Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-29 (2018) 329–334

Straight-Path Following for Underactuated Marine Vessels using Deep Reinforcement Learning

Andreas B. Martinsen * **Anastasios M. Lekkas** *

* *Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, NO-7491 Norway (e-mail: andreabm@stud.ntnu.no, anastasios.lekkas@ntnu.no)*

Abstract: We propose a new framework, based on reinforcement learning, for solving the straight-path following problem for underactuated marine vessels under the influence of unknown ocean current. A dynamic model from the Marine Systems Simulator is employed to simulate the motion of a mariner-class vessel, however the policy search algorithm has no prior knowledge of the system it is assigned to control. A deep neural network is used as function approximator and the *deep deterministic policy gradients* method is employed to extract a suitable policy that minimizes the cross-track error. Two intuitive reward functions, which in addition prevent noisy rudder behavior, are proposed and compared. The simulation results demonstrate excellent performance, also in comparison with the line-of-sight guidance law.

Keywords: Deep reinforcement learning, path following, marine control systems, deep deterministic policy gradients

1. INTRODUCTION

The path following and tracking problems for underactuated marine vehicles have attracted the attention of the marine control research community for many years, resulting in a vast literature. The main task is to develop heading and speed control laws to follow or track a predefined path with minimum position error. More specifically, "following" pertains to the case where no temporal constraints are imposed, whereas "tracking" implies the vessel should be at a specific position each time instant. In most cases, previous works utilized existing or newly-presented models to represent the vessel dynamics and kinematics before employing methods from nonlinear control theory for developing suitable kinematic (i.e. guidance) and dynamic (i.e. control) laws for achieving the control objective.

The guidance and control systems are often treated as a cascaded system, where guidance is the driving (or perturbing) system, and control is the driven (or perturbed) system that takes longer time to converge. This structure helps simplify the stability analysis and has been used extensively in the past, see for instance the works by Lapierre et al. (2003); Fredriksen and Pettersen (2006); Lekkas and Fossen (2014a). In addition, unknown environmental forces need to be taken into account (Caharija et al., 2016; Aguiar and Hespanha, 2007; Lekkas and Fossen, 2014b; Moe et al., 2016), often through augmentation of the original line-of-sight guidance law (Fossen et al., 2003; Skjetne et al., 2011). To summarise, path following is a challenging problem due to its highly nonlinear nature and the large model and environment uncertainties involved.

The Artificial Intelligence (AI) community has developed its own theory for optimal system performance under uncertain conditions, which is based on evaluative, rather than instructive, feedback. It is most frequently referred to as *reinforcement learning (RL)* but also known as *neuro-dynamic programming* and *approximate dynamic programming* (Sutton and Barto, 1998; Bertsekas, 2012; Bertsekas and Tsitsiklis, 1996). Reinforcement learning comes in different forms, which might, or not, include partial knowledge of the environment or the system. The most important element is the *reward function* which is directly related to the objective and dictates which actions are good and receive a reward, and which are undesired and receive a penalty. The algorithm then attempts to explore the space of possible solutions until an appropriate *policy* (series of control actions) that achieves the control objective in the best possible way is found. In the past, a challenge for RL has been to keep track of everything the algorithm learns about the system. Recently, Deep Mind proposed a breakthrough solution that involves the use of Deep Neural Networks (DNNs) for representation purposes, hence resulting in the field of Deep Reinforcement Learning (DRL). Significant results include Mnih et al. (2013); Silver et al. (2017) for problems with discrete state and action spaces, and for continuous state and action spaces, algorithms such as Lillicrap et al. (2015); Mnih et al. (2016); Levine et al. (2016) have been shown to work for a number of complex problems such as robotic manipulation, bipedal locomotion, and game play.

The main contribution of this paper is a DRL-based framework for solving the straight-path following problem for underactuated marine vehicles exposed to unknown ocean currents. A mariner-class vessel model from the MSS toolbox by Fossen and Perez (2004) is employed for simulations, but the DRL algorithm does not have available any prior knowledge about the the vessel it is

2405-8963 © 2018, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.09.502

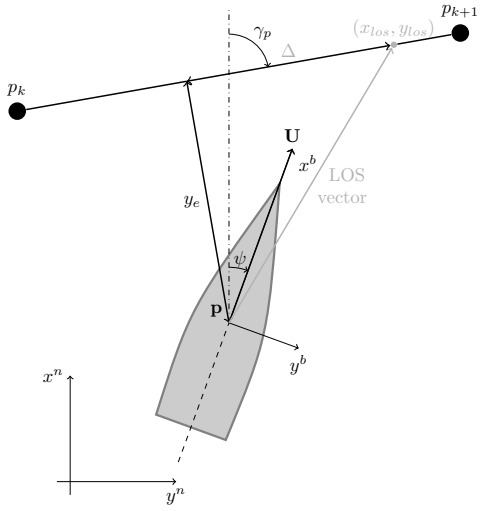

Fig. 1. Geometry of the path-following problem. The faded fonts pertain to LOS guidance, which is used for comparison.

assigned to control. Two reward functions for minimizing the cross-track error are proposed and, based on them, the Deep Deterministic Policy Gradient (DDPG) method is implemented for finding the desired policy (rudder actions). We show how to perform training to compensate for ocean currents, avoid noisy behavior of the actuators (in this case the rudder), and eliminate steady-state errors. The proposed approach constitutes an alternative solution to the path-following problem by combining the idea of optimizing performance under uncertainty, with the benefits of being model-free.

The paper is organized into five sections. Section 2 gives a brief overview of the path following problem, deep reinforcement learning and the vessel model used for simulations. Section 3 describes the implementation. Section 4 summarizes the main results, and finally, Section 5 concludes the paper.

2. PRELIMINARIES

2.1 Path following

Path following is a motion control scenario where the vehicle must converge to a predefined trajectory without temporal constraints. The vehicle can be assumed to have a constant total speed and, consequently, suitable heading angles must be reached to achieve the control objective.

Consider a straight-line path defined by two waypoints, $\boldsymbol{p_k} = [x_k, y_k]^\top$ and $\boldsymbol{p_{k+1}} = [x_{k+1}, y_{k+1}]^\top$, defined in the the North-East-Down (NED) coordinate system. The angle of the path defined by the waypoints can be computed by:

$$\gamma_p = \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k). \tag{1}$$

Using this we can find the cross-track error normal to the path using the following equation.

$$y_e = -(x - x_k)\sin(\gamma_p) + (y - y_k)\cos(\gamma_p). \tag{2}$$

The path-centered coordinate frame, from which the cross-track error is computed, is illustrated in Fig. 1.

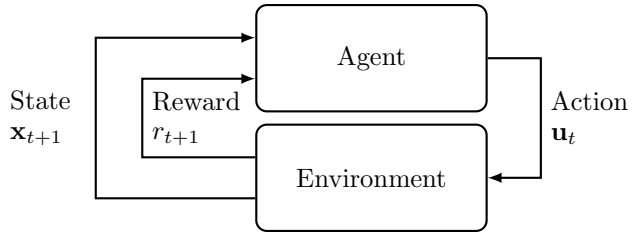

Fig. 2. Visualization of the agent's interactions with the environment. Reinforcement learning is based on evaluative feedback.

For the path following task, the objective is for the vessel to converge to the desired path, which means the cross-track error must converge to zero: $\lim_{t\to\infty} y_e(t) = 0$.

2.2 The vessel model

The motion of a surface vessel can be represented by the pose vector $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times \mathbb{S}$, and velocity vector $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$. Here, (x, y) describe the Cartesian position in the earth-fixed reference frame, ψ is yaw angle, (u, v) is the body fixed linear velocities, and r is the yaw rate. From Fossen (2011) we can describe a 3-DOF vessel model:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu}, \tag{3}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \tag{4}$$

where $\boldsymbol{M} \in \mathbb{R}^{3\times3}$, $\boldsymbol{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, $\boldsymbol{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, $\boldsymbol{\tau}$ and $\boldsymbol{R}(\psi) \in SO(3)$ are the inertia matrix, Coriolis matrix, dampening matrix, control input vector, and rotation matrix respectively. With the addition of current the relative velocity vector $\boldsymbol{v}_r = \boldsymbol{v} - \boldsymbol{v}_c$ is used, where $\boldsymbol{v}_c$ is the current velocity in the body frame.

The model and vessel parameters used in the simulation were taken from the Mariner vessel in the Marine Systems Simulator (MSS) toolbox Fossen and Perez (2004), and they include actuator constraints and asymmetries. It should be noted that the vessel model was treated as a black box, and only used for simulations during training and verification of the policy-search (control) algorithm.

2.3 Reinforcement learning

In reinforcement learning we assume the environment behaviour can be described as a Markov Decision Process (MDP) as presented by Bertsekas (2012). An MDP consists of a set of states $\mathcal{X}$, a set of actions $\mathcal{U}$, a discrete time transition model, which describes the probability of transitioning from one state $\mathbf{x}_t$ to another state $\mathbf{x}_{t+1}$ when taking an action $\mathbf{u}_t$, and a reward function $r_t = R(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. The reinforcement learning agent acts on the environment by taking actions and observing the resulting state and reward, as can be seen in Fig. 2.

The goal of the reinforcement learning algorithm is to find the optimal policy for selecting control actions $\mathbf{u_t}^* = \pi^*(\mathbf{x_t})$ which maximizes the reward gathered over time, that is, we wish to maximize the expected discounted return:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{5}$$

where $0 \leq \gamma \leq 1$ is called the discount rate. This can be expressed in terms of an action-value function

$$Q(x, u) = \mathbb{E}\left[G_t | x_t = x, u_t = u\right] \tag{6}$$

describing the expected discounted reward when taking action u in the state x. In reinforcement learning the optimal policy π^* is found by interacting with the environment, learning from the feedback, and in this way the agent generates experience which it uses to improve the policy.

DRL is a field of machine learning which combines the ideas of deep learning, i.e. Artificial Neural Networks (ANNs), and reinforcement learning. Within the field of DRL there are two main classes of algorithms for control problems with continuous action and state spaces, namely, *actor-only* algorithms, and *actor-critic* algorithms. In actor-only algorithms, only the control policy, called the actor, is approximated by a function approximator (in this case, the neural network). In actor-critic algorithms, both the policy as well as a value function, which is called the critic and evaluates the policy, are approximated.

In this paper, we implement an actor-critic method called Deep Deterministic Policy Gradients (DDPG) by Lillicrap et al. (2015), where both the policy and action-value functions are learned using ANNs. We denote the parameterized policy and action-value functions as $\pi(\mathbf{x}; \theta_a)$, and $Q(\mathbf{x}, \mathbf{u}; \theta_c)$, respectively, where θ_a and θ_c are the vectors of the ANN parameters. Learning takes place by implementing gradient descent on parameter vectors, θ_a and θ_c, as follows:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta), \tag{7}$$

where α is the learning rate, and $J(\theta)$ is the loss function which we want to minimize. The update rules for the policy and action-value function parameterizations are given in the next section, Eqs. 11–12 respectively.

3. IMPLEMENTATION

For the path following problem, the objective is to steer the vessel in such a way that it converges to the path. The main indicator of convergence is the cross-track error y_e, which informs how far away the vessel is from the path. Based on this, we propose two reward functions: A boundary reward function, and a Gaussian reward function, see Fig. 3. The boundary reward function gives a reward of 1 when the vessel is within a certain rectangular-shaped bound b of the path, and a reward of 0 otherwise:

$$R(y_e, \tilde{\psi}) = \begin{cases} 1 \text{ if } |y_e| < b \text{ and } |\tilde{\psi}| < \dfrac{\pi}{2}, \\ 0 \text{ otherwise.} \end{cases} \tag{8}$$

It should be noted that a bound on the yaw error $\tilde{\psi}$ between the yaw angle of the vessel and the path heading is also included in order to ensure the vessel travels along the path in the correct direction. The boundary reward will maximize the cumulative discounted time that the vessel is within the bound b around the path. The Gaussian reward function is given as a Gaussian curve with an amplitude a and a standard deviation σ.

$$R(y_e, \tilde{\psi}) = \begin{cases} a e^{-\frac{y_e^2}{2\sigma}} & \text{if } |\tilde{\psi}| < \dfrac{\pi}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The Gaussian reward function loses the minimum time interpretation, however it also includes several benefits

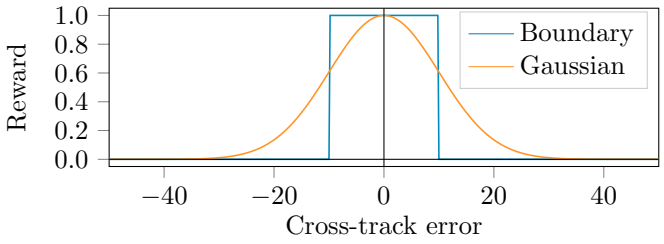

Fig. 3. Boundary and Gaussian reward functions. Rewards are given depending on the distance from the path.

compared to the boundary reward, such as better convergence properties, as the maximum reward is given when the cross-track error $y_e = 0$, and less sparse reward, which leads to faster learning.

The state representation for the learning algorithm is given in terms of the following error dynamics in a path-relative coordinate frame:

$$\mathbf{x} = [y_e, \dot{y}_e, \tilde{\psi}, \dot{\tilde{\psi}}, u, v]^\top, \tag{10}$$

where y_e is the cross-track error and $\tilde{\phi}$ is the heading error relative to the path-tangential angle γ_p. We also include the surge u and sway v velocities, in order for the learning algorithm to be able to observe all the system states, and hence be able to learn how to compensate for all the nonlinear dynamics. The action available to the algorithm is controlling the desired rudder angle δ_c, which is saturated according to the vessel model.

We implement an actor-critic method, based on Lillicrap et al. (2015), in where the policy $\pi(\mathbf{x}; \theta_\mathbf{a})$ and the action-value function $Q(\mathbf{x}, \mathbf{u}; \theta_\mathbf{c})$, are neural networks, which each consist of two fully connected layers with 400 and 300 units, and rectified linear unit activation functions.

Training was performed by simulating the environment in episodes of 1000 seconds. Each episode was randomly initialized in where the position of the vessel was chosen within a certain bound of a straight-line path. This was done in order to generate new training data for better generalization. During training, the state x_t, action u_t, reward r_t and successive state x_{t+1} were recorded and saved in a replay buffer at each time-step t.

In order to improve the policy and action-value function estimates, $\pi(\mathbf{x}; \theta_\mathbf{a})$ and $Q(\mathbf{x}, \mathbf{u}; \theta_\mathbf{c})$, stochastic gradient decent is performed on batches $\mathbb{B}$ of transitions (x_i, u_i, r_i, x_{i+1}), using the following update rules:

$$\theta_c \leftarrow \theta_c - \alpha_c \frac{1}{N} \sum_{i \in \mathbb{B}} \nabla_{\theta_c} \left(y_i - Q(x_i, u_i; \theta_c)\right)^2, \tag{11}$$

$$\theta_a \leftarrow \theta_a + \alpha_a \frac{1}{N} \sum_{i \in \mathbb{B}} \nabla_{u_i} Q(x_i, u_i; \theta_a) \nabla_{\theta_a} \pi(x_i; \theta_a), \tag{12}$$

where y_i is the action-value estimate of the state action (x_i, u_i), given as:

$$y_i = r_i + \gamma Q(x_{i+1}, \pi(x_{i+1}; \theta_{a'}); \theta_{c'}). \tag{13}$$

Additionally, soft target updates were used in order to stabilize training, where the target networks given by the parameterizations $\theta_{a'}$ and $\theta_{c'}$ slowly tracked the learned parameterizations using the following update rule at each time step:

$$\theta_{a'} = (1-\tau)\theta_{a'} + \tau\theta_a, \quad (14)$$
$$\theta_{c'} = (1-\tau)\theta_{c'} + \tau\theta_c. \quad (15)$$

During training, batches of 64 transitions were drawn randomly from a buffer of previously-observed transitions, with learning rates $\alpha_a = 1e-4$ and $\alpha_c = 1e-3$, target network update rate $\tau = 1e-3$, and discounting rate $\gamma = 0.99$. An overview of the algorithm architecture is given in Fig. 4.

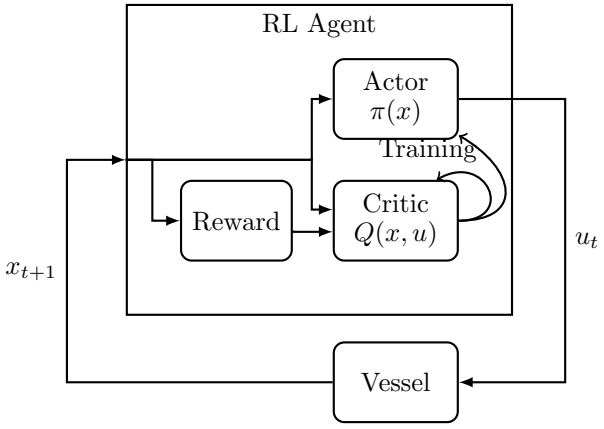

Fig. 4. DDPG architecture used in this paper. Note that the vessel model is unknown to the RL agent.

4. SIMULATION

4.1 Boundary reward

After training using the boundary reward with $b = 10m$, we get the path-following behavior seen in Fig. 5, and cross-track error seen in Fig. 6. It is observed that the algorithm has learned how to reach the desired boundary around the path and remain within it. Note that the vessel does not converge completely to the path (see Fig. 9 further below), due to the way the boundary reward function has been designed, that is, the algorithm receives the same reward for converging to the path, or navigating up to $10m$ away from it. The spikes in the cross-track error occur due to straight-line switching, which is expected.

Fig. 7 shows the actual rudder angle δ and commanded rudder angle δ_c when using the boundary reward. We observe that the policy is taking some very aggressive and undesired control actions, a common behavior in optimal control approaches (i.e. bang-bang control). To counteract this side effect, we propose adding a small penalty term, based on the derivative of the rudder movement, to the reward function. This will favour slower and smoother control actions. We therefore add the quadratic penalty term $-c_{\dot{\delta}\dot{\delta}}\dot{\delta}^2$ on the rudder derivative, which gives the reward function:

$$R(y_e, \tilde{\psi}, \dot{\delta}) = -c_{\dot{\delta}\dot{\delta}}\dot{\delta}^2 + \begin{cases} ae^{-\frac{y_e^2}{2\sigma}} & \text{if } |\tilde{\psi}| < \frac{\pi}{2}, \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

When adding the rudder derivative penalty to the reward function and retraining the policy-search algorithm, we observe a significantly smoother rudder behaviour, as seen in Fig. 8, while having little impact on the path following dynamics of the system. Note that the rudder bias observed at the end of the time series is due to vessel asymmetry.

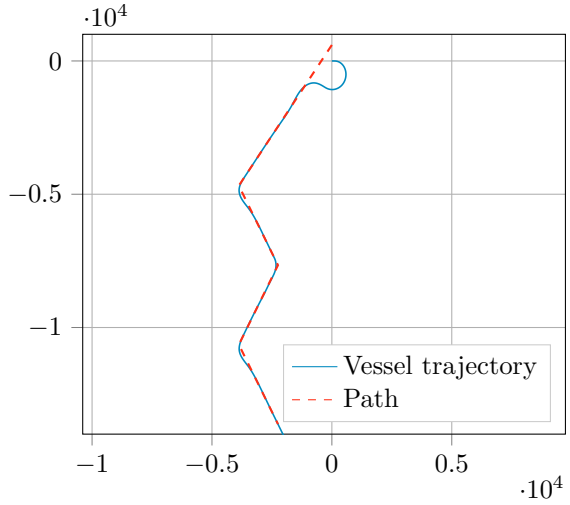

Fig. 5. Path used for all results presented, and the path following behaviour when using boundary reward.

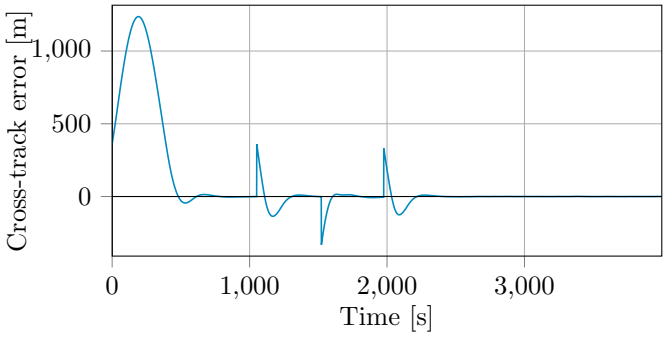

Fig. 6. Cross-track error when using boundary reward

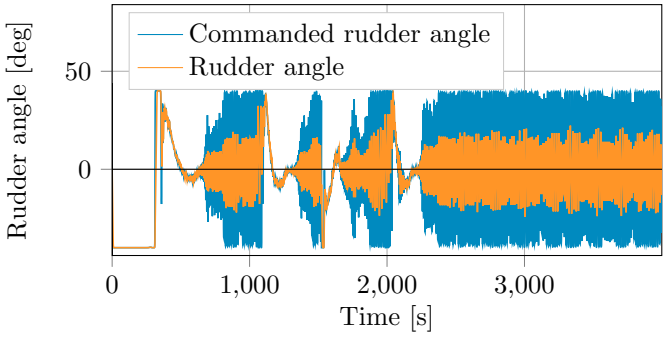

Fig. 7. Rudder angle when using boundary reward without rudder derivative penalty, resulting in aggressive and noisy control actions.

4.2 Gaussian reward

Using a Gaussian reward function with a standard deviation $\sigma = 10$ and amplitude $a = 1$, a similar path following behaviour to the algorithm trained with the boundary reward is seen. The Gaussian reward function does however have some advantages in terms of training time, as training seems to be faster. This is likely due to the reward being

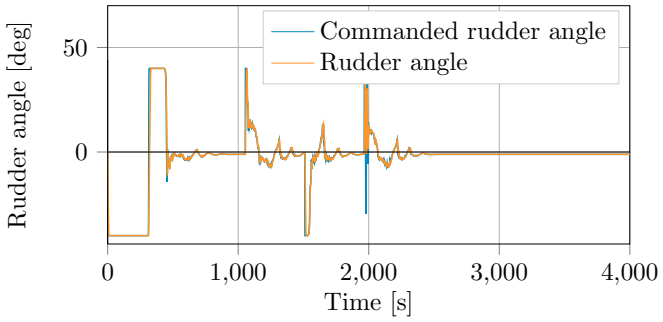

Fig. 8. Rudder angle when using boundary reward and rudder velocity penalty, resulting in smooth control actions. The rudder bias is due to vessel asymmetry.

Table 1. Reward

Reward	DRL	LOS
Boundary	622	598
Boundary + rudder penalty	619.64	596.44
Gaussian	666.37	592.54
Gaussian + rudder penalty	626.15	590.99

less sparse, which makes for more informative training data and hence faster training. The Gaussian reward also gives better path convergence than the boundary reward as can be seen in Fig. 9. This is due to the Gaussian reward having the maximum reward at $y_e = 0$, while the boundary gives the same reward within the boundary b and hence there is no incentive for the algorithm to converge to $y_e = 0$. With the Gaussian reward, aggressive and noisy control actions still occur, however this can be removed using the same rudder derivative penalty used for the boundary reward. When using the Gaussian reward we also still see that there is a steady state error, which should in theory converge to zero as the policy approaches the optimal policy. One potential remedy is to allow for additional training until the algorithm has explored more additional solutions. Another solution would be to tune the reward function to be steeper around zero. A third option is to feed the policy-search algorithm with additional states that will facilitate finding the optimal policy. In Section 4.4, though, we show that the problem can be solved by adding integral action to the cross-track error.

4.3 Comparison with LOS guidance

In order to give an indication of the performance of the learning algorithm, we simulated the LOS guidance law and the trained reinforcement learning algorithm for 1000 seconds, with the same initial conditions, and following the same straight-line path without current. Using boundary and Gaussian reward functions both with and without rudder derivative penalty, we got the cumulative rewards seen in Table 1. From the results we can see that overall the RL approach is able to accumulate more reward in similar situations. This is not unexpected, as the RL approach is trained to maximize the reward, while LOS guidance is not optimized for the particular reward function. It should be noted that the results for LOS guidance may vary depending on how it is tuned, for our purpose, a lookahead distance of three times the vessel length $583m$ was used, together with a feedback linearizing heading controller.

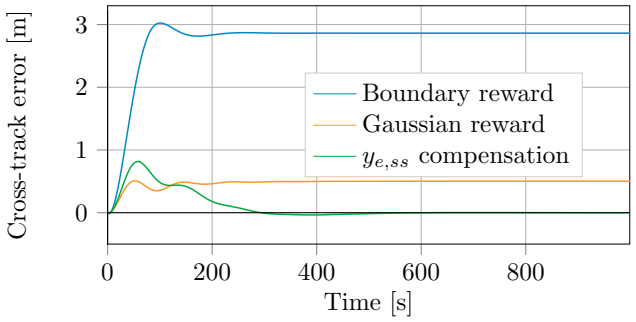

Fig. 9. y_e convergence with boundary reward, Gaussian reward, and Gaussian reward with SSE compensation.

4.4 Steady-state error (SSE)

To address the problem of SSE, we propose to first estimate it using integral action on the cross-track error, and then compensate for it by augmenting the state vector accordingly. The SSE is estimated as follows:

$$\hat{y}_{e,ss}(t) = k_i \int_0^t y_e(t)dt, \tag{17}$$

where k_i describes the rate at which the error estimate changes. Using the augmented state vector

$$\mathbf{x} = [y_e + \hat{y}_{e,ss}, \dot{y}_e, \tilde{\psi}, \dot{\tilde{\psi}}, u, v]^\top \tag{18}$$

resulted in path convergence, as can be seen in Fig. 9. It should be noted that adding the steady state compensation should only be done on a trained policy, and not performed during training, as doing so will cause interference and possibly divergence due to exploration noise. Additionally, an integration strategy with anti-windup action was used in order to reduce overshoot, and improve stability.

4.5 Compensating for ocean currents

Ocean currents stationary in the NED frame can be handled with the same architecture as described in the previous sections, the only difference being that training must be performed with current. In order to make the policy generalize to currents with different magnitudes and directions, training was performed using currents with a random magnitude between 0 and $2m/s$ and a random direction in each episode. For a current of $0.9m/s$ and angle of 45°, the resulting policy achieved the performance seen in Figs. 10–11. From the results we can clearly see that the control policy brings the vessel very close to the path. At the end of the simulation, and when the current is normal to the path, the learned policy has a steady state error of $5m$. By adding SSE compensation, we are able to further improve the performance and achieve convergence to the path, similarly (or, in this particular case, better) to that of Integral LOS guidance.

5. CONCLUSION

We have addressed the path-following problem for a mariner-class vessel using deep reinforcement learning. The DDPG algorithm was implemented to extract suitable policies so as to minimize the cross-track error, also in the presence of ocean currents. We proposed two simple

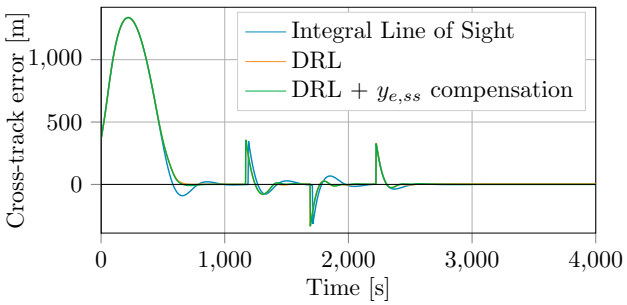

Fig. 10. Comparison of cross-track error when exposed to current.

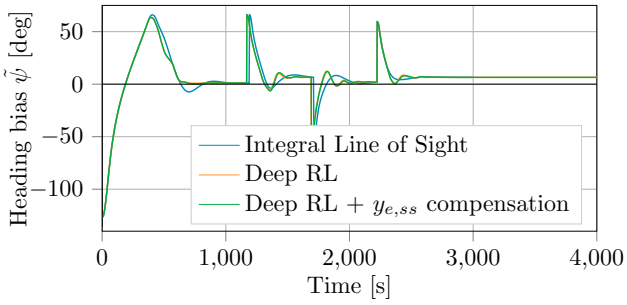

Fig. 11. Angle $\tilde{\psi}$ between the vessel's heading angle and the path-tangential angle, when exposed to ocean current.

reward functions and showed how to eliminate steady-state errors and avoid noisy rudder behavior. The method gave excellent results, and comparisons with LOS guidance were presented. One main advantage of the approach is that it does not require any beforehand knowledge of the vessel model or the environment. In fact, the algorithm was able to detect and take into account the effects of asymmetries and actuator constraints. Another advantage is the possibility to perform transfer learning and implement the same policy on different vessels with minimum additional training. Some preliminary results are presented by the authors in (Martinsen and Lekkas, 2018). On the downside, training the algorithm can be time-consuming, excessive training can lead to instabilities, and choosing the neural network size is a task that can involve substantial experimentation.

REFERENCES

Aguiar, A.P. and Hespanha, J.P. (2007). Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8), 1362–1379.

Bertsekas, D.P. (2012). *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 4th edition.

Bertsekas, D.P. and Tsitsiklis, J.N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Caharija, W., Pettersen, K.Y., Bibuli, M., Calado, P., Zereik, E., Braga, J., Gravdahl, J.T., Sørensen, A.J., Milovanović, M., and Bruzzone, G. (2016). Integral line-of-sight guidance and control of underactuated marine vehicles: Theory, simulations, and experiments. *IEEE Transactions on Control Systems Technology*, 24(5), 1623–1642.

Fossen, T.I. and Perez, T. (2004). Marine systems simulator (MSS). URL `http://www.marinecontrol.org`.

Fossen, T.I. (2011). *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons.

Fossen, T.I., Breivik, M., and Skjetne, R. (2003). Line-of-sight path following of underactuated marine craft. *IFAC Proceedings Volumes*, 36(21), 211–216.

Fredriksen, E. and Pettersen, K.Y. (2006). Global κ-exponential way-point maneuvering of ships: Theory and experiments. *Automatica*, 42(4), 677–687.

Lapierre, L., Soetanto, D., and Pascoal, A. (2003). Nonlinear path following with applications to the control of autonomous underwater vehicles. In *42nd IEEE Conference on Decision and Control*, 1256–1261.

Lekkas, A.M. and Fossen, T.I. (2014a). Integral LOS path following for curved paths based on a monotone cubic Hermite spline parametrization. *IEEE Transactions on Control Systems Technology*, 22(6), 2287–2301.

Lekkas, A.M. and Fossen, T.I. (2014b). Trajectory tracking and ocean current estimation for marine underactuated vehicles. In *IEEE Conference on Control Applications (CCA)*, 905–910.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1), 1334–1373.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Martinsen, A.B. and Lekkas, A.M. (2018). Curved-path following with deep reinforcement learning: Results from three vessel models. In *OCEANS MTS/IEEE (accepted)*.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Moe, S., Pettersen, K.Y., Fossen, T.I., and Gravdahl, J.T. (2016). Line-of-sight curved path following for underactuated USVs and AUVs in the horizontal plane under the influence of ocean currents. In *24th IEEE Mediterranean Conference on Control and Automation (MED)*, 38–45.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676), 354.

Skjetne, R., Jørgensen, U., and Teel, A.R. (2011). Line-of-sight path-following along regularly parametrized curves solved as a generic maneuvering problem. In *50th IEEE Conference on Decision and Control and European Control Conference*, 2467–2474.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement learning: An introduction*. MIT Press.