floor(time))*1000000);
        send(obj.location_publisher,msg)
    end

    function ros_pub_actuation(obj)
        % Publish object actuation on ROS
        msg = rosmessage(obj.actuation_publisher);
        msg.Data = [obj.thrSpd(1),obj.thrSpd(2),obj.thrAngle(1),obj.thrAngle(2)];
        send(obj.actuation_publisher,msg);
    end

    %% Display
    function plotVessel(obj,ax)
        % Displays the outline of the vessel, and thrusters and
        % actuator forces if configured

        R = R2d(obj.x(3));

        % Plot the hull
        plotVessel@Vessel(obj,ax);

        % Plot the thrusters
        if obj.plotThr == true
            for i= 1:length(obj.r_thr)
                thruster_vector = R*(R2d(obj.thrAngle(i))*obj.thrOutline + obj.r_thr(i,:
                plot(ax,thruster_vector(1,:),thruster_vector(2,:),'color',obj.color);
            end
        end

        % plot the force vectors assigned to be applied by the thrusters
        if obj.plotF == true
            for i= 1:length(obj.r_thr)
                thr_pos_n = R*(obj.r_thr(i,:)') + obj.x(1:2);
                f_n = R*(R2d(obj.thrAngle(i))*[obj.thrForce(i);0]);
                quiver(thr_pos_n(1),thr_pos_n(2),f_n(1),f_n(2),obj.forceScale,'LineWidth
            end
        end
    end

    %% Force and speed conversion
    function bound_motor_speeds(obj)
        for i = 1:length(obj.r_thr)
            if (obj.thrSpd(i) <= obj.RPSmin) && (obj.thrSpd(i) >= -obj.RPSmin)
                obj.thrSpd(i) = 0;
            elseif obj.thrSpd(i) >= obj.RPSmax
                obj.thrSpd(i) = obj.RPSmax;
            elseif obj.thrSpd(i) <= -obj.RPSmax
                obj.thrSpd(i) = -obj.RPSmax;
            end
        end
    end

    function calcThrSpd(obj)
        % Converts force to propeller speed and assigns it to object. This is based on c
        for i = 1:length(obj.thrForce)
            obj.thrSpd(i) = obj.thrForce(i)/0.0018;
```

```
            end
        end

        function calcThrF(obj)
            % convert RPS to force and assigns it to object. This is based on coarse measurements
            for i = 1:length(obj.thrForce)
                obj.thrForce(i) = obj.thrSpd(i)*0.0018;
            end
        end
    end
end
```

C.2. Matlab Class: Platform Controller

```
classdef MultiVesselPlatformController < Vessel & handle
%MultiVesselPlatformController controls an arbitrary configured assembly of
%Delfia-1* vessels
    %   An arbitrary amount of vessels (such as the Delfia.m class) can be
    %   registered as connected in this object. This agent will then take
    %   control of the added vessel, assuming that it is rigidly connected to
    %   all other modules, if present. As a result of changing configuration,
    %   various parameters are approximated for the platform including position
    %   of centre of mass, dynamical model, control gains and parameters to
    %   help distribute control effort over all modules.
    %
    % (c) Bart Boogmans 2020/2021 bartboogmans@hotmail.com,
    % You may use, distribute and modify this code under given that clear
    % credit is given to the author's work.

    properties
        % Control structure / hierarchy
        bodies          % array contains handles to vessel objects that are connected or 'ow

        %% Control effort generation
        PIDs            % Array if discrete PID controllers
        x_r             % Reference state
        e               % State error
        t_last_update   % Time of last controller update

        Kxy_0           % PID gains for location control reference value
        Ka_0            % PID gains for heading control reference value

        %% Control effort allocation
        Cm              % Actuator Participation coefficient for moment to achieve: Fi, = Cm
        Cf              % Participation coefficient for force

        %% Configuration dependent parameters

        Mres_max        % Maximum moment that a configuration can attain (linear atm)
        Fres_max        % Maximum force that a configuration can attain (all thrusters on ma

        m_av            % estimated mass, for PID scaling
        I_cg            % estimated moment of inertia around CG

        %% Communication & task scheduling

        refSub          % ROS subscriber to listen to reference updates
        refPub          % ROS publisher to broadcast reference updates
        controlPub      % Publisher to broadcast control related information
        node            % This object's dedicated ROS node

        ixx             % Control update cheduling counter
        nr              % Optional object identifier

        %% Display parameters
        plotref      % Boolean for obj.display to plot reference
        plotvessels  % Boolean for obj.display to plot connected modules

    end
```

```
methods
    % Constructor
    function obj = MultiVesselPlatformController(x_,y_,yaw_,name_,~)
        obj = obj@Vessel(x_,y_,yaw_,name_);
        obj.bodies = Delfia.empty;

        obj.ixx = 0;
        obj.w = 0.5;
        obj.l = 0.5;

        % Default values of control gains that worked for a single vessel
        % which are scale to time constant and maximum control effort
        % see obj.set_PID_gains
        obj.Kxy_0    = [1.0           ,0.0         ,0.0         ];
        obj.Ka_0     = [0.70/(pi/2)   ,0           ,0.15        ];

        obj.PIDs = dPID.empty;
        obj.PIDs(1) = dPID(obj.Kxy_0(1), obj.Kxy_0(2), obj.Kxy_0(3));
        obj.PIDs(2) = dPID(obj.Kxy_0(1), obj.Kxy_0(2), obj.Kxy_0(3));
        obj.PIDs(3) = dPID(obj.Ka_0(1), obj.Ka_0(2), obj.Ka_0(3));
        obj.t_last_update = 0;

        % Display
        obj.outline = [];
        obj.plotref = true;
        obj.plotCG = true;
    end

    % Destructor
    function delete(obj)

        if ~isempty(obj.node)
            disp(join(['Deleting ',obj.name,' .node ',obj.node.Name]))
            delete(obj.node)
        end

        if ~isempty(obj.refSub)
            disp(join(['Deleting ',obj.name,' .refSub ',obj.refSub.TopicName]))
            delete(obj.refSub)
        end

        if ~isempty(obj.refPub)
            disp(join(['Deleting ',obj.name,' .refPub ',obj.refPub.TopicName]))
            delete(obj.refPub)
        end

    end

    %% Adding and removing bodies
    function attachBody(obj,body)
        obj.bodies(length(obj.bodies)+1) = body;
        if ~isempty(obj.node)
            settings = delfiaSettings();
            body.actuation_publisher =ros.Publisher(obj.node,join(['/actuation',body.name]),s
            body.location_subscriber = ros.Subscriber(obj.node,join(['/vrpn_client_node/',bod
    end

    % Adjust agent behavior to a new configuration
    obj.set_configuration_control_parameters;
    obj.adaptControlAllocationGains;
    obj.set_PID_gains;
end

function detatchBody(obj,index)
    delete(obj.bodies(index).location_subscriber);
    delete(obj.bodies(index).actuation_publisher);
    obj.bodies = [obj.bodies(1:index-1),obj.bodies(index+1:length(obj.bodies))];

    % Adjust agent behavior to a new configuration
    obj.set_configuration_control_parameters;
    obj.adaptControlAllocationGains;
    obj.set_PID_gains;
end


%% ROS interaction
function initROSNode(obj)
    % Initiate this object's ROS node
    settings = delfiaSettings();
    obj.node = ros.Node(obj.name,settings.hostname,'NodeHost',settings.myIP);
end

function bodyPoseCallback(obj,~,message,body)
    % Respond to a position update of a connected module

    body.ros_sub_pose(message);
    if strcmp(body.name,obj.bodies(1).name)
        % Only processes location updates of first module for
        % control purposes. Other approaches are possible too.

        obj.ixx = obj.ixx +1;
        if obj.ixx == 3
            obj.ixx = 0;
            % Reduces frequency of control loop by 2/3.
            % Only updates controls once per 3 messages, to reduce
            % frequency from 30 to 10 hz.
            obj.platform_pose_state_estimation(body) % Set platform state
            if ~isempty(obj.x_r)
                t = message.Header.Stamp.Sec + message.Header.Stamp.Nsec* 10^-9;
                obj.run_controls(t);
            end
        end
    end
end

function ros_sub_ref(obj,message)
    obj.x_r(1:3) =  [message.X;message.Y;message.Theta];
end

function ros_pub_ref(obj)
    msg = rosmessage(obj.refPub);
        msg.X = obj.x_r(1);
        msg.Y = obj.x_r(2);
        msg.Theta = obj.x_r(3);
        send(obj.refPub,msg);
    end

    function ros_pub_actuation(obj)
        for vessel = obj.bodies
            vessel.ros_pub_actuation;
        end
    end

    function ros_pub_controlparams(obj)
        if ~isempty(obj.node)
            if ~isempty(obj.node)
                msg = rosmessage(obj.controlPub);
                msg.Data = [obj.thrSpd(1),obj.thrSpd(2),obj.thrAngle(1),obj.thrAngle(2)];
                send(obj.actuation_publisher,msg);
            end
        end
    end

    %% State estimation
    function platform_pose_state_estimation(obj,body)
        p_pnn = body.x(1:2) + R2d(body.x(3))*R2d(-body.platf_eta(3))*-body.platf_eta(1:2);
        psi_pn = body.x(3) -body.platf_eta(3);
        obj.x(1:3) = [p_pnn;psi_pn];
    end

    %% Control planning

    function run_controls(obj,systime)
        dt = systime-obj.t_last_update;

        % calculate errors
        obj.e = obj.calcError(obj.x_r);

        % Run controllers to update desired control effort
        Fx = obj.PIDs(1).run(obj.e(1),dt);
        Fy = obj.PIDs(2).run(obj.e(2),dt);
        Mc = obj.PIDs(3).run(obj.e(3),dt);

        % Allocate control effort
        F_ = R2d(-obj.x(3))*[Fx;Fy]; % convert global input to force vector in local frame
        obj.actuate(F_,Mc);

        % Broadcast module commands over ROS

        if ~isempty(obj.node)
            if ~isempty(obj.controlPub)
                msg = rosmessage(obj.controlPub);
                msg.Data = [obj.e(1),obj.e(2),obj.e(3),F_(1),F_(2),Mc];
                send(obj.controlPub,msg);
            end
        end
    for vessel = obj.bodies
        vessel.calcThrSpd();
        vessel.ros_pub_actuation;
    end

    obj.t_last_update =  systime;
end


% actuate allocates control effort
% Connected modules have their participating actuation set.
function actuate(obj,F,M)
    for vessel = obj.bodies % For each vessel on rigid body
        for i = 1:length(vessel.r_thr) % For each thruster on vessel

            r_pf = R2d(vessel.platf_eta(3))*vessel.r_thr(i,:)' + vessel.platf_eta(1:

            Fm = obj.Cm * M*([-r_pf(2);r_pf(1)]);     % Desired force to do a share i
            Ff = obj.Cf * F;                         % Desired force to do a share in

            F_thr_pf = Fm+Ff;                                 % Thruster force in pl
            F_thr_vf = R2d(-vessel.platf_eta(3))*F_thr_pf;    % Thruster force conve

            %Set the vessel's thruster
            vessel.thrForce(i) = sqrt(F_thr_vf(1)^2 + F_thr_vf(2)^2);
            if vessel.thrForce(i) > 0 % Only adjust thruster angle if it is active
                vessel.thrAngle(i) = angle(F_thr_vf(1)+1i*F_thr_vf(2));
            end
        end

        % Set propeller speeds
        vessel.calcThrSpd;
        vessel.bound_motor_speeds();
    end
end

% Calculate resultant forces of actuator input, given the centre of
% mass has already been identified.
function tau = resultant_control_effort(obj)
    Mres = 0;
    Fres = [0;0];
    for vessel = obj.bodies
        for i = 1:length(vessel.r_thr)
            r_c_pf = R2d(vessel.platf_eta(3))*(vessel.r_thr(i,:)') + vessel.platf_et

            Fthr_vessel = R2d(vessel.thrAngle(i)) * [vessel.thrForce(i);0];
            Fthr_pf = R2d(vessel.platf_eta(3))*Fthr_vessel;

            Mi = cross([r_c_pf',0],[Fthr_pf',0]);
            Mres = Mres + Mi(3);
            Fres = Fres+ Fthr_pf;
        end
    end
    tau = [Fres;Mres];
end
```

```
%% Control strategy adaptation to configuration
function adaptControlAllocationGains(obj)
    % Control allocation gain adaptation
    CmtotInv = 0;
    numActuators = 0;
    for n = 1:length(obj.bodies) % For each vessel on rigid body
        vessel = obj.bodies(n);
        for i = 1:length(vessel.r_thr) % For each thruster on vessel
            r_c = R2d(vessel.platf_eta(3))*vessel.r_thr(i,:)' + vessel.platf_eta(1:2) - c
            CmtotInv = CmtotInv+ r_c(1)^2 + r_c(2)^2;
            numActuators = numActuators +1;
        end
    end
    obj.Cm = 1/CmtotInv;
    obj.Cf = 1/numActuators;
end

function M = findMaxMoment(obj)
    % While aplying a linear moment/r, find Maximum M
    % Find furthest thruster distance to CoM
    r_max = 0;
    for n = 1:length(obj.bodies) % For each vessel on rigid body
        vessel = obj.bodies(n);
        for i = 1:length(vessel.r_thr) % For each thruster on vessel
            r_pf = R2d(vessel.platf_eta(3))*vessel.r_thr(i,:)' + vessel.platf_eta(1:2) -
            r_abs = sqrt(r_pf(1)^2+r_pf(2)^2);
            if r_abs>r_max
                r_max = r_abs;
            end
        end
    end

    % Set all actuators to full moment participation
    for n = 1:length(obj.bodies) % For each vessel on rigid body
        vessel = obj.bodies(n);
        for i = 1:length(vessel.r_thr) % For each thruster on vessel
            r_pf = R2d(vessel.platf_eta(3))*vessel.r_thr(i,:)' + vessel.platf_eta(1:2) -
            %r_pf_abs = sqrt(r_pf(1)^2+r_pf(2)^2);
            Fi = [-r_pf(2);r_pf(1)]/r_max* vessel.thrFmax;
            F_vf = R2d(-vessel.platf_eta(3))*Fi;
            %vessel.thrusterforce_to_actuation(i,F_vf)

            %Set the vessel's thruster
            vessel.thrForce(i) = sqrt(F_vf(1)^2 + F_vf(2)^2);
            if vessel.thrForce(i) > 0 % Only adjust thruster angle if it is active
                vessel.thrAngle(i) = angle(F_vf(1)+1i*F_vf(2));
            end
        end
    end

    % Calculate the Moment of current actuation
    tau = obj.resultant_control_effort;
    M = tau(3);
end
```

```
function F = findMaxForce(obj)
    % find maximum resultant force due to propellors in x,y
    Fsum = 0;
    for n = 1:length(obj.bodies) % For each vessel on rigid body
        vessel = obj.bodies(n);
        for i = 1:length(vessel.r_thr) % For each thruster on vessel
            Fsum = Fsum+vessel.thrFmax;
        end
    end
    F = Fsum;
end


function set_PID_gains(obj)
    % Adapt control gains to current configuration and model
    % parameters

    % Make sure used model parameters are computed
    obj.set_configuration_control_parameters;

    % Estimated time constant for linear motion
    % The time constant is a_current/a_singleVessel
    Fmax_single = 0.4320000;                % N
    m_av_single = 4.1275;                   % Kg ( this value was earlier set to 0.0
    Fmax_current = obj.findMaxForce;        % N
    m_av_current = obj.m_av ;               % Kg
    C_t_xy = (Fmax_current/Fmax_single)*(m_av_single/m_av_current);

    % Estimated time constant for angular motion
    % Time constant is similar, based on maximum angular
    % acceleration ratio
    Mmax_single = 0.066960000000;           % Nm
    I_single = 0.1410;                      % Kg * m^2
    Mmax_current = obj.findMaxMoment;       % Nm
    I_cg_current = obj.I_cg;                % Kg * m^2
    C_t_yaw = (Mmax_current/Mmax_single)*(I_single/I_cg_current);

    % Pid gains are based on time constant with respect to single
    % vessel configuration tuning
    obj.PIDs(1).Kp = obj.Kxy_0(1)*Fmax_current;             % N/m
    obj.PIDs(1).Ki = obj.Kxy_0(2)*Fmax_current*C_t_xy;      % N*s/m
    obj.PIDs(1).Kd = obj.Kxy_0(3)*Fmax_current/C_t_xy;      % N/s/m

    obj.PIDs(2).Kp = obj.Kxy_0(1)*Fmax_current;             % N/m
    obj.PIDs(2).Ki = obj.Kxy_0(2)*Fmax_current*C_t_xy;      % N*s/m
    obj.PIDs(2).Kd = obj.Kxy_0(3)*Fmax_current/C_t_xy;      % N/s/m

    obj.PIDs(3).Kp = obj.Ka_0(1)*Mmax_current;              % N*m /rad
    obj.PIDs(3).Ki = obj.Ka_0(2)*Mmax_current*C_t_yaw;      % N*m*s/rad
    obj.PIDs(3).Kd = obj.Ka_0(3)*Mmax_current/C_t_yaw;      % N*m/s/rad
end

function M = set_Inertial_Matrix(obj)
    M = zeros(3);
    for v = 1:length(obj.bodies)
        H = H3_transf(obj.bodies(v).platf_eta(1:2));
```

```
            J =  R2d3(-obj.bodies(v).platf_eta(3));
            M = M + H'*J'*obj.bodies(v).M*J*H;
        end
        obj.M = M;
    end

    function cg = set_CG(obj)
        if ~isempty(obj.M)
            if size(obj.M) == [3,3]
                m11 = obj.M(1:2,1:2);
                m12 = obj.M(1:2,3);
                r_ = inv(m11)*m12; % in 6d this would be S(r) = m11^(-1) * m12
                cg = [r_(2);-r_(1)];
                obj.r_CoM = cg;
            end
        end
    end

    function set_configuration_control_parameters(obj)
        obj.set_Inertial_Matrix;
        obj.set_CG;

        M_cg = H3_transf(-obj.r_CoM)'*obj.M*H3_transf(-obj.r_CoM);
        obj.m_av = sum(eig(M_cg(1:2,1:2)))/2;
        obj.I_cg = M_cg(3,3);
    end

    function [Dxy,Dyaw]  = estimate_dampening_linear(obj)
        % Using a linear dampening model from modules to estimate
        % dampening on platform
        [~,~,D,~] = Delfia_Model([0 0 0 0 0 0]);
        Dp = zeros(3);
        for v = 1:length(obj.bodies)
            H = H3_transf(obj.bodies(v).platf_eta(1:2));
            J =  R2d3(-obj.bodies(v).platf_eta(3));
            Dp = Dp + H'*J'*D*J*H;
        end
        Dp_cg = H3_transf(-obj.r_CoM)'*Dp*H3_transf(-obj.r_CoM);

        % Coupling of dampening wrt the centre of gravity can
        % have off diagonal terms, including coupling between rotation
        % and translation. The latter is neglected for estimating order
        % of magnitude of dampening
        Dxy = sum(eig(Dp_cg(1:2,1:2)))/2;
        Dyaw = Dp_cg(3,3);
    end

    %% Display

    function plotPlatform(obj,UI_axis)
        % Display a representation of the assembly in axes
        if obj.plotvessels == true
            if ~isempty(obj.bodies)
                for body = obj.bodies
                    body.plotVessel(UI_axis);
                end
                end
            end
            if obj.plotref==true
                obj.plotReferenceAxis(UI_axis)
            end
        end

        function plotReferenceAxis(obj,UI_axis)
            % Display reference point in axes
            if ~isempty(obj.x_r)
                L_axis = max(obj.w,obj.l)/4; % to scale axis to vessel
                R=R2d(obj.x_r(3));
                rx = R*[0,L_axis;0,0]+ obj.x_r(1:2)';
                ry = R*[0,0;0,L_axis]+ obj.x_r(1:2)';
                plot(UI_axis,rx(1,:),rx(2,:),'color',[0.7 0 0]);
                plot(UI_axis,ry(1,:),ry(2,:),'color',[0 0.7 0]);
            end
        end

        function plotPlatform_blueprint(obj,UI_axis)
            % Display opaque blueprint of assembly at reference pose
            if ~isempty(obj.bodies)
                for body = obj.bodies
                    eta = R2d3(obj.x(3))*body.platf_eta + obj.x(1:3);
                    tempDelfia = Delfia(eta(1),eta(2),eta(3));
                    tempDelfia.plotThr = false;
                    tempDelfia.plotF = false;
                    tempDelfia.plotname = false;
                    tempDelfia.plotaxis = true;
                    tempDelfia.color = 1-(1-obj.color)*0.8; % slightly opaque
                    tempDelfia.plotVessel(UI_axis)
                end
            end
            if obj.plotCG==true
                if ~isempty(obj.r_CoM)
                    r = R2d(obj.x(3))*obj.r_CoM + obj.x(1:2);
                    plot(UI_axis,r(1),r(2),'*k');
                end
            end
            % Plot reference axis
            if obj.plotref==true
                obj.plotReferenceAxis(UI_axis)
            end
        end
    end
end
```

C.3. Matlab Class: Fleet Manager

```
classdef FleetManager <handle
%FleetManager Manages and tracks a fleet of (Delfia) Vessels and controllers
    %   This class can be assigned a set of Vessel and
    %   MultiVesselPlatformController (MVPC) objects. Various tasks of the fleet
    %   can be phasewise scripted in this object.
    %   Phases can be ran through in obj.objectiveTimedFnc, which can
    %   change phase automated or by human control
    %
    %   This architecture is designed to work via middleware: Robotic
    %   Operating System (ROS)
    %
    %   Fleet control for a fleet of Delfia's implemented here consists of
    %   1) Initialization
    %       - Register 3 Delfias to 3 MVPC objects
    %       - Set platform reference of MVPC's for initial positioning
    %   2) Assembly
    %       - Set platform reference of MVPC's for assembly
    %       - When connected, re-assign control over modules between MVPC's
    %         For assembly this means transfering control of Delfias from
    %         multiple MVPC's to a single, such that a single controller
    %         manages multiple vessels.
    %       - Controllers (MVPC's) compute parameters to enable control for
    %         new configurations.
    %   3) Assembled Motion
    %       - Assemblies perform motion to do a particular job, such as:
    %         - Forward motion (1m step)
    %         - Rotation (pi/2 rad step)
    %         - Sideways motion (0.5m step)

    % (c) Bart Boogmans 2020/2021 bartboogmans@hotmail.com,
    % You may use, distribute and modify this code under given that clear
    % credit is given to the author's work.
    %
    % Developed in cooperation with the Researchlab Autonomous Shipping
    % Delft & Department Maritime and Transport Technology of faculty 3mE, TU Delft.
    % https://rasdelft.nl/nl/
    %
    % Changelog
    % Last update 29/08/2021: commenting : Bart Boogmans

    properties
        node
        fleet
        controllers
        objectiveTimer
        objectiveTic
        phase
        phaseTic
        singleDelfiaInertia
        singleDelfiaMass
        singleDelfiaMaxForce
        singleDelfiaMaxTorque
    end

    methods
```

```
% Constructor
function obj = FleetManager()
%FleetManager() Construct an instance of this class
    obj.phase = -1;
    obj.phaseTic = tic;
    settings = delfiaSettings();
    obj.node = ros.Node('objective_planning_node',settings.hostname,'NodeHost',setti


    % Create vessel objects
    obj.fleet = Delfia.empty;
    for i = 1:settings.n_vessels
        obj.fleet(i) = Delfia(0,0,0,settings.vesselnames{i});
    end

    % Create controller objects
    obj.controllers = MultiVesselPlatformController.empty;
    for i = 1:settings.n_vessels
        obj.controllers(i) = MultiVesselPlatformController(0, 0, 0, settings.control
    end


    % Configure ROS settings of vessel and controller objects
    for i = 1:settings.n_vessels
        obj.controllers(i).node = ros.Node(obj.controllers(i).name,settings.hostname
    end
    for i = 1:settings.n_vessels
        obj.controllers(i).refPub = ros.Publisher(obj.controllers(i).node,settings.t
        obj.controllers(i).refSub = ros.Subscriber(obj.controllers(i).node,settings.
    end

    % Initiate timer object that supports control phase transition
    obj.objectiveTic = tic;
    obj.objectiveTimer = timer(...
        'ExecutionMode', 'fixedRate', ...
        'Period', (1), ...
        'BusyMode', 'drop',...
        'TimerFcn', {@obj.objectiveTimedFnc},...
        'Name', 'objectiveTimer' );
    start(obj.objectiveTimer);
end

% Destructor
function delete(obj)

    if ~isempty(obj.objectiveTimer)
        delete(obj.objectiveTimer)
    end

    obj.stopvessels();

    if ~isempty(obj.node)
        delete(obj.node)
    end
```

```
        if ~isempty(obj.fleet)
            delete(obj.fleet)
        end

        if ~isempty(obj.controllers)
            delete(obj.controllers)
        end
    end

    % Phasewise fleet behavior
    function objectiveTimedFnc(obj,~,~,~,~,~)
        % This function is called repetitively by this class' timer
        % object. It runs commands from the current phase. Phase
        % transition can be done manually, or by implementing checks
        % to advance to another.

        switch obj.phase

            case 0
                settings  = delfiaSettings();
                % Initiation
                % Set up initial configuration: All vessels are operating alone
                for i = 1:settings.n_vessels
                    obj.controllers(i).attachBody(obj.fleet(i));
                    obj.controllers(i).initiateControlParameterBroadcast();
                end
                % Wait for user input.
                obj.phaseTic = tic;
                obj.phase = -1;

            case 1
                % Set individual vessel starting position to align for
                % assembly
                obj.controllers(1).x_r = [0,-0.75,0,0,0,0];
                obj.controllers(1).ros_pub_ref();
                obj.controllers(2).x_r = [0,0,0,0,0,0];
                obj.controllers(2).ros_pub_ref();
                obj.controllers(3).x_r = [0,0.75,0,0,0,0];
                obj.controllers(3).ros_pub_ref();

                if(toc(obj.phaseTic)>=100)
                    obj.phase = obj.phase+1;
                    obj.phaseTic = tic;
                    disp(join(['Going to phase: ',string(obj.phase)]))
                end

            case 2
                % First phase of assembly. Vessels approach one-another
                % to docking position.
                obj.controllers(1).x_r = [0,-0.1,0,0,0,0];
                obj.controllers(1).ros_pub_ref();
                obj.controllers(3).x_r = [0,0.1,0,0,0,0];
                obj.controllers(3).ros_pub_ref();

            case 3
                % Optional: Temporarily increase controller gains to
            % increase normal forces, to aid proper assembly.
            for n = 1:3
                obj.controllers(n).PIDs(1).Kp = obj.controllers(n).PIDs(1).Kp *2;
                obj.controllers(n).PIDs(2).Kp = obj.controllers(n).PIDs(2).Kp *2;
                obj.controllers(n).PIDs(3).Kp = obj.controllers(n).PIDs(3).Kp *2;
            end

        case 4
            % If succesful assembly is registered or manually
            % observed: Register succesful connection:
            % Connect Delfia 1 & 3 to Delfia 2 (& thus to
            % controller 2)

            obj.controllers(1).detatchBody(1);
            obj.controllers(3).detatchBody(1);

            obj.fleet(1).platf_eta=[0;-obj.fleet(1).w;0];
            obj.fleet(2).platf_eta=[0;0;0];
            obj.fleet(3).platf_eta=[0;obj.fleet(3).w;0];

            obj.controllers(2).attachBody(obj.fleet(1));
            obj.controllers(2).attachBody(obj.fleet(3));

            % This automatically triggers adjusting control
            % protocol
            obj.phase = 5;
            obj.phaseTic = tic;
        case 5
            % Pause after assembly
            obj.controllers(2).x_r = [0,0,0,0,0,0];
            obj.controllers(2).ros_pub_ref();
            if(toc(obj.phaseTic)>=100)
                obj.phase = obj.phase+1;
                obj.phaseTic = tic;
                disp(join(['Going to phase: ',string(obj.phase)]))
            end
        case 6
            % 1m forward motion
            obj.controllers(2).x_r = [1,0,0,0,0,0];
            obj.controllers(2).ros_pub_ref();
            if(toc(obj.phaseTic)>=100)
                obj.phase = obj.phase+1;
                obj.phaseTic = tic;
                disp(join(['Going to phase: ',string(obj.phase)]))
            end
        case 7
            % 90deg rotation
            obj.controllers(2).x_r = [1,0,pi/2,0,0,0];
            obj.controllers(2).ros_pub_ref();
            if(toc(obj.phaseTic)>=100)
                obj.phase = obj.phase+1;
                obj.phaseTic = tic;
                disp(join(['Going to phase: ',string(obj.phase)]))
            end
        case 8
            % 0.5m sideways motion
```

```
                    obj.controllers(2).x_r = [1.5,0,pi/2,0,0,0];
                    obj.controllers(2).ros_pub_ref();
                    if(toc(obj.phaseTic)>=100)
                        obj.phase = obj.phase+1;
                        obj.phaseTic = tic;
                        disp(join(['Going to phase: ',string(obj.phase)]))
                    end
                case 9
                    % Return to initial position of motion test
                        obj.phase = 5;
                        disp(join(['Going to phase: ',string(obj.phase)]))
            end
        end

        % Register reference change of control objects
        function ros_sub_ref(~,~,message,controller)
            controller.ros_sub_ref(message);
        end

        % Stop object functionality
        function stop(obj)
            try
                stop(obj.objectiveTimer);
            catch
                disp('Warning: could not stop ObjectivePlanner.objectiveTimer');
            end
            try
                delete(obj.objectiveTimer);
            catch
                disp('Warning: could not delete ObjectivePlanner.objectiveTimer');
            end
            try
                delete(obj.node);
            catch
                disp('Warning: could not delete ObjectivePlanner.node');
            end
        end

        % Halt actuation of entire fleet
        function stopvessels(obj)
            for vessel = obj.fleet
                try
                    vessel.thrAngle = [0;0];
                    vessel.thrSpd = [0;0];
                    vessel.thrForce = [0;0];
                    vessel.ros_pub_actuation();
                catch

                end
            end
        end
    end
end
```

C.4. Matlab superclass: Vessel

```
classdef Delfia < Vessel & handle
%Delfia creates a vessel object representing RAS' Delfia-1* model.
    %   The Delfia object encompasses major variables, parameters, functions,
    %   and components that help various tasks related to control automation.
    %
    %   Purpose of this class is including, but not limited to:
    %   - Storing vessel specific data in a structured fashion
    %   - Support connectivity over ROS networks
    %   - Provide Delfia-1* model specific actuator control functions
    %   - Provide display functions
    %
    %   d1 = Delfia(x,y,yaw,'delfiaName')

    % (c) Bart Boogmans 2020/2021 bartboogmans@hotmail.com,
    % You may use, distribute and modify this code under given that clear
    % credit is given to the author's work.
    %
    % Developed in cooperation with the Researchlab Autonomous Shipping
    % Delft & Department Maritime and Transport Technology of faculty 3mE, TU Delft.
    % https://rasdelft.nl/nl/
    %
    % Changelog
    % Last update 29/08/2021: commenting : Bart Boogmans

    properties
        % Actuator states
        thrAngle                    % [radians]     (1x2)   Angles of thrusters
        thrForce                    % [N]           (1x2)   Estimated propulsion force of a
        thrSpd                      % [rounds/s]    (1x2)   Reference speed of thrusters

        % Communication
        actuation_publisher         % [ROS publisher]       Object for sending data to ROS t
        actuation_subscriber        % [ROS subscriber]      Object for subscribing to ROS to
        location_publisher
        location_subscriber

        % Multi-vessel platforming related variables
        platf_eta                   % [m,m,radian]  (1x3)   Pose of this vessel in platform

        % Parameters
        r_thr                       % [m]           (2x2)   Location of thrusters in local f
        RPSmin                      % [rounds/s]    (1x1)   Minimum speed of azimuth propell
        RPSmax                      % [rounds/s]    (1x1)   Maximum speed of azimuth propell
        thrFmax                     % [N]           (1x1)   Maximum force of single azimuth
        I_z                         % [kg*m^2]      (1x1)   Estimated moment of inertia arou

        % Display
        plotThr                     % [Bool]        (1x1)   Boolean true if the thruster is
        plotF                       % [Bool]        (1x1)   Boolean true if thruster forces
        forceColor                  % [rgb colormap](2x3)   Color of the force vectors from
        refColor                    % [rgb colormap](1x3)   Color of the reference
        thrOutline                  % [double]      (2xi)   Array with data on the thruster
        forceScale                  % [double]      (1x1)   Scaling factor for force arrow d

        % Other
    nr                            % [integer]     (1x1)   Optional identifier to distinguish it
end

methods
    % Constructor
    function obj = Delfia(x_,y_,yaw_,name_,~)
        obj = obj@Vessel();
        if nargin > 2
            obj.x(1:3) = [x_;y_;yaw_];
        end
        if nargin == 4
            obj.name = name_;
        end

        % Set parameters specific to Delfia-1* models
        obj.RPSmin = 8;
        obj.RPSmax = 100;
        obj.w = 0.20;
        obj.l = 0.38;
        obj.m = 4;
        obj.r_thr = [-0.155,0;0.155,0];
        [obj.M,~,~,~] = Delfia_Model([0;0;0;0;0;0]);
        obj.thrFmax = 0.2160;          % [N] per thruster at ~120rps. Error is expected to be

        % Set initial state for control related variables
        obj.thrAngle = [0,0];
        obj.thrForce = [0,0];
        obj.thrSpd = [0,0];
        obj.platf_eta = [0;0;0];

        % Initiate display settings
        cf = 0.015; % decorative chamfer
        obj.outline = [   obj.l/2      ,obj.l/2-cf   ,-obj.l/2+cf    ,-obj.l/2        ,-obj.l/
                                obj.w/2-cf   ,obj.w/2      ,obj.w/2         ,obj.w/2-cf     ,-
        obj.thrOutline =   [   0.006,      -0.013,     -0.013,      -0.037,     -0.037,     -0
                                    -0.0075,    -0.0075,    -0.023,     -0.020,     0.020,

        obj.plotThr = true;
        obj.plotF = true;
        obj.forceColor = [0,1,0 ; 1,0,0];
        obj.forceScale = 2.0;
    end

    % Destructor
    function delete(obj)
        if ~isempty(obj.actuation_publisher)
            disp(join(['Deleting ',obj.name,' .actuation_publisher ',obj.actuation_publisher.
            delete(obj.actuation_publisher)
        end
        if ~isempty(obj.actuation_subscriber)
            disp(join(['Deleting ',obj.name,' .actuation_subscriber ',obj.actuation_subscribe
            delete(obj.actuation_subscriber)
        end
        if ~isempty(obj.location_publisher)
            disp(join(['Deleting ',obj.name,' .actuation_publisher ',obj.location_publisher.T
            delete(obj.location_publisher)
        end
    if ~isempty(obj.location_subscriber)
        disp(join(['Deleting ',obj.name,' .actuation_publisher ',obj.location_subscr
        delete(obj.location_subscriber)
    end
end

%% Interaction with ROS
function ros_sub_pose(obj,message)
    % Processes an update on ROS pose topic
    obj.x(1) = message.Pose.Position.X; % Horizontal, in line with the tank (positiv
    obj.x(2) = message.Pose.Position.Y; % Horizontal, perpendicular to length of tan
    eul = euler(quaternion([message.Pose.Orientation.W,message.Pose.Orientation.X,me
    obj.x(3) = eul(1); % Range [pi to +pi]
end

function ros_sub_actuation(obj,message)
    % Processes an update on ROS actuation topic
    obj.thrSpd = [message.Data(1),message.Data(2)];
    obj.thrAngle = [message.Data(3),message.Data(4)];
    obj.calcThrF;
end

function ros_pub_pose(obj,time)
    % Publish object pose on ROS
    msg = rosmessage(obj.location_publisher);
    msg.Pose.Position.X = obj.x(1);
    msg.Pose.Position.Y = obj.x(2);

    quat = eul2quat([obj.x(3),0,0]);
    msg.Pose.Orientation.W = quat(1);
    msg.Pose.Orientation.X = quat(2);
    msg.Pose.Orientation.Y = quat(3);
    msg.Pose.Orientation.Z = quat(4);

    msg.Header.Stamp.Sec = floor(time);
    msg.Header.Stamp.Nsec = floor((time - floor(time))*1000000);

    send(obj.location_publisher,msg)
end

function ros_pub_actuation(obj)
    % Publish object actuation on ROS
    msg = rosmessage(obj.actuation_publisher);
    msg.Data = [obj.thrSpd(1),obj.thrSpd(2),obj.thrAngle(1),obj.thrAngle(2)];
    send(obj.actuation_publisher,msg);
end

%% Display
function plotVessel(obj,ax)
    % Displays the outline of the vessel, and thrusters and
    % actuator forces if configured

    R = R2d(obj.x(3));

    % Plot the hull
    plotVessel@Vessel(obj,ax);
```

```
            % Plot the thrusters
            if obj.plotThr == true
                for i= 1:length(obj.r_thr)
                    thruster_vector = R*(R2d(obj.thrAngle(i))*obj.thrOutline + obj.r_thr(i,:)') +
                    plot(ax,thruster_vector(1,:),thruster_vector(2,:),'color',obj.color);
                end
            end

            % plot the force vectors assigned to be applied by the thrusters
            if obj.plotF == true
                for i= 1:length(obj.r_thr)
                    thr_pos_n = R*(obj.r_thr(i,:)') + obj.x(1:2);
                    f_n = R*(R2d(obj.thrAngle(i))*[obj.thrForce(i);0]);
                    quiver(thr_pos_n(1),thr_pos_n(2),f_n(1),f_n(2),obj.forceScale,'LineWidth',1.5
                end
            end
        end

        %% Force and speed conversion
        function bound_motor_speeds(obj)
            for i = 1:length(obj.r_thr)
                if (obj.thrSpd(i) <= obj.RPSmin) && (obj.thrSpd(i) >= -obj.RPSmin)
                    obj.thrSpd(i) = 0;
                elseif obj.thrSpd(i) >= obj.RPSmax
                    obj.thrSpd(i) = obj.RPSmax;
                elseif obj.thrSpd(i) <= -obj.RPSmax
                    obj.thrSpd(i) = -obj.RPSmax;
                end
            end
        end

        function calcThrSpd(obj)
            % Converts force to propeller speed and assigns it to object. This is based on coarse
            for i = 1:length(obj.thrForce)
                obj.thrSpd(i) = obj.thrForce(i)/0.0018;
            end
        end

        function calcThrF(obj)
            % convert RPS to force and assigns it to object. This is based on coarse measurements
            for i = 1:length(obj.thrForce)
                obj.thrForce(i) = obj.thrSpd(i)*0.0018;
            end
        end
    end
end
```

C.5. Matlab Class: discrete PID controller

```
classdef dPID
%A simple discrete Proportional Integral and Derivative controller
    %   dPID1 = dPID(10, 0.01, 2) Creates a discrete PID controller with
    %   Kp, Ki, Kd = 10, 0.01 and 2.
    %
    %   y = dPID1.run(e,dt) calculates output y of dPID1 with given feed
    %   error e after a timestep of dt. This callback needs to be called
    %   repetitively by the system that uses this class

    % (c) Bart Boogmans 2020/2021 bartboogmans@hotmail.com,
    % You may use, distribute and modify this code under given that clear
    % credit is given to the author's work.
    %
    % Developed in cooperation with the Researchlab Autonomous Shipping
    % Delft & Department Maritime and Transport Technology of faculty 3mE, TU Delft.
    % https://rasdelft.nl/nl/
    %
    % Changelog
    % Last update 29/08/2021: commenting : Bart Boogmans
    properties
        Kp
        Ki
        Kd
        integrator
        e_last
        interpolationStyle
        limits_set
        limit_upper
        limit_lower
        int_buildup_limit
    end

    methods
        % Constructor
        function obj = dPID(kp, ki, kd)
            obj.limits_set = false;
            obj.Kp =  kp;
            obj.Ki =  ki;
            obj.Kd =  kd;
            obj.integrator = 0;
            obj.e_last = 0;
            obj.interpolationStyle = 'rectangle';
        end

        % Set PID output limits
        function obj = setlimits(obj,lower,upper)
            obj.limit_upper = upper;
            obj.limit_lower = lower;
            obj.limits_set = true;
        end

        % Runs an iteration of the PID controller
        function y = run(obj,e_new,dt)
            if size(e_new,1) ~= 1 || size(e_new,2) ~= 1 || size(dt,1) ~= 1 || size(dt,2) ~=
                disp('input to dPID.run is incorrect size')
                else
                   if dt>0
                        if strcmp(obj.interpolationStyle,'rectangle')
                            obj.integrator = obj.integrator + dt*e_new;
                        elseif strcmp(obj.interpolationStyle,'trapezoidal')
                            obj.integrator = obj.integrator + dt*(e_new+obj.e_last)/2;
                        end

                        if  ~isempty(obj.int_buildup_limit)
                            if obj.integrator > obj.int_buildup_limit
                                obj.integrator = obj.int_buildup_limit;
                            elseif obj.integrator < -obj.int_buildup_limit
                                obj.integrator = -obj.int_buildup_limit;
                            end
                        end

                        y_proportional =    obj.Kp*         e_new;
                        y_derivative =      obj.Kd*         (e_new-obj.e_last)/dt;
                        y_integrator =      obj.Ki*         obj.integrator;

                        obj.e_last = e_new;
                        y = y_proportional + y_integrator + y_derivative;
                        if obj.limits_set
                            if y>obj.limit_upper
                                y = obj.limit_upper;
                            end
                            if y<obj.limit_lower
                                y = obj.limit_lower;
                            end
                        end
                   else
                        disp('Warning: dt input of dPID<=0')
                        y =0;
                    end
                end
            end
        end
    end
```