Robotics and Autonomous Systems 101 (2018) 93–102

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

RL-based path planning for an over-actuated floating vehicle under disturbances

Konstantinos Blekas *, Kostas Vlachos

Department of Computer Science and Engineering, University of Ioannina, Ioannina, 45110, Greece

ARTICLE INFO

Article history:
Received 16 March 2017
Received in revised form 20 October 2017
Accepted 18 December 2017
Available online 30 December 2017

Keywords:
Reinforcement learning
Over-actuated control
Marine vehicle
Autonomous navigation

ABSTRACT

This paper investigates the use of reinforcement learning for the path planning of an autonomous triangular marine platform in unknown environments under various environmental disturbances. The marine platform is over-actuated, i.e. it has more control inputs than degrees of freedom. The proposed approach uses a high-level online least-squared policy iteration scheme for value function approximation in order to estimate sub-optimal policy. The chosen action is considered as the desired input to a fast and efficient low-level velocity controller. We evaluate our approach in a simulated environment, including the dynamic model of the platform, the dynamics and limitations of the actuators, and the presence of wind, wave, and sea current disturbances. Simulation results are presented that demonstrate the performance of the proposed approach. Despite the model dynamics, the actuation dynamics and constrains, and the environmental disturbances, the presented results are promising.

1. Introduction

Marine vehicle motion planning aims at finding a route through obstacles and constructing a motion planner in terms of a feasible sequence of actions that allow to move a marine vehicle from an initial "configuration" to a goal "configuration". Ideally, such planner tries to optimize an objective function consisting of attributes such as plan duration, energy consumption, etc.

There is a tremendous need for developing fast analytic algorithms for predicting the collision probability due to model uncertainty and random disturbances in the environment for a planar vehicle such as a marine surface vessel [1,2]. These predictions lead to a robust motion planning algorithm that discovers the (sub)-optimal motion plan quickly and efficiently. Incorporating model learning into the predictions exhibits emergent active learning strategies to safely and effectively complete the mission.

A flexible framework for motion planning and autonomous vehicle navigation is through Reinforcement Learning (RL) [3,4]. RL aims at controlling an autonomous agent in unknown stochastic environments. Typically, the environment is modeled as a Markov Decision Process (MDP), where the agent receives a scalar reward signal that evaluates every transition. The objective is to maximize its long-term profit that is equivalent to maximizing the expected total discounted reward. Value function is used for measuring the quality of a policy, which associates to every state the expected discounted reward when starting from this state and all decisions are made following the particular policy. A plethora of methods have been proposed during the last two decades using a variety of value-function estimation techniques [4,5].

The temporal difference algorithms provide a nice framework for policy evaluation since they have the flexibility to handle large or continuous state space of real world applications. More specifically, least-squares temporal difference (LSTD) family of methods is very popular mechanism for approximating the value function that performs an iterative procedure for optimal policy estimation. Finally, *model-based* approaches for value function approximation have been also proposed based on on-line schemes, through Gaussian processes [6], clustering schemes [7], or regression tree models [8,9].

In the literature there are some marine robotic applications, mostly involving autonomous underwater vehicles (AUV), using reinforcement learning, see for a survey in [1]. In [10] for example a neural networks-based reinforcement learning scheme is presented for high-level control of AUV's. In [11] another approach is proposed for motion planning of under-actuated AUV in unknown non-uniform sea flow. A recent work, presented in [12], proposes a path planning algorithm for the kinematic model of an under-actuated marine vehicle in the presence of sea current disturbances, based on reinforcement learning.

In this paper, we focus on the development of an intelligent autonomous navigation scheme based on reinforcement learning, for a novel over-actuated autonomous triangular marine platform shown in Fig. 1. The required forces and moment for the motion

* Corresponding author.
E-mail addresses: kblekas@cse.uoi.gr (K. Blekas), kostaswl@cse.uoi.gr (K. Vlachos).

https://doi.org/10.1016/j.robot.2017.12.009
0921-8890/

Fig. 1. The triangular marine platform during construction. *Source:* http://www.inp.demokritos.gr/nestor/dberenike/index.html.

of the platform are provided by three rotating pump jets, consequently the system is over-actuated, i.e., it has more control inputs than degrees of freedom (DOF). A proper control allocation scheme is implemented, see [13], to allow for optimal allocation of the effort without violating thruster capabilities. Controllers for the problem of the autonomous dynamic positioning of the platform have been proposed in [13], and [14]. The detailed description of the platform can be found in [13]. Here, we examine the problem of the determination, and realization of a desired path in an unknown environment under various environmental disturbances, and actuation constraints. These include (i) sea current disturbance forces/torque, (ii) wind disturbance forces/torque, (iii) wave disturbance forces/torque, (iv) state measurement noise, (v) actuation limits, and (vi) actuation delays.

The proposed scheme comprises two layers. The first layer is an on-line reinforcement learning algorithm which is based on least square policy iteration (LSPI) [15,16], and aims at the determination of a sub-optimal path in the presence of realistic environmental wind, wave and sea current disturbances. The output of the reinforcement learning algorithm is the desired direction of the velocity of the marine vehicle. This is fed, as desired input, to the second layer, a low-level velocity PD controller that achieve and maintain the desired velocity, against environmental, and other disturbances.

Simulation results show that the generated path is tracked successfully by the marine platform, despite the disturbances. The dynamics of the marine platform, and the dynamics and limitations of the actuation system are modeled into the simulation environment. One of the main advantages of this method is that it is model free, meaning that in the design process we did not use any explicit knowledge about the system model. This makes the method robust to model uncertainties and noise, as it is demonstrated in our results. Another advantage is that it can be implemented as an online learning algorithm which brings us a step further towards fully autonomous marine platform.

In this paper we build upon our previous work presented in [17], and [18], where initial results are reported. With respect to [17], and [18], this work (i) implements a more comprehensive RL agent suited to the task, (ii) proposes an enhanced low-level controller, taking under consideration the disturbances acting on the platform, to ensure that the desired velocity commanded by the RL agent, is realized fast and efficiently, (iii) uses a more complex model of environmental and measurement disturbances, and (iv) makes a more comprehensive and extensive study of error tolerance and sensor failure scenarios of the system during navigation, by presenting suitable solutions.

2. The marine platform

The marine platform is designed to assist in the deployment of a deep-sea cubic kilometer neutrino telescope, see http://www.inp.demokritos.gr/nestor/. A comprehensive description of the platform can be found in [13]. However, for the sake of completeness, a brief description of the platform is given next.

It consists of an isosceles triangular structure mounted on three hollow double-cylinders, one at each corner of the structure, see Fig. 1. The plane of the triangle is parallel to the sea surface. The cylinders provide the necessary buoyancy, as part of them is immersed in the water. The platform actuation is realized using three fully submerged pump-jets, located at the bottom of each cylinder. Diesel engines drive the pumps, while electro-hydraulic motors rotate the jets providing vectored thrust, [13]. Considering only the platform's planar motion, the kinematics equations are described by:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}u\\ v\\ r\end{bmatrix} \Rightarrow {}^{I}\dot{\mathbf{x}} = {}^{I}\mathbf{R}_{B}\,{}^{B}\mathbf{v} \tag{1}$$

where, x and y represent the platform inertial coordinates, and ψ describes the orientation of the body-fixed frame {B}. u and v are the surge and sway velocities, and r is the yaw angular velocity of the platform.

We consider three types of forces acting on the center of mass (CM) of the platform: (a) the control forces/torque from the jets, (b) the hydrodynamic forces due to the motion of the cylinders with respect to the moving water, and (c) the disturbance forces/torque due to wind, wave, and sea current.

The vectored thrust from the jets, result in the control forces acting on the CM of the platform, and a torque about the vertical axis. They are generated according to:

$${}^{B}\mathbf{q}_{c} = [F_x, F_y, M_z]^{\top} = \mathbf{B}\,{}^{B}\mathbf{f}_{c} \tag{2}$$

$$\mathbf{B} = \begin{bmatrix}1 & 0 & 0\\ 0 & -1 & -d_{AG}\\ 1 & 0 & -d_{DC}\\ 0 & -1 & d_{DG}\\ 1 & 0 & d_{DC}\\ 0 & -1 & d_{DG}\end{bmatrix}^{\top}, \quad {}^{B}\mathbf{f}_{c} = \begin{bmatrix}J_A\sin\phi_A\\ J_A\cos\phi_A\\ J_B\sin\phi_B\\ J_B\cos\phi_B\\ J_C\sin\phi_C\\ J_C\cos\phi_C\end{bmatrix} \tag{3}$$

where ${}^{B}\mathbf{q}_{c}$ represents the control force/torque vector. $\mathbf{B}$ contains dimensional parameters, where d_{ij} denote the distance between points i, and j. A, B, C represent the three corners of the structure while D, and G are the midpoint of the triangle base, and the CM of the platform respectively. The J_A, J_B, and J_C in (3) denote the magnitudes of the thrusts while ϕ_A, ϕ_B, and ϕ_C denote the force directions. The vector ${}^{B}\mathbf{f}_{c}$ can be retrieved by the pseudoinversion of $\mathbf{B}$ in (2). The desired jet thrust and direction are calculated according to,

$$J_i = \sqrt{(f_i\sin\phi_i)^2 + (f_i\cos\phi_i)^2} \tag{4}$$

$$\phi_i = \arctan(f_i\sin\phi_i, f_i\cos\phi_i) \tag{5}$$

where $i = A, B, C$. Note that the desired jet thrust and direction cannot be applied immediately due to actuator dynamics and limitations. Based on the hardware specifications, we modeled the jet rotation dynamics as a first order lag with a time constant equal to 1 s, and the rotation speed is limited by an upper bound of 0.84 rad/s. In addition, the thrust model includes the dynamics of the diesel engine (diesel engine thrust time constant equal to 0.25 s), the diesel engine maximum torque (1323.00 Nm), and the maximum shaft speed limit (1600.00 rpm). These limits result to

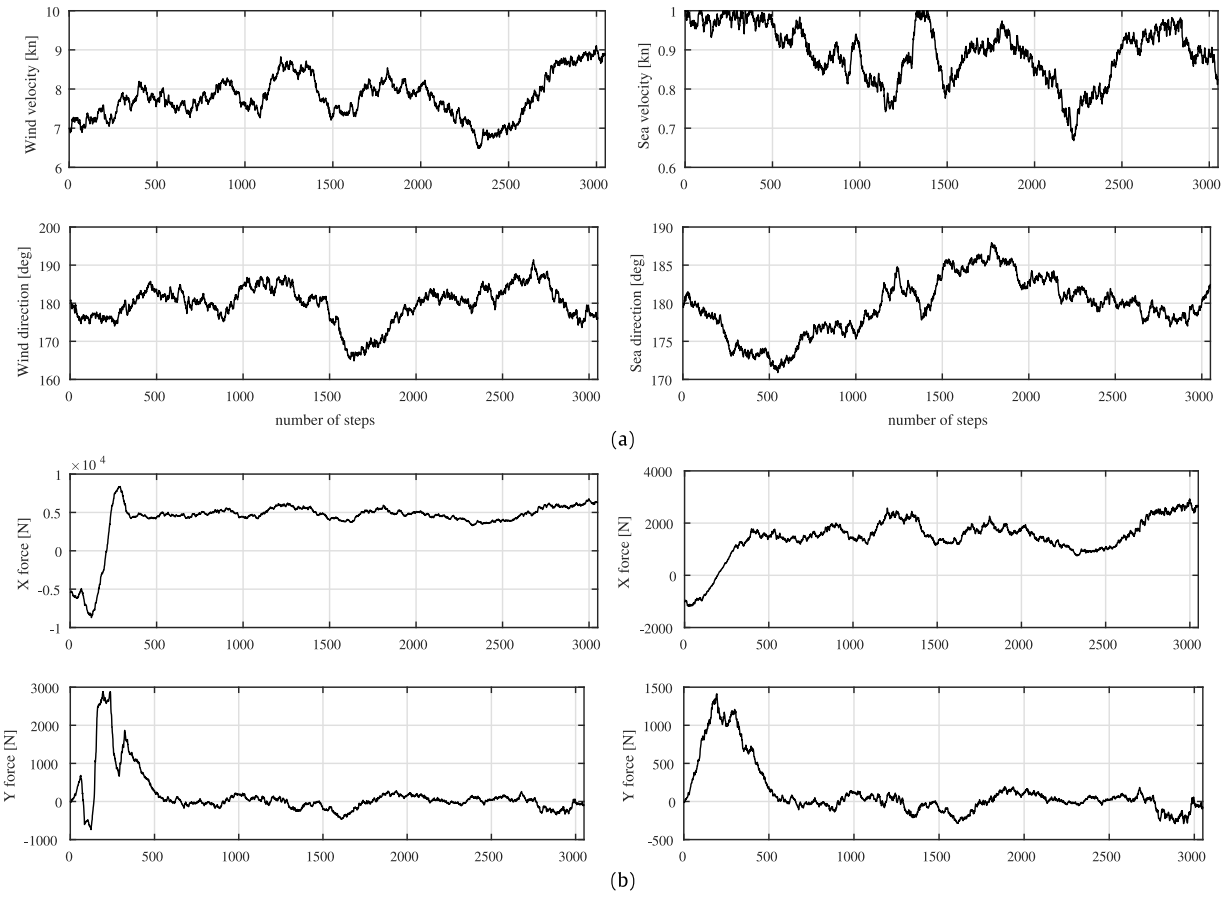


Fig. 2. (a) Wind and sea current velocity and direction. (b) Resultant wind and wave disturbance forces.

a maximum magnitude of thrust for each jet equal to 20 kN. A detailed description of the dynamics and limitations of the actuation system of the platform can be found in [13].

The hydrodynamic force acting on each submerged cylinder is the sum of the added mass force, and the drag force, see [19]. Each force result in a force acting on the platform CM and a moment about it. The terms that include the drag forces are collected in vector ${}^{B}\mathbf{q} = [f_x, f_y, n_z]^\top$ while the added mass is included in the mass matrix, $\mathbf{M}$, see [13] for a detailed description.

The disturbances acting on the platform are due to wind, wave (depended on wind velocity and direction), and sea current. We simulated the inertial wind, and sea current velocity magnitude and direction, by integrating Gaussian white noise. Example time series are shown in Fig. 2, together with the resultant wind and wave disturbance forces acting on the marine platform. The wind and sea current velocity upper bounds are equal to 7.9 m/s ($\approx$ 15 kN or 4 Beaufort), and 0.5 m/s ($\approx$ 1 kN), respectively. The resultant disturbance forces, and torque are collected in vector ${}^{B}\mathbf{q}_{dist} = [f_{dist,x}, f_{dist,y}, n_{dist,z}]^\top$. A detailed description of the generation of wind, wind depended wave, and sea current signals, and the calculation of the respected disturbance forces and torque can be found in [20].

Accordingly, the planar equations of motion of the platform, expressed in the body-fixed frame {B} is given by:

$$\mathbf{M}{}^{B}\dot{\mathbf{v}} = {}^{B}\mathbf{q} + {}^{B}\mathbf{q}_{dist} + {}^{B}\mathbf{q}_{c} \tag{6}$$

with

$$\mathbf{M} = \begin{bmatrix} m - 3m_a & 0 & 0 \\ 0 & m - 3m_a & 0 \\ 0 & 0 & m_{33} \end{bmatrix}, \tag{7}$$

$$m_{33} = I_{zz} - (d_{AG}^2 + 2d_{BD}^2 + 2d_{DG}^2)m_a, \tag{8}$$

where m is the mass of the platform, m_a is its added mass, and I_{zz} is its mass moment of inertia about the z_b axis.

Eq. (6) describes the model implemented in the simulated environment presented in Section 5. In addition, measurement noise is superimposed to the position and orientation of the platform. To this end, we extracted measurement noise from real GPS readings. The used GPS receivers have an accuracy of ± 1 m with an update frequency of 5 Hz.

3. Reinforcement learning for autonomous marine vehicle navigation

As mentioned in the introduction, the proposed scheme comprises two layers, and the first layer involves the Reinforcement Learning (RL) agent, see Fig. 3. The main part of the proposed decision system is based on a RL agent which receives the inertial coordinates (x, y) of the floating platform and makes an action related to the *direction* of its linear velocity. Note that the desired *magnitude* of the platform's velocity is constant and not affected by the RL agent. In addition, the orientation of the platform, ψ, is controlled by the low-level controller, see Section 4.

According to the RL framework, we consider that the environment where the marine platform acts can be modeled as a *Markov decision process* (MDP). An MDP can be described as a five-tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is a set of states; $\mathcal{A}$ a set of actions; $P : \mathcal{S} \times \mathcal{A} \times S \to [0, 1]$ is a Markovian transition model that specifies the probability $P(\boldsymbol{s}'|\boldsymbol{s}, a)$ of transition from state s to a new state $\boldsymbol{s}'$ after taking an action a. Finally $R : \mathcal{S} \to \mathbb{R}$ is the reward function

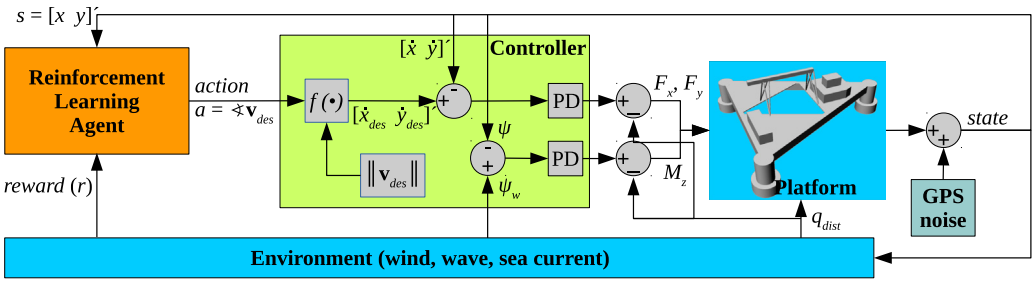


Fig. 3. The control scheme.

for a state–action pair, and $\gamma \in (0, 1)$ is the discount factor for future rewards. A *stationary policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is a mapping from states to actions and denotes a mechanism for choosing actions. An *episode* is a sequence of transitions: $(\boldsymbol{s}_1, a_1, r_1, \boldsymbol{s}_2, \ldots)$.

As noted before, in the proposed RL agent we have considered that the state space consists of the platform inertial coordinates, i.e. $\boldsymbol{s} = (x, y)$, while the action a is related to the direction of the marine platform velocity. In our case we have divided this direction into a set of five (5) discrete values: $a \in \{0, 45, 90, 135, 180\}$. The size of the action set affects the complexity of the methodology. In our application the chosen set is proven adequate. However, any given set of actions can be used.

The Q-function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ of the policy π gives for every state–action pair $(\boldsymbol{s}, a)$ the expected return when starting in s applying action a and following π thereafter. Q-values can be evaluated by solving the following set of linear Bellman equations:

$$Q^{\pi}(\boldsymbol{s}, a) = R(\boldsymbol{s}, a) + \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}, a) \max_{a} Q(\boldsymbol{s}', a). \tag{9}$$

The objective of RL problems is to estimate an optimal policy π^* by choosing actions that yield the appropriate action–state value function

$$Q^* : \pi^*(\boldsymbol{s}) = \arg\max_{a} Q^*(\boldsymbol{s}, a).$$

For determining the Q function we can use a value function approximation scheme. A common choice is to consider a linear model consisting of a set of k basis functions $\phi(\boldsymbol{s}, a) = [\phi_1(\boldsymbol{s}, a), \ldots, \phi_k(\boldsymbol{s}, a)]^{\top}$:

$$Q(\boldsymbol{s}, a) = \boldsymbol{\phi}(\boldsymbol{s}, a)^{\top} \boldsymbol{w} = \sum_{j=1}^{k} \phi_j(\boldsymbol{s}, a) w_j, \tag{10}$$

where $\boldsymbol{w} = (w_1, \ldots, w_k)$ is a vector of weights which are unknown and must be estimated so as to minimize the approximation error. The selection of the basis functions is very important and must be chosen to encode properties of the state and action relevant to the proper determination of the Q values.

In our work we have considered RBF basis functions:

$$\phi_j(\boldsymbol{s}) = \exp(-\beta_j \|\boldsymbol{s} - c_j\|^2), \tag{11}$$

for a given collection of k centers c_j and precision (inverse variance) β_j. Note that we have assumed common precision to all k functions, i.e. $\beta_j = \beta$. In order to obtain the parameters of these k basis functions we have adopted the *tile coding* scheme. In particular, we have partitioned the state space on k non-overlapping and equally in size (width $1/\beta_j$) regions. Then, their center (c_j) were found and used for constructing the feature space of the k kernels.

The proposed RL agent follows a policy iteration scheme for learning. Policy iteration is a dynamic programming algorithm, which starts with an arbitrary policy and steadily improves it. It discovers the optimal policy by generating a sequence of monotonically improving policies. The policy iteration algorithm manipulates the policy directly instead of finding it via the value function, as happens in the case of value iteration. Policy iteration consists of two successive, interactive phases: the *policy evaluation* where the value function of policy π is computed and the *policy improvement* where policy is updated. These two phases are executed iteratively until policy π cannot be further improved. In this case, the policy iteration algorithm converges to the optimal policy π^*.

Least-Squares Policy Iteration (LSPI) [15] is a known approximate policy iteration algorithm that uses a *model-free* version of the least-squares temporal difference learning (LSTD). The action-value function $Q(s, a)$, is approximated instead of the state-value function, while action selection and policy improvement are permitted without the need of any prior knowledge of the environment dynamics. In its original form, the LSPI is an off-line algorithm and requires a set of training examples: $D = \{s_i, a_i, r_i, s_i'|i = 1, \ldots, n\}$, which are used at each iteration in order to evaluate the derived policies. During the policy evaluation step, the matrix A (size $k \times k$) and the vector b (size $k \times 1$), are computed following the previously learned policy π, as follows:

$$A = \sum_{i=1}^{n} \phi(s_i, a_i)(\phi(s_i, a_i) - \gamma \phi(s_i', \pi(s_i')))^{\top}, \tag{12}$$

$$b = \sum_{i=1}^{n} \phi(s_i, a_i) r_i. \tag{13}$$

At the policy improvement step, matrix A and vector b are used in order to yield an improved policy. In this way, the least-squares projection error for the state-value function Q, is minimized as:

$$w = A^{-1} b. \tag{14}$$

The whole procedure is implemented iteratively, until a convergence criterion is satisfied. The model parameters are initialized arbitrarily or are set to 0.

Since our goal is to achieve an online learning scheme of the marine platform, in this work we employ an online variant of LSPI where policy improvement is performed every few transitions [16]. Based on (12) and (13) we can apply an incremental update scheme for the model parameters:

$$A_{t+1} = A_t + \phi(s_t, a_t)(\phi(s_t, a_t) - \gamma \phi(s_{t+1}, \pi(s_{t+1})))^{\top}, \tag{15}$$

$$b_{t+1} = b_t + \phi(s_t, a_t) r_t. \tag{16}$$

Then, the current policy is evaluated according to the least-squares estimation rule: $w = A_{t+1}^{-1} b_{t+1}$, and is applied to obtain new transition samples. Thus a new learning cycle repeats. In our study policy improvement was performed either every 10 transitions, or at the end of each episode.

Balancing the ratio of exploration/exploitation is a great challenge in reinforcement learning, since it may have significant impact on the quality of learned policy. A common choice is to employ the ϵ-greedy exploration scheme. According to this, at each learning step t an action is selected greedily, based on the estimated action-value function with probability $1 - \epsilon_t$, while a

random action is chosen with probability ϵ_t, ($\epsilon_t \in [0, 1]$). Initially, the parameter ϵ_0 is set to a large value (e.g., $\epsilon_0 = 0.7$), while it decays exponentially over time with a decay rate $\epsilon_d \in (0, 1)$. In the particular scheme, policy improvement can be implemented after a number of consecutive transitions.

4. Low-level controller design

The second layer of the proposed scheme involves the low-level controller. The goal of the low-level controller is (a) to ensure that the desired linear velocity commanded by the RL-agent, is realized fast and efficiently, (b) controls the orientation of the platform, and (c) achieve (a) and (b) despite the presence of system and measurement limits, and environmental disturbances. The following list summarizes all limits and disturbances that are included into the simulation environment.

- GPS error of ± 1 m
- GPS update delay of 0.2 s (5 Hz update frequency)
- Jet rotation speed limit equal to 0.84 rad/s
- Jet rotation time constant equal to 1 s
- Diesel engine thrust time constant equal to 0.25 s
- Diesel engine torque limit equal to 1323 Nm
- Shaft speed limit equal to 1600 rpm
- Maximum magnitude of thrust for each jet equal to 20 kN
- Wind disturbance forces/torque
- Wave disturbance forces/torque
- Sea current disturbances.

To this end, the control of the motion of the triangular platform is achieved using the system depicted in Fig. 3. In addition, in the controller design we take under consideration (a) the environmental disturbances, based on the knowledge of the wind speed, and direction, and (b) the model uncertainty. According to this, the desired forces/torque vector ${}^B\mathbf{q}_c$, is the output of a controller scheme with two independent closed loops. The first closed loop realize a velocity PD controller where the input is the desired velocity of the floating platform (as commanded by the RL-agent), and the output is the desired forces according to

$$[F_x, F_y]^\top = {}^I\mathbf{R}_B{}'(K_{p,f}([\dot{x}_{des} \quad \dot{y}_{des}]' - [\dot{x} \quad \dot{y}]') - K_{d,f}[\ddot{x} \quad \ddot{y}]') - l[f_{dist,x} \quad f_{dist,y}]' \tag{17}$$

where $K_{p,f}$ and $K_{d,f}$ are the controller gains related to the desired forces calculation, and $\dot{x}_{des}$ and $\dot{y}_{des}$ are the desired inertial linear velocities. $f_{dist,x}$, and $f_{dist,y}$ are the disturbance forces expressed in the body frame.

The second closed loop is a PD controller where the input is the desired orientation of the floating platform, and the output is the desired torque according to

$$M_z = K_{p,m}(\psi_w - \psi) - K_{d,m}r - l(n_{dist,z}) \tag{18}$$

where $K_{p,m}$, and $K_{d,m}$ are the controller gains related to the desired torque calculation, and ψ_w denotes the direction of the wind. $n_{dist,z}$ is the disturbance torque expressed in the body frame. As (18) suggests, the desired orientation of the platform coincides with the direction of the wind. This configuration results to reduced disturbance forces/torque, due to the reduction of the projected area of the platform to the wind. The desired inertial linear velocities, $\dot{x}_{des}$ and $\dot{y}_{des}$, are calculated using a constant velocity magnitude defined by design, and a desired direction, which is the action of the RL agent, presented in the previous section. The calculation of the disturbance forces/torque are based on the knowledge of the model parameters (dimensions of the platform), and on the assumption that the wind speed and direction can be measured. A detailed description of the calculation of the respected disturbance forces and torque can be found in [20]. Moreover, we use the error

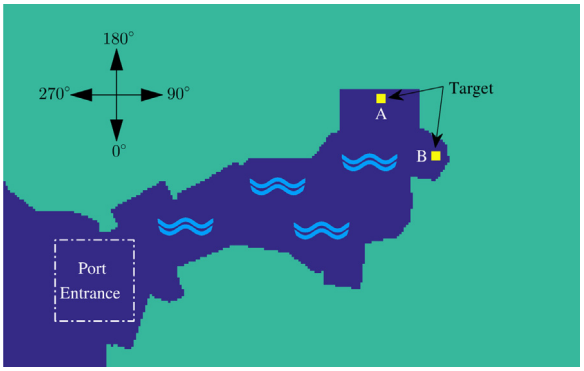


Fig. 4. A map of the Piraeus port used as the test environment in our experiments.

factor $0 \leq l \leq 1$, which represents how accurately we calculate the environmental disturbances based on the model parameters, and the measurements of the wind speed and direction.

As stated there, in the proposed RL agent we have considered that the state space consists of the platform inertial coordinates, (x, y). However, including the orientation ψ of the body-fixed frame to the set of state features in our RL-based framework constitutes a subject for future research study.

5. Simulation results

We have studied the performance of the proposed scheme using several simulated experiments. The simulation environment has been implemented using the MATLAB software package. The environment includes the kinematic and dynamic model of the marine platform, and simulated wind, wave and sea current disturbances, Moreover, the dynamics and limits of the actuators are also implemented into the simulation environment.

During the simulation runs we have used a Map of the port of Piraeus, as taken by the Google maps. Fig. 4 shows this test environment as a binary image, where the darker area denotes the sea. In this task, the objective of the marine vehicle is to find a steady landmark (rectangular target) at a minimum number of steps, starting from the entrance of the port (a wide box area). The map was completely unknown to the platform and the study was focused on the proposed method's ability to generate a physically realizable path at a reasonable computational cost under its motion constraints and the external disturbances.

Several experiments were conducted in this simulated environment considering a variety of wind conditions, in terms of the following variables:

- wind velocity direction: three values {90°, 180°, 270°}
- wind velocity magnitude: three values {1, 4, 7} kN.

In all cases the integration time step was set to $dt = 0.2$ s, which is the same as the GPS output delay (GPS frequency equal to 5 Hz). It must be noted that the RL learning step, t, is fixed equal to 10 s, i.e. fifty times the default GPS output delay. Table 1 presents the system data and parameter values used during the simulation runs. At the learning process a new episode starts when (i) the maximum allowed number of learning steps per episode is expired (in our case was set to 1000), (ii) an obstacle is hit, or (iii) the target is reached. The reward function follows the next rule: At each learning step, t, the marine vehicle receives an immediate reward of -0.1. In the case of failure, i.e. an obstacle is hit or being out of grid, the received reward is -30, and when the target is found a reward of $+15$ is returned. Finally, in all cases we have used a number of $k = 100$ RBF kernel functions following the tile

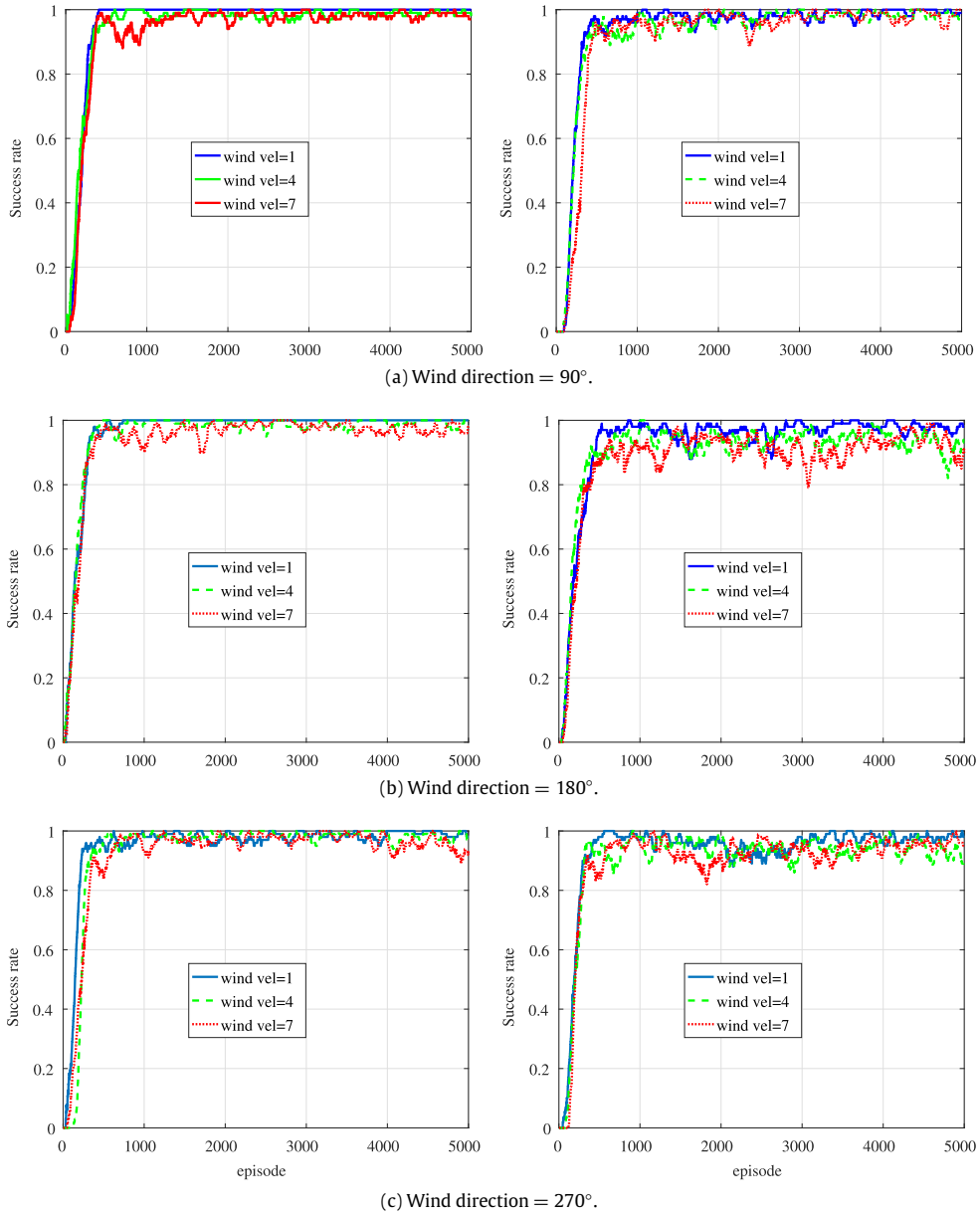


Fig. 5. The frequency of target found per episode in the case of the test environment with wind direction equals to (a) 90°, (b) 180°, (c) 270°, and variation of wind velocity.

coding scheme described above, while the discount factor γ was set equal to 0.99.

Fig. 5 illustrates the simulation results of the proposed approach for different environmental cases. In particular, we present the learning curve of the RL agent in a variety of environmental conditions, concerning wind velocity direction and velocity magnitude, see Table 1. Every curve in Fig. 5 shows the relative frequency of reaching the target in the last 100 episodes (mean success rate) in the case of two different target positions, A and B, shown in Fig. 4: target A (left column in Fig. 5) and target B (right column in Fig. 5). According to these results, the capability of successfully discovering targets is apparent. Even in difficult environmental conditions: opposite wind direction (270°) and large wind velocity (7 kN) or wind direction 0° in the case of the target A (left column in Fig. 5), the agent showed a remarkable behavior. On average, the probability of reaching the target was constantly above 85% after a short period of learning (250 episodes).

An advantage of the proposed scheme is that the application of the low-level closed-loop controller enables the realization of the action selected by the RL framework, despite the presence of disturbance forces/torque and actuation limits. Various successful navigation paths are illustrated in Fig. 6 in the case of wind direction 180° and 0°. Furthermore, we present in Fig. 7 the learned policies (Q-values) of the agent, for the same navigation paths, as obtained after 1000 episodes. As it can be observed, the proposed method successfully found sub-optimum policies in both test environments.

The performance of the low-level controller is depicted in Fig. 8. The left column shows the forces exerted from the jets during a successful run under various wind velocities. The exerted forces are bounded from the maximum allowed force limit, which is equal to 20 kN for each jet. The right column shows the desired (red dashed lines), and simulated (black solid lines) linear inertial platform velocities, and platform orientation during the same successful run under various wind velocities. In all cases the wind direction is equal to 180°. As shown, the velocities, and the orientation of the platform successfully follow the desired values despite the disturbances, measurement noise, and actuators constraints, indicating

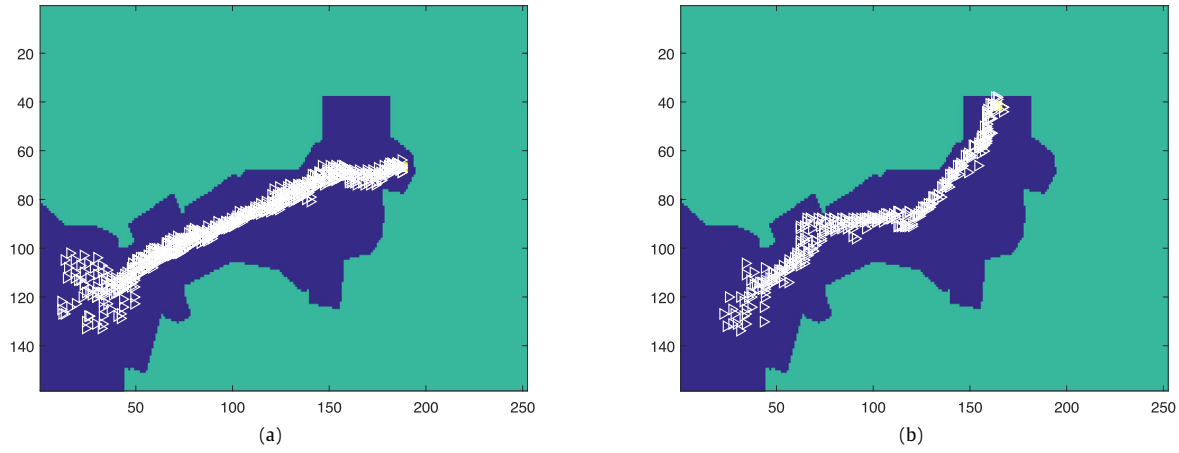


Fig. 6. The navigation path in the case of the (a) first, and the (b) second test environment.

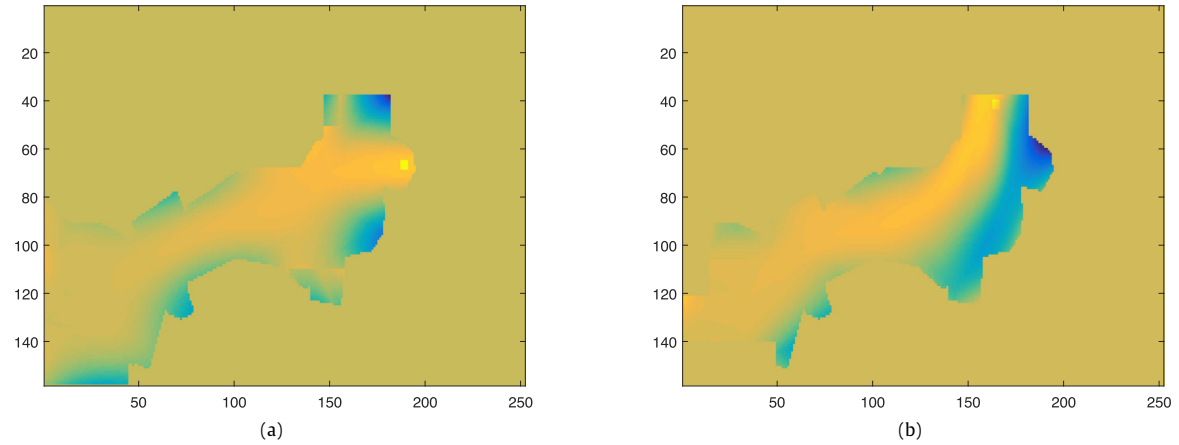


Fig. 7. The Q-function in the case of the (a) first, and the (b) second test environment.

Table 1
Parameter values during simulation.

Parameter	Symbol	Value
Total mass of the structure	m	425,000.00 kg
Length of the triangular structure side	L_{AB}	45.00 m
Length of the triangular structure base	L_{BC}	35.00 m
Integration time step	dt	0.2 s
Learning step	t	10 s
error factor	l	0.8
Wind velocity direction	θ_w	{90, 180, 270}°
Wind velocity magnitude	v_w	{1, 4, 7} kN
Platform's desired velocity magnitude	v	0.6 m/s
Velocity controller P gain	$K_{p,f}$	500,000
Orientation controller P gain	$K_{p,m}$	100
Orientation controller D gain	$K_{d,m}$	10,000
Max steps per episode	–	1000
Number of RBF kernel functions	k	100
Discount factor	γ	0.99
Found target reward	–	+15
Step reward	–	−0.1
Hit obstacle reward	–	−30

the disturbance cancellation capabilities of the proposed control scheme. Finally, Fig. 9 shows the mean value of the velocity, and the orientation of the platform during the last 100 episodes in the case of the first test environment (Fig. 6(a)) with wind direction equals to 90°, 0°, and 270°. The desired values are indicated by the red lines. Again, as shown in Fig. 9, the platform, under the control of the low-level controller, obtain the desired velocity and orientation values, despite the disturbances, measurement noise, and actuators constraints.

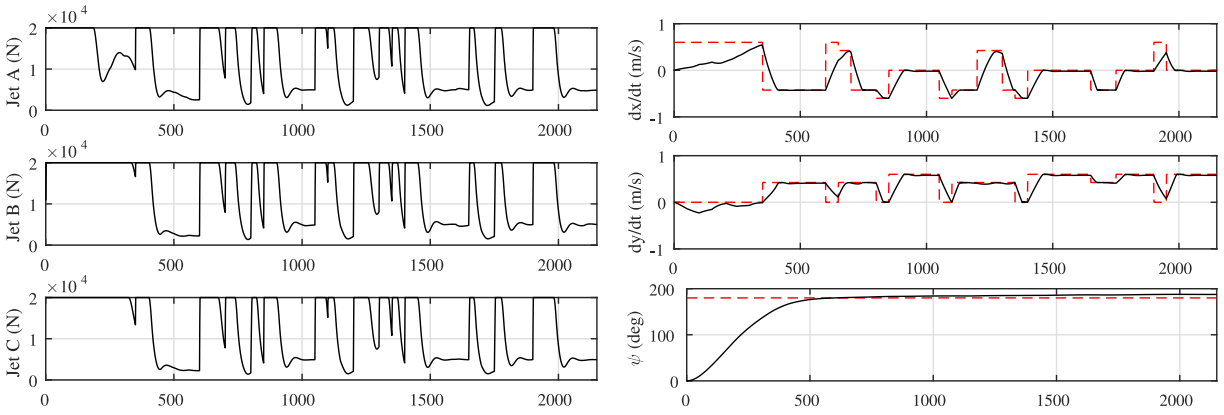
5.1. Simulation results in cases of system/sensor failure

In order to test the robustness of the proposed control scheme in cases of sensor and system failure, we implemented two more simulated scenarios. In the first, we assume that one of the jets of the platform is failing during the navigation, due to mechanical malfunction or fuel limitations. In this case, the platform remains an over-actuated system with four control inputs instead of six, and the control allocation scheme described in (2) and (3) is adapted to the new configuration, depending on which jet is failing. For example, in the case of the failure of jet C the control forces acting on the CM of the platform, and the torque about the vertical axis are generated according to

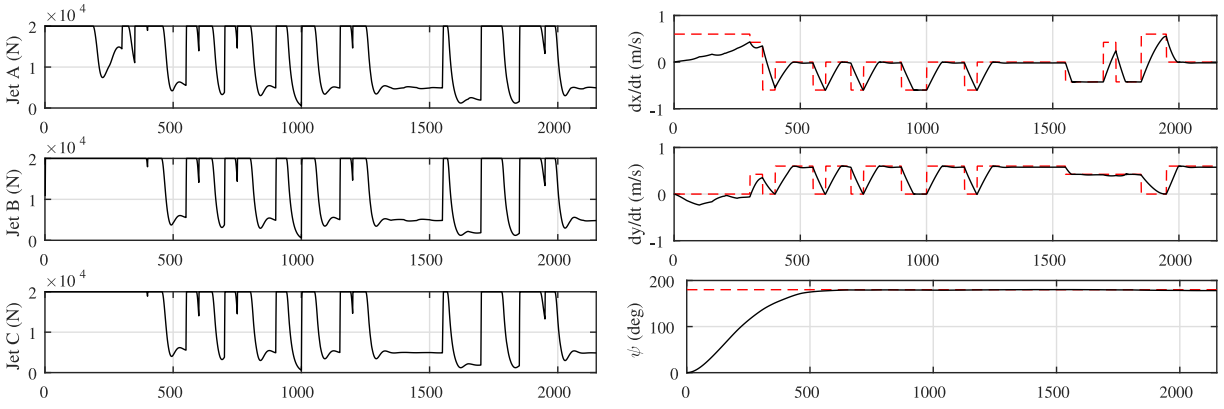
$$^{B}\mathbf{q}_c = [F_x, F_y, M_z]^\top = \mathbf{B}\,^{B}\mathbf{f}_c \tag{19}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & -d_{AG} \\ 1 & 0 & -d_{DC} \\ 0 & -1 & d_{DG} \end{bmatrix}^\top, \quad ^{B}\mathbf{f}_c = \begin{bmatrix} J_A \sin\phi_A \\ J_A \cos\phi_A \\ J_B \sin\phi_B \\ J_B \cos\phi_B \end{bmatrix}. \tag{20}$$

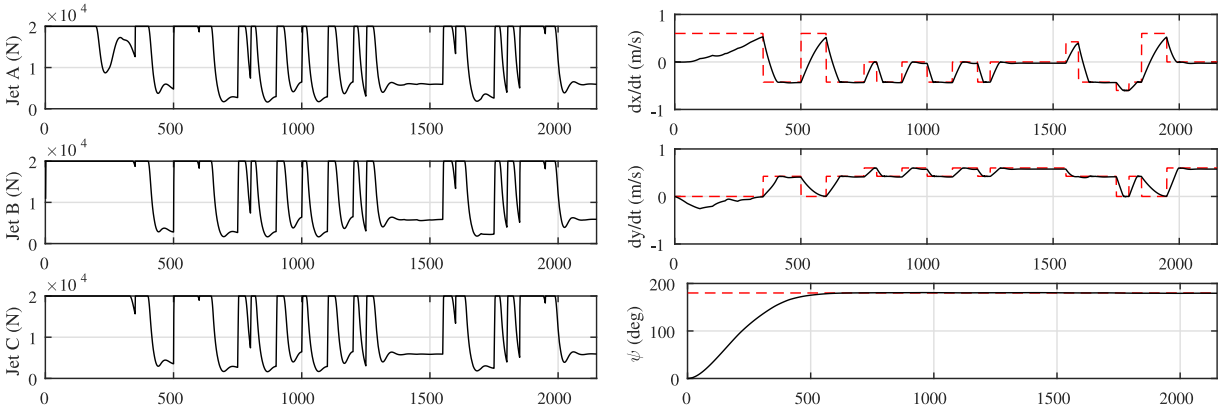
In this example, the performance of the low-level controller is depicted in Fig. 10. The left figure shows the forces exerted from the jets during a successful run, and the right figure shows the desired



(a) Wind velocity = 1 kN.



(b) Wind velocity = 4 kN.



(c) Wind velocity = 7 kN.

Fig. 8. (left) forces exerted from each jet, and (right) desired, red dashed line, and simulated, black solid line, linear inertial platform velocities, and platform orientation during a successful run under various wind velocities. In all cases the wind direction is equal to 180°.

(red dashed lines), and simulated (black solid lines) linear inertial platform velocities, and platform orientation.

Apparently, the force exerted from each jet is now higher (compare to Fig. 8), and this is to be expected, since the 1/3 of the available thrust is now missing. However, the platform successfully follows the desired velocity as commanded by the RL-agent, and the desired orientation. We must note that the failing of one engine, and consequently the loss of the 1/3 of the available thrust, limits the ability of the platform to compensate for high environmental disturbance forces. In case of the failure of two engines the system in under-actuated and its control is beyond the scope of this paper.

In the second scenario, we assume that the GPS sensor is temporarily failing, and cannot feed the control system with the state of the system in the default frequency (5 Hz). We compensate for this problem by taking under consideration the delay of the GPS sensor output into the RL-agent algorithm. For example, if the delay of the GPS sensor output is five times the default (1 s), then the RL-agent algorithm is modified on-the-fly to keep the learning step, t, equal to 10 s, i.e. the learning step is now ten times the GPS output delay. Assuming a reasonable GPS output delay (no more than 5 s), we implemented several simulation runs with various GPS output delays, and the results are virtually the same as in Figs. 5–8.

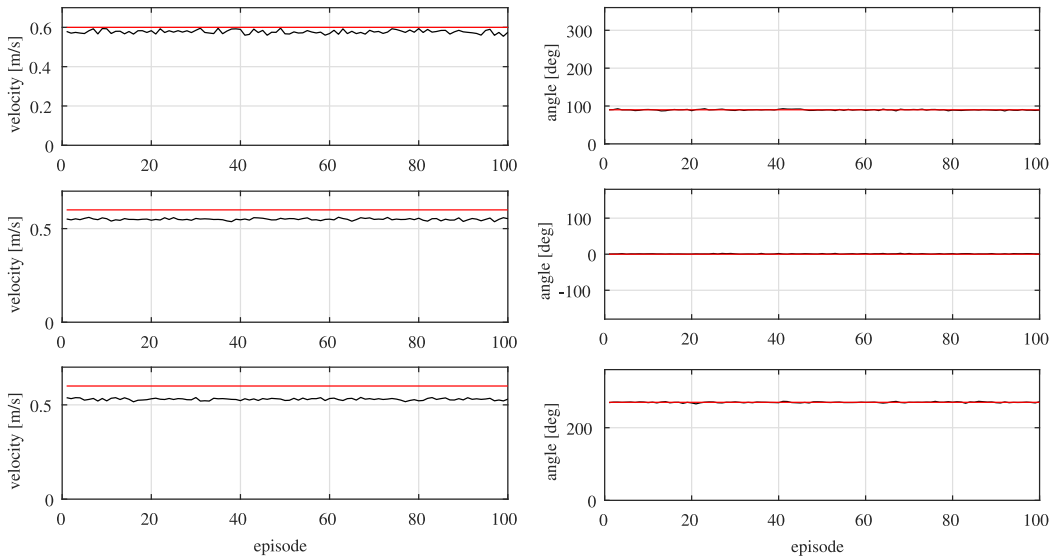


Fig. 9. The mean value of (left) the velocity and (right) the orientation of the platform during the last 100 episodes in the case of the first test environment (Fig. 6(a)) with wind direction equals to 90°, 0°, and 270°.

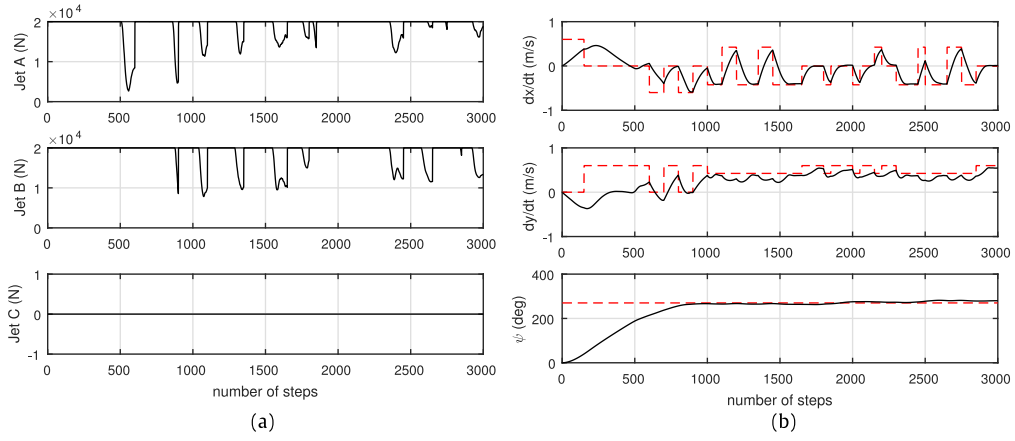


Fig. 10. (a) forces exerted from each jet, and (b) desired, red dashed line, and simulated, black solid line, linear inertial platform velocities, and platform orientation during a successful run, with only two jets functioning, for wind velocity equal to 1 kN, and wind direction equal to 270°.

6. Conclusion

In this study we presented an autonomous navigation framework of a floating marine platform involving a reinforcement learning agent, and a low-level velocity controller. The marine platform under control is over-actuated and allows efficient learning mechanisms. It consists of a triangular structure mounted on three hollow double-cylinders, one at each corner of the structure. The platform actuation is realized using three fully submerged pump-jets, located at the bottom of each cylinder. Each jet can rotate, providing vectored thrust.

The proposed RL agent tries to learn the policy function and estimate a target position according to a least-squares mechanism that uses RBF kernel functions. It receives the inertial coordinates of the platform, and makes an action related to the direction of the platform's velocity. The low-level velocity controller (a) ensures that the desired velocity commanded by the RL agent, is realized fast and efficiently, and (b) controls the orientation of the platform so that it coincides with the direction of the wind resulting to reduced disturbance forces/torque. The controller compensates for the environmental disturbances, measurement noise, and actuation delays and limitations. In addition, provide suitable solutions to GPS sensor temporary failure, and jet failure due to mechanical malfunction or fuel limitations.

Simulation results were presented to demonstrate the performance of the proposed autonomous navigation framework. The simulation environment includes the dynamic model of the marine platform, the dynamics and limitations of the actuation system, a broad model of environmental disturbances, and real GPS measurement noise. According to the results, the system shows a satisfactory behavior.

Our primitive aim is to address the marine vehicle navigation problem under two aspects: It is true that the number and the structure of the basis functions constitutes a significant issue to the performance of the linear model for the value function approximation scheme. According to the experiments, the proposed RL scheme exhibits small sensitivity to the number of basis functions. Even when the number of them was equal to 25, the method gave a satisfactory performance in the simulation environment. However, the proper determination and adaptation of basis functions suggests a future direction in our study. In addition, the application of a model-based low-level controller to further improve the closed-loop response of the system, and the possibility to model the marine navigation task with partially observable Markov decision processes (POMDPs), are also issues of future directions in our study.

The reinforcement learning scheme we followed in our approach is model-free, i.e. the optimal policy is derived without explicitly learning the model and without having access to the transition model of the MDP. In general, the RL has the ability to find the optimal solution to such a stochastic MDP by averaging over sufficient experiences. However, it is possible the learning process of an RL agent to be affected by the presence of disturbances, especially large and infrequent ones which could be considered as outliers. The disturbances can occur in sensor readings, timing or in the dynamics of the system or its environment. The experimental study in our case shows that the RL agent and its learning process are robust enough against any kind of outliers. However, it is true that in real-time dynamic systems a special treatment for disturbance rejection properties should be applied, by including functions such as: detection, rejection and correction of outliers, [21]. This study could be an excellent direction for future work dealing with this platform in more realistic environments.

Moreover, an interesting research topic that we are currently working on, is the RL-based autonomous navigation of the marine platform, taking under consideration the energy consumption of the diesel engines, aiming at its reduction. Finally, the implementation of experimental trials in a real environment with the real platform is always our goal, but unfortunately and despite our intentions, the platform is still under construction, so we are not able in the foreseeable future to proceed with experiments in a marine environment.

References

[1] M. Seto, Marine Robot Autonomy, Springer, New York, 2013.
[2] G. Antonelli, Underwater robots, in: Springer Tracts in Advanced Robotics, Vol. 96, Springer International Publishing, 2014.
[3] R.S. Sutton, Generalization in reinforcement learning: Successful examples using sparse coarse coding, in: Advances in Neural Information Processing Systems, Vol. 8, MIT Press, 1996, pp. 1038–1044.
[4] R.S. Sutton, A.G. Barto, Introduction to Reinforcement Learning, first ed., MIT Press, Cambridge, MA, USA, 1998.
[5] J.N. Tsitsiklis, Asynchronous stochastic approximation and q-learning, Mach. Learn. 16 (3) (1994) 185–202. http://dx.doi.org/10.1023/A:1022689125041.
[6] Y. Engel, S. Mannor, R. Meir, Reinforcement learning with gaussian processes, in: Proceedings of the 22nd International Conference on Machine Learning, ICML '05, ACM, New York, NY, USA, 2005, pp. 201–208. http://doi.acm.org/10.1145/1102351.1102377.
[7] N. Tziortziotis, K. Blekas, A model based reinforcement learning approach using on-line clustering, in: 2012 IEEE 24th International Conference on Tools with Artificial Intelligence, Vol. 1, Nov. 2012, pp. 712–718.
[8] N. Tziortziotis, C. Dimitrakakis, K. Blekas, Linear Bayesian reinforcement learning. in: 23rd International Joint Conference on Artififical Intelligence, IJCAI 2013, 2013, pp. 1721–1728.
[9] N. Tziortziotis, C. Dimitrakakis, K. Blekas, Cover tree Bayesian reinforcement learning, J. Mach. Learn. Res. 15 (1) (2014) 2313–2335.
[10] M. Carreras, J. Yuh, J. Batlle, P. Ridao, A behavior-based scheme using reinforcement learning for autonomous underwater vehicles, IEEE J. Ocean. Eng. 30 (2) (2005) 416–427.
[11] H. Kawano, Method for applying reinforcement learning to motion planning and control of under-actuated underwater vehicle in unknown non-uniform sea flow, in: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Aug. 2005, pp. 996–1002.
[12] B. Yoo, J. Kim, Path optimization for marine vehicles in ocean currents using reinforcement learning, J. Mar. Sci. Technol. 21 (2) (2016) 334–343. http://dx.doi.org/10.1007/s00773-015-0355-9.
[13] K. Vlachos, E. Papadopoulos, Modeling and control of a novel over-actuated marine floating platform, Ocean Eng. 98 (2015) 10–22. http://www.sciencedirect.com/science/article/pii/S0029801815000207.
[14] A. Tsopelakos, K. Vlachos, E. Papadopoulos, Backstepping control with energy reduction for an over-actuated marine platform, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), May 2015, pp. 553–558.
[15] M.G. Lagoudakis, R. Parr, Least-squares policy iteration, J. Mach. Learn. Res. 4 (2003) 1107–1149. http://dl.acm.org/citation.cfm?id=945365.964290.
[16] L. Busoniu, D. Ernst, B.D. Schutter, R. Babuska, On-line least-squares policy iteration for reinforcement learning control, in: Proceedings of the 2010 American Control Conference, June 2010, pp. 486–491.
[17] K. Tziortziotis, N. Tziortziotis, K. Vlachos, K. Blekas, Autonomous navigation of an over-actuated marine platform using reinforcement learning, in: Proceedings of the 9th Hellenic Conference on Artificial Intelligence, SETN '16, ACM, New York, NY, USA, 2016, pp. 11:1–11:7. http://doi.acm.org/10.1145/2903220.2903254.
[18] K. Tziortziotis, K. Vlachos, K. Blekas, Reinforcement learning-based motion planning of a triangular floating platform under environmental disturbances, in: 2016 24th Mediterranean Conference on Control and Automation (MED), June 2016, pp. 1014–1019.
[19] S.F. Hoerner, Fluid-Dynamic Drag: Practical Information on Aerodynamic Drag and Hydrodynamic Resistance, Hoerner Fluid Dynamics, 1965.
[20] T.I. Fossen, GUidance and Control of Ocean Vehicles, John Wiley & Sons Inc, 1994.
[21] E. Schuitema, W. Caarls, M. Wisse, P. Jonker, R. Babuska, The effects of large disturbances on on-line reinforcement learning for a walking robot, in: Proc. 22nd Benelux Conference on Artificial Intelligence, 2010.

Konstantinos Blekas received the Diploma degree in Electrical Engineering in 1993 and the Ph.D. degree in Electrical and Computer Engineering in 1997, both from the National Technical University of Athens. He is currently on the faculty of the Department of Computer Science & Engineering, University of Ioannina, Greece. He has co-authored more than 60 refereed journal and conference articles. His research interests include machine learning, statistical pattern recognition and reinforcement learning with applications to robotics and autonomous systems, computer vision, and medicine.

Kostas Vlachos received the B.Sc. degree in electrical engineering from the Technical University of Dresden, Dresden, Germany, in 1993. He received from the National Technical University of Athens, Athens, Greece, the M.S. (2000) and Ph.D. (2004) degrees in the area of Automatic Control and Robotics respectively. From 1996 to 1998, he worked as a Software Analyst in INTRACOM S.A. From 2007 to 2013, Dr. Vlachos was a visiting Lecturer at the Mechanical Engineering Department, University of Thessaly, where he taught courses in the areas of Control Systems, and Robotics. Currently, he is an Assistant Professor with the Department of Computer Science and Engineering, University of Ioannina. His research interests include control of robotic mechanisms, microrobotics, autonomous navigation, haptic mechanisms, and medical robotic simulators.