Ocean Engineering 235 (2021) 109433

Contents lists available at ScienceDirect

Ocean Engineering

journal homepage: www.elsevier.com/locate/oceaneng

Dynamic Positioning using Deep Reinforcement Learning

Simen Sem Øvereng [a], Dong Trong Nguyen [a,b,*], Geir Hamre [b]

[a] Department of Marine Technology, NTNU, 7491 Trondheim, Norway
[b] DNV, Veritasveien 1, 1363 Høvik, Norway

ARTICLE INFO

Keywords:
Dynamic Positioning
Deep Reinforcement Learning
Proximal policy optimization
Reward shaping

ABSTRACT

This paper demonstrates the implementation and performance testing of a Deep Reinforcement Learning based control scheme used for Dynamic Positioning of a marine surface vessel. The control scheme encapsulated motion control and control allocation by using a neural network, which was trained on a digital twin without having any prior knowledge of the system dynamics, using the Proximal Policy Optimization learning algorithm. By using a multivariate Gaussian reward function for rewarding small errors between the vessel and the various setpoints, while encouraging small actuator outputs, the proposed Deep Reinforcement Learning based control scheme showed good positioning performance while being energy efficient. Both simulations and model scale sea trials were carried out to demonstrate performance compared to traditional methods, and to evaluate the ability of neural networks trained in simulation to perform on real life systems.

1. Introduction

Dynamic Positioning (DP) of marine vessels is concerned with maintaining a vessel's position and heading while using a computer program to control the vessel's actuators. Due to the nonlinear dynamics of marine vessels and the stochastic behavior of the environment, the DP task can become quite complex. Traditional methods for solving DP consist of a state estimation component, responsible for signal processing and estimation of the vessel's states given the various sensor inputs, a guidance system responsible for calculating set points to the control system, and a control system which typically consists of a motion control law and a thrust allocation (TA) method. A simplified overview of the DP system is shown in Fig. 1, showing the connectivity between the above-mentioned DP system components, including the forces acting on the vessel coming from the environmental loads and the actuators.

A considerable amount of research has gone into the state estimation algorithms, as the DP system should only counteract the low-frequency wave motions which cause the vessel to drift over time, filtering out the wave-frequency motions which cause the vessel to oscillate. Early work included the use of notch filters and low-pass filtering, while today's methods tend to linearize the dynamics and use linear quadratic estimation, based on the work by Kalman in the 1960's with applications like in Balchen et al. (1980) who combined the Kalman filter (KF) with optimal control. The extended Kalman filter (EKF) has become a standard, in which the dynamics are linearized around working points using the Taylor expansion, but it lacks global stability proofs since the dynamics are linearized within certain regions of heading angle. One option to the EKF is the nonlinear passive observer, derived by Fossen and Strand (1999), providing global stability proofs while using less tuning parameters, with demonstrated performance on a supply vessel. Since an EKF was already implemented on the vessel used for demonstrations in this paper, the focus was on the control aspect of DP, namely the motion control and the thrust allocation.

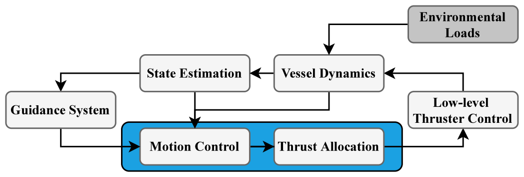


Fig. 1. Flowchart of a traditional DP systems. The blue area marks the focus of this paper. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Traditional methods used in a control system divide the control problem itself into two parts: the motion controller, and the thrust allocation. The motion controller calculates the generalized control forces $\boldsymbol{\tau}_d$ to put on the vessel in order to move it towards a setpoint, while the TA translates these generalized forces into thruster commands, resulting in the actual forces $\boldsymbol{\tau}_b$ on the vessel. A widely used motion

* Corresponding author.
E-mail address: dong.t.nguyen@ntnu.no (D.T. Nguyen).

https://doi.org/10.1016/j.oceaneng.2021.109433
Received 4 March 2021; Received in revised form 28 May 2021; Accepted 28 June 2021
Available online 6 July 2021

control law for controlling the horizontal motions of a vessel is to use three decoupled PID feedback control laws in combination with a feedforward law that combines the modeled system dynamics with information such as measured wind speeds or changes in desired states in order to calculate desired forces up front. However, as the vessel dynamics are highly nonlinear, inaccuracies arise due to modeling errors in addition to time varying environmental and/or operating conditions. More advanced control methods aiming to combat these issues include methods that use gain scheduling, exemplified by Tannuri et al. (2006) using model-reference adaptive control. For time varying environmental loads, methods using PID with fuzzy logic which adapts controller's coefficients has been demonstrated, such as the fuzzy PID controller by Xu et al. (2019). Other methods aimed at including the nonlinear dynamics directly in the controller design, such as backstepping control, a recursive control design methodology introduced by Kokotovic (1992), and widely exemplified by in the works of Fossen and Grøvlen (1998) and Skjetne and Fossen (2004), and Du et al. (2018). Katebi et al. (2001) used the H_∞ controller design method based on the wind and wave disturbances, providing an option for multivariate systems with coupling terms. Also sliding mode control has been investigated, a nonlinear controller design method that consists of a "sliding surface" which is meant to guarantee stable dynamics when the trajectory lies on it, demonstrated for marine vessels in DP operations by Tannuri et al. (2010). Nguyen et al. (2007) used hybrid control as a means of expanding the range of varying environmental conditions, using a switching logic for selecting between a predetermined set of observers and controllers according to the estimated sea state spectre's peak frequency. The work was later extended to apply for various operating speeds as well (Nguyen et al., 2008).

The thrust allocation algorithm's goal is to output actuator forces that is as close to the ones calculated from the motion controller, i.e. minimizing $\|\boldsymbol{\tau}_b - \boldsymbol{\tau}_d\|_2$. For most marine surface vessels, the number of actuators are larger than the degrees of freedom to be controlled, and hence there might exist an infinite amount of thruster configurations that yields the desired forces. Therefore, "optimal" solutions must be chosen with respect to some objective criteria. As a consequence, TA methods are typically based on optimization (Johansen and Fossen, 2013) which takes aim at minimizing a cost function subject to the physical constraints of the thrusters and the relationship between thruster forces and total force generation. This allows for flexible constraint handling while optimizing a designable objective function. For minimizing fuel consumption, a Quadratic Programming (QP) approach is common in which the cost function includes a quadratic cost on thruster usage. Optimization based methods could suffer from complex actuator setups leading to nonlinear optimization constraints and nonconvex objective functions which can be computationally demanding to solve online. Avoiding some of the nonlinearities that creates computational issues, the application of an extended thrust formulation for calculating a pseudoinverse of the linear actuator model has become fairly standard in the industry (Sørdalen, 1997; Jenssen and Realfsen, 2006), though requiring some iterative logic to ensure that the commanded thruster signals are feasible. A variety of workarounds of the nonlinearities has been proposed, see e.g. Johansen et al. (2008) for allocation with a propeller and rudder interaction, where the proposed method divide the nonconvex problem into several convex QP problems, with a switching logic to decide on the best solution by comparing final costs of each separate sub-solution. Other approaches include multiparametric Quadratic Programs (mpQP) (Gupta et al., 2011) which pre-computes optimal parametric functions offline, and as demonstrated by Johansen et al. (2005), offers real-time applications by searching for the optimal solutions within the pre-computed functions. The method was further expanded by Leavitt (2008), combining several mpQP formulations with different properties, including a blending and a switching logic for deciding the best solution from the sub-problems. Newer methods improve the accuracy of TA methods as shown by Arditti et al. (2018) which used sequential QP with slack variables, including actual thrust constraints and hydrodynamic thruster interactions in high detail to increase TA accuracy while reducing energy consumption. Through the rapid development of computer hardware, Model Predictive Control (MPC) has become a viable option for the allocation of actuator commands (Vermillion et al., 2007; Naderi et al., 2019). It has also been shown to be applicable for encapsulating both the motion controller and the TA into one entity, as demonstrated by Veksler et al. (2016). They formulated the controller as an optimization over a given prediction horizon, calculating optimal thruster commands through minimizing a cost function which included thruster usage and state deviations, while formulating the vessel and thruster dynamics, in addition to their limitations, as constraints to the optimization procedure.

Reinforcement Learning (RL) algorithms have been used with promising results in a large variety of decision-making tasks, including control problems. A comprehensive overview of RL can be found in Sutton and Barto (2017). As opposed to the traditional methods which are based on instructive design based on some sort of modeling of the vehicle dynamics, RL based controllers is developed in a trial-and-error fashion, finding (or "learning") a control policy through choosing actions while receiving feedback from a designed reward function signal. The behavior is corrected based on a reward mechanism. An incentive for exploring methods within model-free RL has been that they have shown to provide frameworks for learning control policies by using methods that has no a priori knowledge of the system dynamics. This has allowed for development of controllers for nonlinear systems without any attention to dynamics modeling. These methods have especially proven performance when combining RL with deep Artificial Neural Networks (ANN), giving rise to Deep Reinforcement Learning (DRL). Widespread attention of DRL-based methods was attained with the demonstration of Deep Q-Networks (DQN) (Mnih et al., 2013), which combined Q-learning with Neural Networks to reach super-human level in Atari computer games. Advancing similar ideas, also the AlphaZero program gained super-human abilities in very complex games such as Chess and Go through RL-based self-play (Silver et al., 2018), considered as state-of-the-art within DRL today.

Recent work using DRL on dynamic systems includes a variety of path-following and station keeping problems for systems which often contain hard-to-model nonlinearities, ranging from aerial vehicles (Koch et al., 2018; Bohn et al., 2019) to underwater vehicles (Kjærnli, 2018; Knudsen, 2019). DRL-based methods for control of marine surface vessels has, similar to traditional methods, divided the control problem into motion control and TA, in addition to motion planning (Chen et al., 2019; Guo et al., 2020), and path following through speed and/or heading control (Martinsen and Lekkas, 2018; Cui et al., 2019). Martinsen et al. (2020) demonstrated an approach that included data-driven system identification used in feedforward control with a RL-based feedback control law to create a motion controller capable of solving both low-speed DP tasks and high-speed path following. TA has also been subject to testing of artificial intelligence based methods. Luman et al. (2015) solved TA by using genetic algorithms, and Wu et al. (2016) used bee colony based optimization techniques, both for energy optimal allocation. Skulstad et al. (2018) showed that a neural network was able to be trained on translating desired forces from a PID motion controller into thruster commands rather efficiently.

The work in this paper was motivated by some of the key results coming from Martinsen et al. (2020), where they combined their motion controller with an existing implementation of a TA algorithm from Det Norske Veritas (DNV) on the ReVolt platform DNV (2015). One of their discussions was based on the observed mismatch between the forces commanded by their motion controller and the commanded forces from the TA, coming from that the desired forces from their motion controller was assumed to be instantaneously achievable, while in reality the TA involves delays in terms of the actual time the podded thrusters use to rotate or to change the propellers' speed. The TA procedure itself was left out of the scope of their work, but

Table 1
Notations.

Symbol	Explanation
$\boldsymbol{\tau}_d$	Generalized forces calculated by the motion controller, subscript d referring to "desired".
$\boldsymbol{\tau}_b$	Control forces and moments.
$\boldsymbol{\eta}^k = [x,\ y,\ \psi]^\top$	Low-frequency position and heading, denoted as *pose*, represented in a certain reference frame k.
$\tilde{\boldsymbol{\eta}}$	Deviation between current pose and a desired pose.
G_t	Accumulated future reward for an RL agent from time t towards episode end.
$G(\tau)$	Accumulated rewards for an RL agent following trajectory τ.
γ	Discount rate $0 \leq \gamma \leq 1$.
$J(\pi)$	Objective function for the RL agent to maximize during learning, following policy π.
π_{θ_a}	Policy (representing a control law) of an RL agent, parameterized by the weights θ_a in a neural network.
$\hat{V}_{\theta_c}$	Value function of an RL agent, parameterized by the weights θ_c in a neural network.
$\bar{r}(\theta_a)$	Probability ratio between taking an action with updated versus old neural network parameters.
$\hat{A}_t$	The advantage of taking a certain action at time t compared to the average return of all available actions.
$n_{k,t}$	Thrust output from thruster k at time t.
$\alpha_{k,t}$	Thruster angle from thruster k at time t.
Σ	Non-negative diagonal, square matrix used for reward shaping based on pose.

was recommended as future work. This motivated the search for a control scheme that uses the advantages of a precomputed control law using DRL which can learn the nonlinear dynamics of the vessel *and* the thrusters, hence possibly circumventing modeling inaccuracies and computational complexity at the same time, while directly optimizing the thruster commands in order to eliminate body frame errors. The main objective of this paper is therefore to solve the DP control problem for a marine surface vessel utilizing the well-proven Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) for training a DRL-based control scheme which encapsulated both the motion controller and the TA into one entity. Access to the TA method from DNV was available for testing in this paper, so the performance is evaluated by comparing with previously implemented control systems on board the ReVolt platform (using a motion control law including PID and feedforward control, and DNV's TA, separately), both in simulation and sea trial demonstrations.

The main contributions of this paper is as follows:

I An end-to-end learning procedure for DRL-based methods, using neural networks to encapsulate both motion control and thrust allocation into one entity for solving DP tasks while at the same time not suffering from computational complexity.
II Introduction of a novel, multi-variable Gaussian reward function used to train the Reinforcement Learning agent in eliminating body-frame errors for marine vessels.
III Presentation of performance in both simulations and in a sea trial, demonstrating the transfer capability of the DRL model from simulation to physical systems, having only trained the model in simulations.

The organization of this paper is as follows: Section 2 gives a description of the DP task, the vessel model, and DRL, while Section 3 explains the development of the DRL framework and the procedure of training the neural networks. Section 4 demonstrates the performance from simulations and the sea trial, and discusses the main findings. Finally, Section 5 concludes the work. The notation used in this paper is shown in Table 1.

2. System description

2.1. Dynamic positioning and vessel dynamics

To analyze the vessel's motions, the geographical reference frame North-East-Down (NED) and the body-fixed reference frame was used. NED is chosen as a tangent plane *fixed* to the surface of the earth, and positions within the frame is denoted (x_n, y_n, z_n), where the x_n-axis points towards true North (N), the y_n-axis points East (E) and the z_n-axis points downwards. The body-fixed reference frame is denoted (x_b, y_b, z_b), and as defined in this paper, the positive direction was defined for the x_b-axis to point in the forward direction of the vessel, the y_b-axis towards starboard, the z_b-axis downwards, with the origin placed in the Center of Gravity (CG) of the vessel. This implies that the vessel's heading ψ was defined to be relative to true North, rotating clockwise. In addition, a hydrodynamic, earth-fixed frame denoted (x_h, y_h, z_h) was used for modeling the vessel motions subject to wave loads. When used in DP, the origin of the frame is moved to the desired coordinate (x_d, y_d) and aligned with the desired heading angle ψ_d. The vessel is assumed to oscillate with small motions about this frame in order to utilize linear theory when modeling the wave-induced motions.

The *pose* $\boldsymbol{\eta}$ was used as the three-dimensional vector of position and heading the vessel, either as $\boldsymbol{\eta}^n$ in the NED-frame or as $\boldsymbol{\eta}^b$ describing position and heading relative to the vessel's body-frame. The body-frame *errors* $\tilde{\boldsymbol{\eta}}^b$ represents the deviation between the vessel's current pose and the desired pose, and a DP system should work to eliminate these deviations, namely ensuring $\tilde{\boldsymbol{\eta}}^b \rightarrow 0$. By first calculating the errors in the NED-frame as the deviation between the current pose and the desired pose in the NED-frame, $\tilde{\boldsymbol{\eta}}^n = \boldsymbol{\eta}^n - \boldsymbol{\eta}^n_d = [\tilde{N}, \tilde{E}, \tilde{\psi}]^\top$, the body-frame errors are calculated as shown in Eqs. (1a), (1b) and (1c).

$$\begin{bmatrix}\tilde{x}\\ \tilde{y}\end{bmatrix} = \begin{bmatrix}cos(\psi) & -sin(\psi)\\ sin(\psi) & cos(\psi)\end{bmatrix}^\top \begin{bmatrix}\tilde{N}\\ \tilde{E}\end{bmatrix}. \tag{1a}$$

$$\tilde{\psi} = \psi - \psi_d. \tag{1b}$$

$$\tilde{\boldsymbol{\eta}}^b = [\tilde{x},\ \tilde{y},\ \tilde{\psi}]^\top. \tag{1c}$$

Modeling the vessel motions are usually done in one of two ways. A high-fidelity *process plant model* (PPM) is used where the physics are modeled as exact as possible to reality for accurate numerical simulations and various analysis. On the other hand, a *control plant model* (CPM) is based on a simplified mathematical model of the vessel dynamics. A CPM is usually used for stability analysis and classic controller design, while this paper based the controller design on using a PPM for training neural networks. Thus, the following briefly explains the PPM used. For further explanation of the terms, the reader is referred to Fossen (2011) and Sørensen (2018). The numeric values of the system matrices were found in Alfheim and Muggerud (2016).

The vessel motions in a PPM could be simplified into two models (Sørensen et al., 1996): low-frequency (LF) wave loads and the wave-frequency (WF) wave loads. The LF wave loads are primarily coming from second-order mean and slowly varying forces from waves, current and wind. The simulation model for the LF loads used in this paper was based on Sørensen (2018), in which the formulation of the 6-DOF equations of motion is given in Eq. (2) for a nonlinear LF model of the vessel,

$$\begin{aligned} M\dot{\boldsymbol{v}} + C_{RB}(\boldsymbol{v})\boldsymbol{v} + C_A(\boldsymbol{v}_r)\boldsymbol{v}_r + D(\boldsymbol{v}_r) + G(\boldsymbol{\eta}) \\ = \boldsymbol{\tau}_{wave_2} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_b \end{aligned} \tag{2}$$

where $\boldsymbol{\nu} \in \mathbb{R}^6$ is the body-frame velocities; $\boldsymbol{\nu}_r \in \mathbb{R}^6$ is the relative velocities between the vessel and the ocean current; $M \in \mathbb{R}^{6\times6}$ represents the system inertia and added mass matrix; $C_{RB} \in \mathbb{R}^{6\times6}$ is the rigid body Coriolis and centripetal matrix; $C_A \in \mathbb{R}^{6\times6}$ is the added mass Coriolis and centripetal matrix; $D(\boldsymbol{\nu}_r) \in \mathbb{R}^{6\times6}$ represents the linear and nonlinear damping; $G(\boldsymbol{\eta}) \in \mathbb{R}^{6\times6}$ is the generalized restoring vector coming from buoyancy and gravitation; $\boldsymbol{\tau}_{wave_2} \in \mathbb{R}^6$ is the second-order wave load; $\boldsymbol{\tau}_{wind} \in \mathbb{R}^6$ is the wind load; $\boldsymbol{\tau}_b \in \mathbb{R}^6$ is the control forces and moments put on the vessel by the actuators.

The second of the two simplified components are linked to the wave-frequency (WF) components that is primarily due to first-order wave loads. Here, the coupled equations of the WF motions in surge, sway, heave, roll, pitch and yaw are assumed to be linear, and can be formulated as shown in Eq. (3),

$$\begin{aligned} M(\omega_i)\ddot{\boldsymbol{\eta}}_{Rw} + D_p(\omega_i)\dot{\boldsymbol{\eta}}_{Rw} + G\boldsymbol{\eta}_{Rw} &= \boldsymbol{\tau}_{wave_1}, \\ \dot{\boldsymbol{\eta}}_w &= J(\overline{\boldsymbol{\eta}}_2)\dot{\boldsymbol{\eta}}_{Rw}, \end{aligned} \tag{3}$$

where $M(\omega_i) \in \mathbb{R}^{6\times6}$ represents the system inertia matrix containing the vessel's mass and moment of inertia in addition to added mass coefficients that are dependent on the wave frequency ω_i; $\boldsymbol{\eta}_{Rw} \in \mathbb{R}^6$ is the WF motion vector in the hydrodynamics frame; $D_p(\omega_i) \in \mathbb{R}^{6\times6}$ is the wave radiation damping matrix; $G \in \mathbb{R}^{6\times6}$ represents the linearized restoring coefficient matrix coming from gravity and buoyancy affecting heave, roll and pitch only; $\boldsymbol{\tau}_{wave_1} \in \mathbb{R}^6$ is the first order wave excitation vector; $\boldsymbol{\eta}_w \in \mathbb{R}^6$ is the WF motion vector in the NED-frame; $J(\overline{\boldsymbol{\eta}}_2) \in \mathbb{R}^{3\times3}$ is the rotation matrix relating the WF velocities between the NED-frame and the hydrodynamic frame as defined in Fossen (2011).

2.2. The ReVolt ship model

The ReVolt ship model was used as a demonstration platform. It is a 3-meter-long ship model, representing a 1:20 scale model of the 60 meter long ReVolt ship concept (DNV, 2015). A digital twin (or a PPM) based on a 6DOF model has been developed for the model scaled vessel by using MATLAB and Simulink, then verified through frequency domain analysis of a 3D model of the hull and through experiments in a towing tank. The PPM was used for running simulations in DNV's CyberSea simulation environment, which is used to simulate the vessel and the relevant equipment onboard, such as thrusters, power system, sensors, position reference systems, and possibly other equipment used in marine operations relevant for DP. This allows for rapid deployment and testing of the control system in various sea states. See Nguyen et al. (2013) for further details.

The real life ship model runs a Tank-720 computer with the Linux Ubuntu LTS 16.04 operating system on board, powered by two 12 V batteries. The Robot Operating System (ROS) runs as a means of communication between the sensors (Global Navigation Satellite System, accelerometer, gyroscope, and compass), software, and the various hardware, having ROS Kinetic as the current version. The propulsion consists of two fully rotatable podded two-bladed thrusters in the stern (referred to as port and starboard thrusters), and a retractable, podded two-bladed bow thruster. The main characteristics of the vessel and the thrusters are listed in Table 2, while the thruster placements are displayed in Fig. 3.

2.3. Reinforcement learning

In RL, it is usually assumed that an agent makes actions in a certain environment (e.g. a simulated environment with vessel dynamics and environmental loads), and the interactions between the agent and the environment can be described as a Markov Decision Process (MDP) (Sutton and Barto, 2017). An MDP consists of a set of states defining the state space, $s \in \mathcal{S}$; a set of actions defining the action space, $a \in \mathcal{A}$; a probability matrix relating the selection of an action a in a certain state s and ending up in a new state s', $P \in \mathcal{P}(s'|s,a)$; and a *scalar* reward function that weights the reward of taking action a in state s, $r \in \mathcal{R}(s,a)$. The RL process with such an MDP is illustrated in Fig. 2, where an agent decides action a_t given a state s_t, yielding the next state s_{t+1} and a reward signal r_t from the environment.

Table 2
ReVolt vessel main characteristics.

Characteristic	Model	Full scale
Length Over All (LOA)	3.00 m	60.0 m
Beam	0.72 m	14.5 m
Depth	0.65 m	13.0 m
Draft	0.25 m	5.0 m
Mass	0.257 t	2056 t
Diameter, Thruster 1 and 2	0.15 m	3.0 m
Diameter, Thruster 3	0.06 m	1.3 m
Max Power, Thruster 1 and 2	96.0 W	3435 kW
Max Power, Thruster 3	16.7 W	600 kW

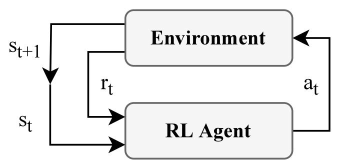


Fig. 2. A basic schematic of Reinforcement Learning.

The goal of the RL algorithm (or the *agent*) is to find an optimal policy π^* which maximizes the rewards over time. Letting the return of a certain time period following time t be denoted G_t, a common formulation is the *infinite-horizon discounted return* as shown in Eq. (4), where $0 \le \gamma \le 1$ is the discount rate, a weighting factor between rewards accumulated immediately, and reward accumulated in future time steps.

$$G_t := r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+(k+1)}, \tag{4}$$

Since the RL agent does not have a perfect model of the world, the task is to maximize the *expected* return. An objective function $J(\pi)$ representing the expected return to be maximized can be formulated as in Eq. (5). It expresses the expected return over the trajectory τ of state–action pairs, following the action-selection of the agent's policy π, interacting with the environment by observing a state, taking an action, and getting a reward accumulated as $G(\tau)$.

$$J(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi}\Big[G(\tau)\Big] \tag{5}$$

Commonly used in RL algorithms is the notion of a value function, describing the estimated value of the return by starting in state s_t and following the actions taken by the policy π into the future,

$$V(s_t) := \mathop{\mathbb{E}}_{\tau\sim\pi}[G(\tau)|s_0 = s_t]. \tag{6}$$

Various RL algorithms combine the policy and value function in different ways in order to approximate the optimal policy. By the inclusion of ANNs for approximation, RL algorithms' expressive power has increased when solving MDPs with continuous state- and action spaces. Where actor-only algorithms only approximate the policy by using an ANN, actor critic algorithms use ANNs for approximating both the policy, called the actor, *and* the value function estimator, called the critic. In this work, the actor's output, being an action when given a state s_t, is denoted $\pi_{\theta_a}(s_t)$, while the value function estimate from the critic is denoted $\hat{V}_{\theta_c}(s_t)$.

The algorithm used in this paper is called Proximal Policy Optimization (PPO) and uses the actor critic structure to learn the policy with the help of a value function. Through PPO, Schulman et al. (2017) presented an objective function for training the actor's ANN in a way that was data efficient, robust with respect to hyperparameter changes, and easy to implement. It was based on maximizing the expected return while limiting the magnitude of the updates to the actor's ANN by using

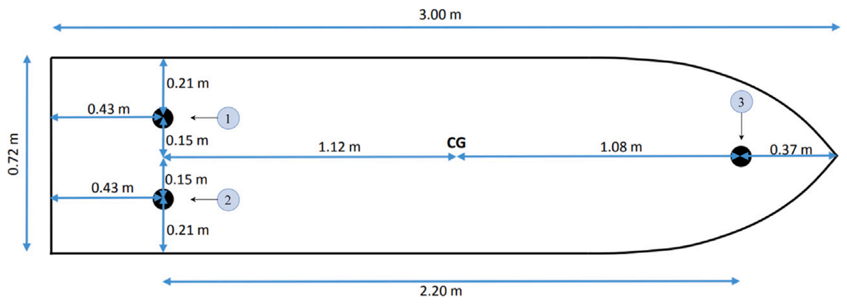


Fig. 3. Model ship thruster enumeration and placements.

the idea of trust regions. It did so by using the conservative estimate of the expected return of a state–action pair (s_t, a_t) as shown in Eq. (7).

$$\mathbb{E}_t\left[G_t\right] \approx \mathbb{E}_t\left[\bar{r}(\theta_a)_t \hat{A}_t\right]. \tag{7}$$

Here, $\bar{r}(\theta_a)_t$ is the probability ratio between taking an action in a current state with the new network parameters θ_a, and the old ones, $\theta_{a,old}$. Note that $\bar{r}(\theta_a) \geq 0\ \forall\ \theta$ and $\bar{r}(\theta_{a,old})_t = 1.0$.

$$\bar{r}(\theta_a)_t = \frac{\pi_{\theta_a}(a_t|s_t)}{\pi_{\theta_{a,old}}(a_t|s_t)}. \tag{8}$$

$\hat{A}_t$ denotes the estimate of the *advantage function*, representing how much better it was to select action a_t in state s_t compared to the average return of all actions available in state s_t. Generalized Advantage Estimation (Schulman et al., 2015) was used for estimating the advantage as shown in Eq. (9), where the critic's value estimates from Eq. (6) was used.

$$\hat{A}_t = \sum_{i=0}^{\infty}(\gamma\lambda)^i \left(r_t + \gamma\hat{V}_{\theta_c}(s_{t+1}) - \hat{V}_{\theta_c}(s_t)\right). \tag{9}$$

By limiting the magnitude of $\bar{r}(\theta_a)_t$ per parameter update, we can perform improvements on the objective function while being careful of not making too large parameter updates in θ_a-space which might reduce the performance in the policy's output space, π_{θ_a}. The proposed objective function from Schulman et al. (2017) included bounding the size of $\bar{r}(\theta_a)_t$ on both sides (called *clipping*) in order to enforce a limitation of the probability ratio so that $\bar{r}(\theta_a)_t \in [1-\epsilon, 1+\epsilon]$. By choosing the objective function to be the minimum of the conventional estimate from Eq. (7) and the estimate using the clipped ratio gave their *clipped surrogate objective* function as shown in Eq. (10), where the conservative objective function estimate is used unless the value of the ratio $\bar{r}(\theta_a)_t$ becomes too large or too small.

$$J(\theta_a)_t = \min\left(\bar{r}(\theta_a)_t\hat{A}_t,\ \text{clip}\left(\bar{r}(\theta_a)_t, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right). \tag{10}$$

The critic network performed parameter updates by attempting to *minimize* the deviations between its value function estimates versus the experienced values of being in a state by using the Mean Squared Error function. Thus, the update rule was as showed in Eq. (11) for the critic's weights, θ_c. The actor attempted to *maximize* the clipped surrogate objective, and therefore used the update rule for its weights, θ_a, as in Eq. (12).

$$\theta_{c,k+1} \leftarrow \underset{\theta_c}{\text{argmin}}\left\{\frac{1}{T}\sum_{t=1}^{T}\left(\hat{V}_{\theta_{c,k}}(s_t) - G_t\right)^2\right\}, \tag{11}$$

$$\theta_{a,k+1} \leftarrow \underset{\theta_a}{\text{argmax}}\left\{\frac{1}{T}\sum_{t=1}^{T}J(\theta_a)_t\right\}. \tag{12}$$

The training process following these update rules can be illustrated as shown in Fig. 4, where the reward from the reward function is propagated to the critic network, which in turn calculates information for the actor network update rule (as seen used in Eqs. (9) and (10)). The following section shows how the input and outputs of the networks were defined, and how the reward function was shaped in order to allow the neural networks to learn how to solve the DP task specifically.

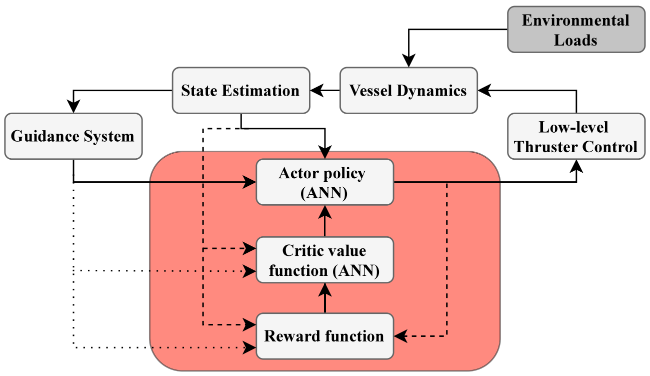


Fig. 4. Architecture of the training process of an DRL agent using an actor critic structure.

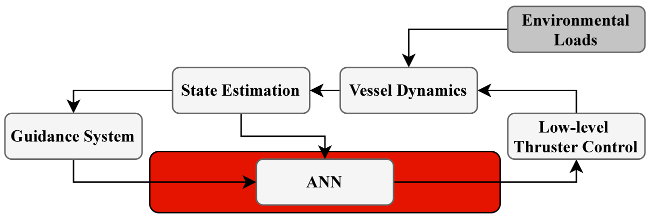


Fig. 5. Flowchart of the proposed method.

3. Controller design and implementation

The characteristic difference between traditional methods and the one proposed in this paper is shown in Fig. 5.

Fig. 5 shows how the motion controller and TA were encapsulated into one entity, represented by a trained ANN (the actor network). Compared to Fig. 1, no information of the vessel dynamics was given to the controller. The neural network directly translated the body frame errors into thruster commands, making it a feedback controller. It should be noted that when training was finished, only the actor ANN was used as the control policy; the critic ANN and the reward function was only used during the training process.

3.1. State and action vector

The state vector used as input to the actor and the critic networks was developed with the goal of DP in mind, thus adding the body-frame errors as the first components. Additionally, it was beneficial for the agent to have access to the body-frame velocities (calculated from the body-frame errors between the two most recent time steps) u, v and ψ, which were added next. In addition, the previous time step's thruster commands were added to the state vector in order to increase the agent's ability to minimize the penalty put on the size

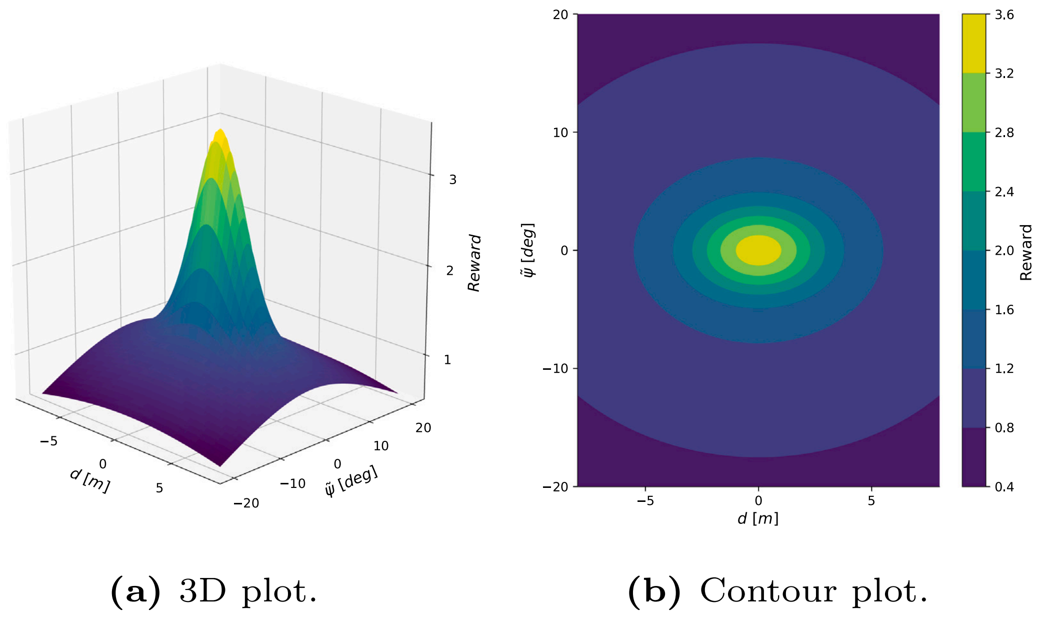


Fig. 6. Plots of $R_{gauss} + R_{AS}$ using $c_{gauss} = 2.0$, $\sigma_d = 1.0$, $\sigma_{\tilde{\psi}} = 5.0$, $c_{AS} = 0.1$ and $c_{const} = 0.5$.

of the output from the network's thrust-components. The quantities in the state vector was extracted straight from the simulator during training, and from the existing observer and reference filter in ROS during testing. The state vector is shown in Eq. (13), where $\tilde{x}_t$, $\tilde{y}_t$ and $\tilde{\psi}_t$ are the body-frame position and heading error, $\tilde{u}_t$, $\tilde{v}_t$ and $\tilde{\dot{\psi}}_t$, represents the surge, sway and yaw velocities respectively (estimated from position differences between time steps), and $\bar{n}_{b,t-1}$, $\bar{n}_{p,t-1}$ and $\bar{n}_{s,t-1}$ represents the previous time step's thruster commands for the bow, port and starboard thruster respectively, all normalized to the region $[-1,\ 1]$ to represent a fraction of maximum thrust.

$$\boldsymbol{s}_t = [\tilde{x}_t,\ \tilde{y}_t,\ \tilde{\psi}_t,\ \tilde{u}_t,\ \tilde{v}_t,\ \tilde{\dot{\psi}}_t,\ \bar{n}_{b,t-1},\ \bar{n}_{p,t-1},\ \bar{n}_{s,t-1}]^\top. \tag{13}$$

Formulating the action vector $\boldsymbol{a}_t$, it was found to increase the rate of learning if letting the policy predict the sines ($s(\cdot)$) and cosines ($c(\cdot)$) of the angles of the port (p) and starboard (s) thrusters instead of the raw angles directly. Previous experience (Alfheim and Muggerud, 2016; Øvereng, 2020) had shown that the DP ability of the vessel improved by setting the bow thruster to a fixed angle of 90°, so the bow thruster's angle was not included in the action vector. The resulting action vector was as shown in Eq. (14).

$$\boldsymbol{a}_t = [n_{b,t},\ n_{p,t},\ n_{s,t},\ s(\hat{\alpha}_{p,t}),\ c(\hat{\alpha}_{p,t}),\ s(\hat{\alpha}_{s,t}),\ c(\hat{\alpha}_{s,t})]^\top. \tag{14}$$

In Eq. (14), $n_{b,t}$, $n_{p,t}$, and $n_{s,t}$ represents the commanded thrust signal for the bow, port and starboard thruster respectively, in the region of $[-100\%,\ 100\%]$ of maximum thruster RPM. $s(\hat{\alpha}_{p/s,t})$ and $c(\hat{\alpha}_{p/s,t})$ represent the outputted sines and cosines of the angles og the port and starboard thruster, while the final command sent to the thrusters where calculated as shown in Eq. (15), where `atan2` is the four-quadrant inverse tangent function.

$$\alpha_{i,t} = \mathrm{atan2}\big(s(\hat{\alpha}_{i,t}),\ c(\hat{\alpha}_{i,t})\big). \tag{15}$$

3.2. Reward shaping

The shape of the reward function for rewarding small deviations from the setpoint in terms of Euclidean distance, $d = \sqrt{\tilde{x}^2 + \tilde{y}^2}$, and heading deviation, $\tilde{\psi}$, was inspired by the shape of the output layer of the policy network itself, using a multivariate Gaussian function. Hence, small deviations was rewarded by using the shape of the multivariate Gaussian as shown in Eqs. (16) and (17), spanning the two-dimensional $(d, \tilde{\psi})$-space, by using a diagonal, square matrix Σ.[1]

$$\Sigma = \mathrm{diag}([\sigma_d^2, \sigma_{\tilde{\psi}}^2]). \tag{16}$$

$$R_{gauss} = c_{gauss} \exp\left(-\frac{1}{2}\begin{bmatrix} d & \tilde{\psi} \end{bmatrix} \Sigma^{-1} \begin{bmatrix} d \\ \tilde{\psi} \end{bmatrix}\right). \tag{17}$$

To reduce the sparsity of the function (as only a negligible reward was given when far away from the setpoint), a function was added by using a distance measurement ϕ in the $(d, \tilde{\psi})$-space for guiding the agent's learning process, in addition to a constant as shown in Eq. (18). The resulting reward function for pose was therefore as shown in Fig. 6.

$$R_{\mathrm{AS}} = \max\big(0,\ \ (1 - c_{AS}\ \phi)\big) + c_{const}. \tag{18}$$

Since actuator penalties were added later, a constant c_{const} was also added in order to avoid cases were the agent could find it more profitable to exert no thrust than to reduce the body-frame errors.

To avoid the agent overshooting the setpoints, penalties on the velocities was added as a small quadratic penalty, using weighting coefficients c_u, c_v and $c_{\tilde{\dot{\psi}}}$ for the surge, sway and yaw velocities respectively.

$$R_{vel} = -\sqrt{c_u(\tilde{u})^2 + c_v(\tilde{v})^2 + c_{\tilde{\dot{\psi}}}(\tilde{\dot{\psi}})^2}. \tag{19}$$

In order to achieve energy efficiency in addition to low wear and tear on the actuators, small penalties were put on the magnitude of the commanded thrust $|n|$ (weighted with $c_{|n|}$) in addition to the derivatives of the commanded thrust $\dot{n}$ and angles $\dot{\alpha}$ (weighted with $c_{\dot{n}}$ and $c_{\dot{\alpha}}$, respectively). The resulting actuator penalties are shown in Eq. (20), where contributions from all three thrusters were summed together.

$$R_{act} = -\sum_i^3 \left(c_{|n|,i}|n_i| + c_{\dot{n},i}\dot{n}_i + c_{\dot{\alpha},i}\dot{\alpha}_i\right), \tag{20}$$

These contributions result in the total reward function that was used for rewarding the agent at each time step during training,

$$R_{tot} = R_{gauss} + R_{AS} + R_{vel} + R_{act}. \tag{21}$$

[1] Σ is usually denoted as a covariance matrix, but for purposes in this paper it was only used for tuning, and did not represent any covariances.

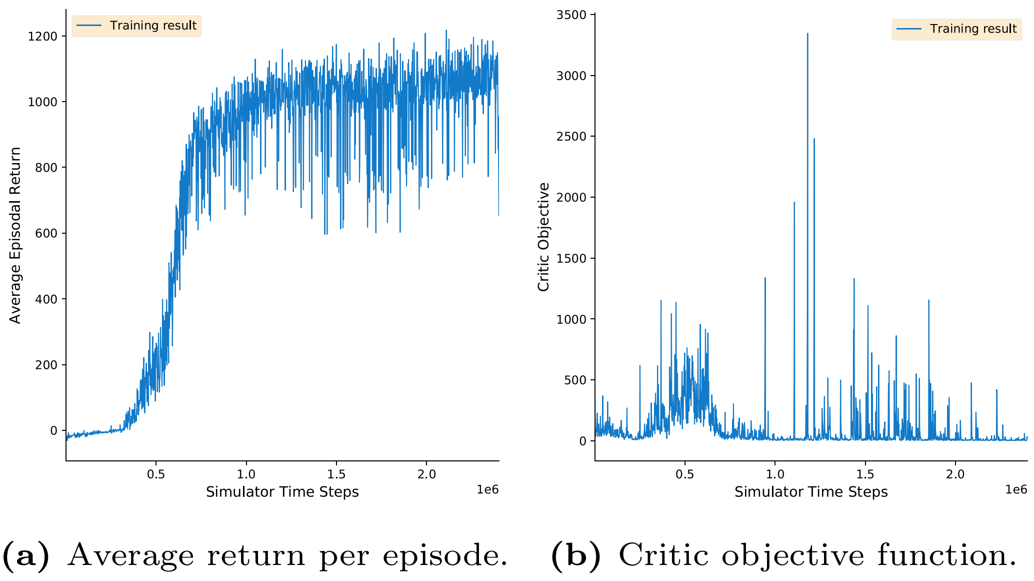


Fig. 7. Resulting training plots using the PPO algorithm.

3.3. Training procedure

The training was done by randomly initializing the pose of the vessel in various locations around a setpoint within a radius of 2.5 vessel lengths and within ±45° of the setpoint's heading. The agent collected experiences as state–action pairs from each time step by interacting with the environment. It should be noted that during training, the DRL method was updated on the same Windows computer that ran the simulator, hence receiving perfect information about the states directly from the CyberSea simulation environment. The critic and actor networks were updated according to Eqs. (11) and (12), respectively, when experience from 1600 time steps was collected. In total, updates to the networks were performed with state–action pairs from 2.4×10^6 time steps, using an implementation of the PPO algorithm based on OpenAI's repository[2] with a stochastic actor network for which the gaussian action sampling noise was completely reduced at the end of training. The actor critic architecture was set up using independent ANNs for the actor and the critic, both using the state vector $\boldsymbol{s}_t \in \mathbb{R}^9$ as input. The networks used three fully connected hidden layers with 80 neurons, and the output of the actor at time step t was $\boldsymbol{a}_t \in \mathbb{R}^7$, while the output from the critic was $V_{\theta_c}(s_t) \in \mathbb{R}^1$. Both networks used the leaky ReLU activation functions for nonlinearities, and the ADAM optimizer for parameter updates.

The average return obtained per episode (having an optimal value of 1400) and the objective function of the critic (optimal value of 0) are shown in Fig. 7. The average episodal return fluctuated due to the random initialization of the vessel's state between each trajectory within each episode, albeit improving steadily on average.

4. Results and discussion

To evaluate the performance of the controllers, a four-corner test was used. The four-corner test was performed by changing setpoints to different corners of a square, illustrated in Fig. 8 and with coordinates as listed in Table 3.

During testing, only the actor ANN was used as control policy in the DRL method, disabling the stochastic noise from OpenAI's implementation. Differing from the training procedure, the actor ANN was loaded into the control system in ROS on a Linux computer, receiving the *estimated* states from the EKF state estimator, which estimated the states from the sensor inputs coming from either the digital twin from the CyberSea environment during simulations, or the actual model ship during sea trials. In simulation, the controllers' robustness against external disturbances was evaluated during the four-corner test while enabling current forces. The current was irrotational and non-fluctuating with velocity $v_c = 0.2$ m/s and direction $\beta_c = 135°$ (from North-West). No information about the environmental loads were given to the control system, and the tests were started when the vessel had been standing still for 30 s. In the sea trial, negligible environmental loads were present.

Table 3
Four-corner coordinate specifications.

Setpoint	Time [s]	North [m]	East [m]	ψ [deg]
0	0–10	0	0	0
1	10–80	5	0	0
2	80–150	5	−5	0
3	150–190	5	−5	−45
4	190–270	0	−5	−45
5	270–350	0	0	0

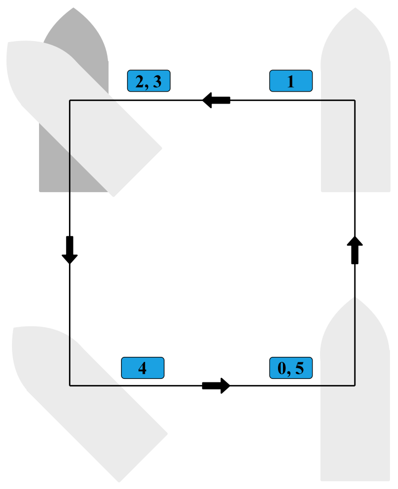
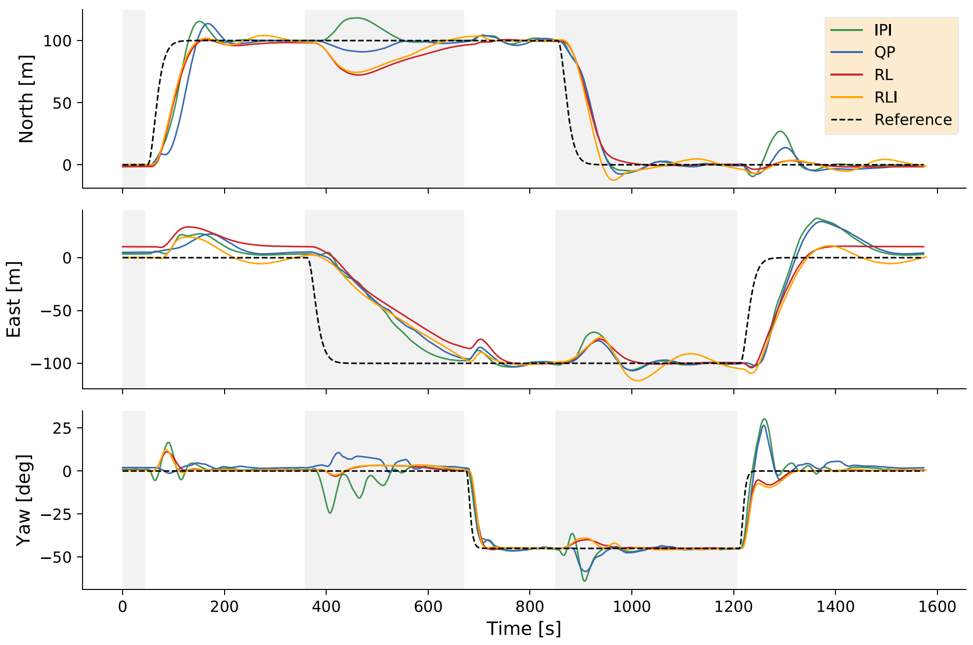
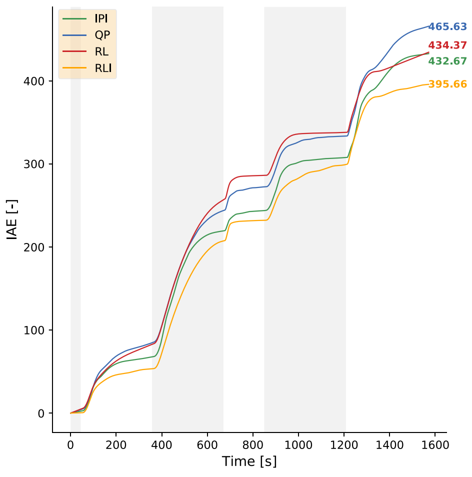


Fig. 8. Vessel poses in the four-corner test.

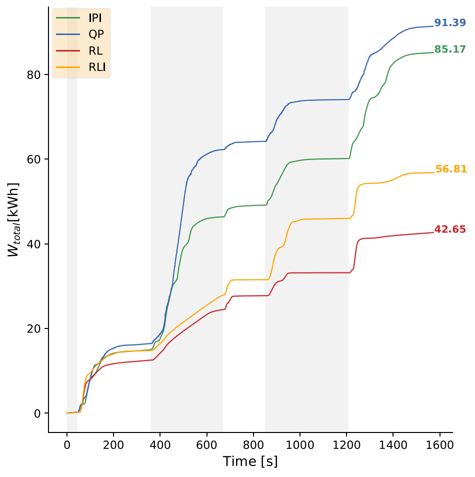
[2] https://spinningup.openai.com/ [last accessed 06. June 2020].



(a) Pose.



(b) IAE.



(c) Energy usage.

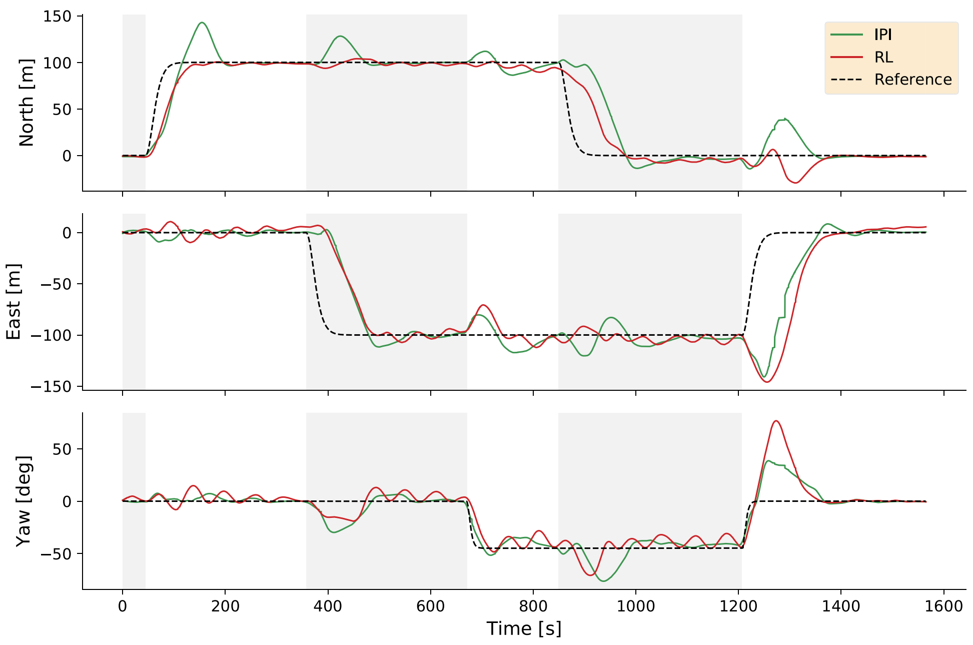
Fig. 9. Results of four-corner test during simulations.

In order to evaluate the effect of an integral controller, an integral effect was added to the state vector of the DRL method, augmenting the body-frame error pose in the state vector according to $\hat{\boldsymbol{\eta}}_t = \boldsymbol{\eta}_t + \hat{\boldsymbol{\eta}}_{ss,t}$, where the discrete integration was calculated according to $\hat{\boldsymbol{\eta}}_{ss,t} = \hat{\boldsymbol{\eta}}_{ss,t-1} + \Delta t k \boldsymbol{\eta}_t$. Δt is the time step, and k is a coefficient deciding the speed of the integral accumulation. A windup guard was also added to prevent overshoots. During the sea trial, the same four-corner test coordinates were used. The simulation was performed using the DRL method with and without integral effect. It was also benchmarked against two other methods, the first being a motion control law consisting of feedforward + a PID feedback controller developed in Alfheim and Muggerud (2016), combined with a pseudoinverse-based TA, and a QP based TA developed in Øvereng (2020). The acronyms for the methods used in the plots going forward were as shown in Table 4.

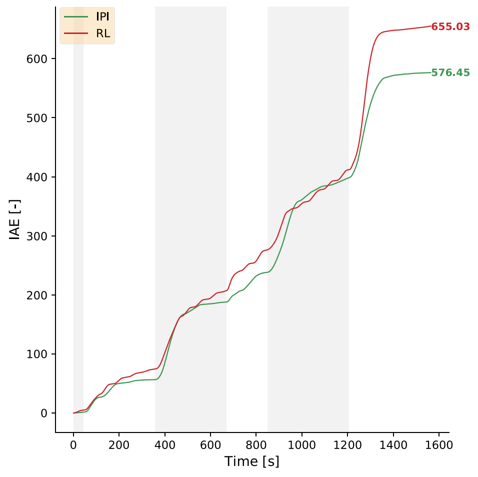
To evaluate performance, the Integral of Absolute Error (IAE) metric was used as a measurement of how accurate the DP system was in terms of reducing the body-frame error between the current pose and the desired pose. The IAE metric was chosen due to its convenience when testing with several setpoints in the same scenario due to its independence of time, compared to other popular metrics such as the Integral of Time-weighted Absolute Error (ITAE) metric. The lower the IAE score, the more accurate a method is evaluated as. The measurement $\tilde{\boldsymbol{\eta}}$ was used, where the body-frame errors was normalized for making the IAE a dimensionless number, dividing the respective deviations with [5 m, 5 m, 25°], meaning that a 5 degree deviation was weighted

Table 4
List of control methods used, with acronym.

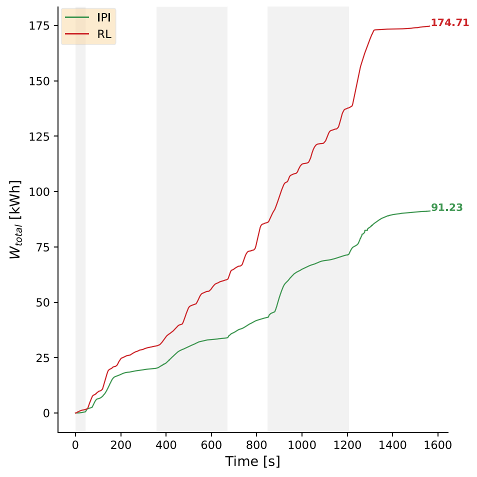
Case	Control method	Acronym
1	DRL without integral effect	RL
2	DRL with integral effect	RLI
3	PID motion control + pseudoinverse TA	IPI
4	PID motion control + QP TA	QP



(a) Pose.



(b) IAE.



(c) Energy usage.

Fig. 10. Results of four-corner test during sea trial.

equally as a 1 m deviation (in model scale).

$$IAE(t) = \int_0^t \sqrt{\tilde{\bar{\eta}}(\sigma)^\top \tilde{\bar{\eta}}(\sigma)}\; d\sigma. \tag{22}$$

The energy usage was calculated by integrating the power for each of the commanded RPS-signals. The power as a function of a propeller's revolution per second (RPS), n, is given as shown under the integral in Eq. (23), where ρ is the sea water density, D is the propeller diameter, and K_Q is the propeller torque coefficient (found through model scale tests in Alfheim and Muggerud (2016)). Lower W means better fuel efficiency.

$$W(t) = \int_0^t 2\pi\rho K_Q D^5 \,\mathrm{sgn}(n(\sigma))n(\sigma)^3 \; d\sigma. \tag{23}$$

Note that both the simulations and sea trials were carried out with the model scaled vessel. The results have been scaled to full scale by using Froude number scaling, meaning that it shows results comparable to other 60 meter vessels of similar size and shape. For details on Froude number scaling in hydrodynamic experiments, the reader is referred to Islam et al. (2016).

4.1. Simulation

The simulation results are displayed in Fig. 9. It was observed that only the DRL method with integral effect was able to totally remove the steady state body-frame error in *all* setpoints, while the classic methods displayed a slight steady state deviation in some setpoints (likely due to control scheme tuning from previous work on ReVolt). The DRL method without integral effect struggled with removing steady-state errors in sway. The classic methods and the DRL method with integral effect showed some overshoots of the setpoints, but only the classic methods showed oscillations (particularly in yaw) before settling on a setpoint. This leads to questioning the quality of the existing implementation of the motion control law on ReVolt, which both the IPI and the QP allocation depended on. It was believed that this came from the fact that the motion controller was tuned on board the physical model of

Table 5
Final reward function coefficients.

c_{gauss}	c_{AS}	c_{const}	σ_d
2.0	0.1	0.5	1.0
σ_ψ	c_u	c_v	$c_{\tilde{\psi}}$
5.0	0.5	0.5	1.0
$c_{\lvert n\rvert,bow}$	$c_{\lvert n\rvert,port}$	$c_{\lvert n\rvert,star}$	$c_{\dot{n},bow}$
0.2	0.3	0.3	0.05
$c_{\dot{n},port}$	$c_{\dot{n},star}$	$c_{\dot{\alpha},port}$	$c_{\dot{\alpha},star}$
0.05	0.05	0.01	0.01

ReVolt in previous work, thus performing poorly in simulation. The QP allocation resulted in having the highest values of IAE and energy usage. The DRL method with integral effect resulted in the lowest values of both metrics, as it resulted in 15% lower IAE and 38% less energy usage compared to the QP allocation, while also having 8% lower IAE and 33% less energy usage compared to the IPI allocation.

4.2. Sea trial

The sea trials were performed with the ReVolt in model scale as presented in Section 2. First, a measurement was taken of the time the ANN used to perform a calculation on the computer on board the model ship. The maximum time between two calculations was 0.015 s, indicating that using ANN based approaches is not likely to suffer from computation time.

On the model ship, it was found that the bow thruster experienced high friction between the propeller and the propeller housing, causing the propeller to get stuck unless being commanded approximately 50% thrust. This came at the cost of overshoots and/or oscillations in sway and yaw due to the differences between the digital twin used in training and the actual model ship. It was also found that the bow thruster's propeller was unsymmetrical, yielding a maximum force when exerting forces towards starboard (when locked to 90°) of 2.4 times the size of the maximum when exerting forces towards port. The DRL method had however been trained with symmetrical bow thruster parameters. Due to time constraints following these experiences, the four-corner test was performed with only the DRL method without any integral effect on the state vector, and the motion controller with the pseudoinverse based allocation method as a baseline.

The results from the four-corner test in Fig. 10 showed that the DRL method was able to reach all setpoints in approximately the same time in the sea trial as in the simulation, and reached the setpoints without much overshoot except when the vessel was traveling with sway motions (at 80 and 270 s), which was also the case for the IPI method. This was presumably due to the bow thruster issue, being more severe for the DRL method which had learned to use the bow thruster more extensively than classic methods in simulation in order to be energy efficient. The oscillatory movements were both coming from the bow thruster issue, in addition to that the stern thrusters had to compensate in order to follow the reference signal. While both methods displayed overshoots of the setpoint changes, the oscillatory behavior, especially in heading, caused the resulting IAE value to be larger for the DRL method than for the classic IPI method. The increased use of all thrusters also made the DRL method significantly less energy effective, mainly due to the compensating stern thrusters, both compared to results from simulation, and to the baseline controller in the sea trial. As the issue with high friction in the bow thruster indicated that the DRL method was less robust to changes in the dynamics between simulation and real life, it also suggests that further training of the neural networks on board the real vessel would improve performance.

5. Conclusion

This paper presented the implementation of the PPO algorithm for developing a DRL control scheme for applications to low speed control problems such as DP. The learning process benefited greatly from including prediction of sines and cosines of the stern thrusters' angles, and by using a multivariate, Gaussian reward function with an additional element to combat sparsity.

The test scenarios were considered satisfactory when compared to traditional methods, where simulations proved that the performance of the DRL method was both accurate and energy efficient, able to suppress steady state deviations when combined with an integrated state vector. The DRL method also showed positional accuracy when employing it to a physical model in a sea trial, deemed excellent considering that the DRL method was trained in simulation with perfect information about the vessel's state, zero environmental forces, and no thruster issues, which was not the case during the sea trial. The proposed method also solves the issue of computational complexity, as the computational time of the control scheme's neural network was negligible.

The work concludes that DRL's potential for accurate and energy efficient control is realizable and proposes that the presented DRL method is a strong contender when looking for new methods for station-keeping and low speed maneuvering in a way that encapsulates both the motion control and thrust allocation. For future work, it is suggested to focus on providing stability guarantees of DRL systems, and to explore *continual learning* as a method for tuning a DRL model from simulations to the real life model by continuing the training process on real life data in order to adapt to differences in the system dynamics and actuator characteristics between simulation and real life.

CRediT authorship contribution statement

Simen Sem Øvereng: Writing- original draft, Methodology, Software, Validation. **Dong Trong Nguyen:** Main supervisor, Reviewing and editing. **Geir Hamre:** Co-supervisor.

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.oceaneng.2021.109433. Dong Trong Nguyen reports financial support was provided by AMOS, NTNU.

Acknowledgments

This paper is based on the work of a master's thesis (Øvereng, 2020) which was written in collaboration with DNV. The work was sponsored by the Research Council of Norway through the Centre of Excellence funding scheme, project number 223254, AMOS. All implementations can be found at https://github.com/simensov/ml4ca.

Appendix. Reward function coefficients

The final coefficients for the reward function used during training of the DRL agent in this paper is given in Table 5.

References

Alfheim, H., Muggerud, K., 2016. Dynamic Positioning of the ReVolt Model-Scale Ship (Master's thesis). NTNU, https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2595418.

Arditti, F., Cozijn, H., van Daalen, E., Tannuri, E., 2018. Robust thrust allocation algorithm considering hydrodynamic interactions and actuator physical limitations. J. Mar. Sci. Technol. 24, http://dx.doi.org/10.1007/s00773-018-0605-8.

Balchen, J., Jenssen, N., Eldar, M., Saelid, S., 1980. A dynamic positioning system based on Kalman filtering and optimal control. Model. Identif. Control 1 (3), 135–163. http://dx.doi.org/10.4173/mic.1980.3.1.

Bohn, E., Coates, E.M., Moe, S., Johansen, T.A., 2019. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization. In: International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, http://dx.doi.org/10.1109/icuas.2019.8798254.

Chen, C., Chen, X.Q., Ma, F., Zeng, X.J., Wang, J., 2019. A knowledge-free path planning approach for smart ships based on reinforcement learning. Ocean Eng. 189, 106299. http://dx.doi.org/10.1016/j.oceaneng.2019.106299.

Cui, Y., Osaki, S., Matsubara, T., 2019. Reinforcement learning boat autopilot: A sample-efficient and model predictive control based approach. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2868–2875. http://dx.doi.org/10.1109/IROS40897.2019.8967630.

DNV, 2015. The ReVolt - A new inspirational ship concept. https://www.dnvgl.com/technology-innovation/revolt/index.html (visited on 2019/10/01).

Du, J., Xin, H., Krstic, M., Sun, Y., 2018. Dynamic positioning of ships with unknown parameters and disturbances. Control Eng. Pract. 76, 22–30. http://dx.doi.org/10.1016/j.conengprac.2018.03.015.

Fossen, T.I., 2011. Handbook of Marine Craft Hydrodynamics and Motion Control. John Wiley & Sons.

Fossen, T.I., Grøvlen, A., 1998. Nonlinear output feedback control of dynamically positioned ships using vectorial observer backstepping. IEEE Trans. Control Syst. Technol. 6 (1), 121–128. http://dx.doi.org/10.1109/87.654882.

Fossen, T.I., Strand, J.P., 1999. Passive nonlinear observer design for ships using lyapunov methods: full-scale experiments with a supply vessel. Automatica (ISSN: 0005-1098) 35 (1), 3–16. http://dx.doi.org/10.1016/S0005-1098(98)00121-6.

Guo, S., Zhang, X., Zheng, Y., Du, A.Y., 2020. An autonomous path planning model for unmanned ships based on deep reinforcement learning. Sensors 20 (2), 426. http://dx.doi.org/10.3390/s20020426.

Gupta, A., Bhartiya, S., Nataraj, P., 2011. A novel approach to multiparametric quadratic programming. Automatica (ISSN: 0005-1098) 47 (9), 2112–2117. http://dx.doi.org/10.1016/j.automatica.2011.06.019.

Islam, M., Jahra, F., Hiscock, S., 2016. Data analysis methodologies for hydrodynamic experiments in waves. J. Nav. Archit. Mar. Eng. 13, 1. http://dx.doi.org/10.3329/jname.v13i1.25347.

Jenssen, N.A., Realfsen, B., 2006. Power Optimal Thruster Allocation. Kongsberg Maritime AS, https://pdfs.semanticscholar.org/064e/1b38204f5d855266a4eeb4af7b705a9bb2b9.pdf (visited on 2019/11/18).

Johansen, T.A., Fossen, T.I., 2013. Control allocation - A survey. Automatica 46, 1087–1103. http://dx.doi.org/10.1016/j.automatica.2013.01.035.

Johansen, T.A., Fossen, T.I., Tøndel, P., 2005. Efficient optimal constrained control allocation via multiparametric programming. J. Guid. Control Dyn. 28 (3), 506–515. http://dx.doi.org/10.2514/1.10780.

Johansen, T.A., Fuglseth, T.P., Tøndel, P., Fossen, T.I., 2008. Optimal constrained control allocation in marine surface vessels with rudders. Control Eng. Pract. (ISSN: 0967-0661) 16 (4), 457–464. http://dx.doi.org/10.1016/j.conengprac.2007.01.012, Special Section on Manoeuvering and Control of Marine Craft.

Katebi, M., Yamamoto, I., Matsuura, M., Grimble, M., Hirayama, H., Okamoto, N., 2001. Robust dynamic ship positioning control system design and applications. Internat. J. Robust Nonlinear Control 11, 1257–1284. http://dx.doi.org/10.1002/rnc.605.

Kjærnli, E.F., 2018. Deep Reinforcement Learning Based Controllers In Underwater Robotics (Master's thesis). NTNU, https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2615067.

Knudsen, K.B., 2019. Deep Learning for Station Keeping of AUVs (Master's thesis). NTNU, https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2622920.

Koch, W., Mancuso, R., West, R., Bestavros, A., 2018. Reinforcement learning for UAV attitude control. ACM Trans. Cyber-Phys. Syst. 3, http://dx.doi.org/10.1145/3301273.

Kokotovic, P., 1992. The joy of feedback: nonlinear and adaptive. IEEE Control Syst. 12, 7–17. http://dx.doi.org/10.1109/37.165507.

Leavitt, J., 2008. Optimal thrust allocation in DP systems. In: Dynamic Positioning Committee, L-3 Communications DPCS, MTS. Dynamic Positioning Conference. https://dynamic-positioning.com/proceedings/dp2008/thrusters_leavitt_pp.pdf (visited on 2019/09/17).

Luman, Z., Roh, M.I., Hong, J.W., 2015. Optimal thrust allocation for dynamic positioning of deep-sea working vessel. J. Adv. Res. Ocean Eng. 1, 94–105. http://dx.doi.org/10.5574/JAROE.2015.1.2.094.

Martinsen, A.B., Lekkas, A.M., 2018. Curved path following with deep reinforcement learning: Results from three vessel models. In: OCEANS 2018 MTS/IEEE Charleston. pp. 1–8. http://dx.doi.org/10.1109/OCEANS.2018.8604829.

Martinsen, A.B., Lekkas, A.M., Gros, S., Glomsrud, J.A., Pedersen, T.A., 2020. Reinforcement learning-based tracking control of USVs in varying operational conditions. Front. Robot. AI (ISSN: 2296-9144) 7, 32. http://dx.doi.org/10.3389/frobt.2020.00032.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv:1312.5602.

Naderi, M., Khaki Sedigh, A., Johansen, T., 2019. Guaranteed feasible control allocation using Model Predictive Control. Control Theory Technol. 17, 252–264. http://dx.doi.org/10.1007/s11768-019-7231-90.

Nguyen, T.D., Pivano, L., Børhaug, B., Smogeli, Ø., 2013. Dynamic station-keeping capability analysis using advanced vessel simulator. In: Proceedings of the 54th SIMS Conference on Simulation and Modelling, Bergen, Norway. pp. 84–102, URL https://www.scansims.org/events.php?sid=13&src=db1557571001&udpview=show-conference.

Nguyen, T.D., Sorensen, A.J., Quek, S.T., 2008. Multi-operational controller structure for station keeping and transit operations of marine vessels. IEEE Trans. Control Syst. Technol. 16 (3), 491–498. http://dx.doi.org/10.1109/TCST.2007.906309.

Nguyen, T.D., Sørensen, A.J., Tong Quek, S., 2007. Design of hybrid controller for dynamic positioning from calm to extreme sea conditions. Automatica (ISSN: 0005-1098) 43 (5), 768–785. http://dx.doi.org/10.1016/j.automatica.2006.11.017.

Øvereng, S.S., 2020. Dynamic Positioning using Deep Reinforcement Learning (Master's thesis). NTNU, https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2731248.

Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P., 2015. High-dimensional continuous control using generalized advantage estimation. https://arxiv.org/abs/1506.02438.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. https://arxiv.org/abs/1707.06347.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D., 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science (ISSN: 0036-8075) 362 (6419), 1140–1144. http://dx.doi.org/10.1126/science.aar6404.

Skjetne, R., Fossen, T., 2004. On integral control in backstepping: Analysis of different techniques. In: Proceedings of the 2004 American Control Conference, Vol. 2. pp. 1899–1904. http://dx.doi.org/10.23919/ACC.2004.1386858.

Skulstad, R., Li, G., Zhang, H., Fossen, T.I., 2018. A neural network approach to control allocation of ships for dynamic positioning. In: IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS, Vol. 51. pp. 128–133. http://dx.doi.org/10.1016/j.ifacol.2018.09.481, ISSN 2405-8963.

Sørdalen, O.J., 1997. Optimal thrust allocation for marine vessels. Control Eng. Pract. 5 (9), 1223–1231. http://dx.doi.org/10.1016/S0967-0661(97)84361-4.

Sørensen, A.J., 2018. Marine Cybernetics. Towards Autonomous Marine Operations and Systems. Department of Marine Technology, NTNU, unpublished.

Sørensen, A., Sagatun, S., Fossen, T., 1996. Design of a dynamic positioning system using model-based control. Control Eng. Pract. (ISSN: 0967-0661) 4 (3), 359–368. http://dx.doi.org/10.1016/0967-0661(96)00013-5.

Sutton, R.S., Barto, A.G., 2017. Reinforcement Learning: An Introduction, second ed. MIT Press, http://incompleteideas.net/book/the-book.html.

Tannuri, E., Agostinho, A.C., Morishita, H.M., Moratelli, L., 2010. Dynamic positioning systems: An experimental analysis of sliding mode control. Control Eng. Pract. (ISSN: 0967-0661) 18 (10), 1121–1132. http://dx.doi.org/10.1016/j.conengprac.2010.06.007.

Tannuri, E.A., Kubota, L.K., Pesce, C.P., 2006. Adaptive techniques applied to offshore dynamic positioning systems. J. Braz. Soc. Mech. Sci. Eng. (ISSN: 1678-5878) 28, 323–330.

Veksler, A., Johansen, T.A., Borrelli, F., Realfsen, B., 2016. Dynamic positioning with model predictive control. IEEE Trans. Control Syst. Technol. (ISSN: 1558-0865) 24, 1340–1353. http://dx.doi.org/10.1109/TCST.2015.2497280.

Vermillion, C., Sun, J., Butts, K., 2007. Model predictive control allocation for overactuated systems - Stability and performance. In: IEEE Conference on Decision and Control, Vol. 47. pp. 1251–1256. http://dx.doi.org/10.1109/CDC.2007.4434722.

Wu, D., Ren, F., Zhang, W., 2016. An energy optimal thrust allocation method for the marine Dynamic Positioning system based on adaptive hybrid artificial bee colony algorithm. Ocean Eng. 118, 216–226. http://dx.doi.org/10.1016/j.oceaneng.2016.04.004.

Xu, S., Wang, X., Yang, J., Wang, L., 2019. A fuzzy rule based PID controller for dynamic positioning of vessels in variable environmental disturbances. J. Mar. Sci. Technol. 25, 914–924. http://dx.doi.org/10.1007/s00773-019-00689-2.