POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Learning Trajectory Tracking For An Autonomous Surface Vehicle In Urban Waterways

Laurea Magistrale in Computer Science and Engineering
Ingegneria Informatica

Author: Toma Sikora

Advisor: Prof. Riccardo Scattolini

Academic year: 2021-2022

---

1. Introduction

Autonomous surface vessels (ASVs) have become a popular solution for marine exploration, search and rescue, environmental monitoring, and hydrology surveying. With congestion becoming a serious problem on the roads, an ASV could take on the task of transportation in many cities with urban waterways, such as Amsterdam or Venice.
Roboat is an example of such a system, developed as a research project by the AMS Institute and the MIT [5]. The platform can provide numerous functions to a city, such as dynamic infrastructure or autonomous garbage collection. Although in a very mature state, the system's control strategy, a Nonlinear Model Predictive Controller (NMPC), proved sensitive to uncertainties and disturbances.
Recent advancements in the field of reinforcement learning, namely the work done by the Robotic Systems Lab in Zurich on quadrupedal locomotion in [3], gives hope that such an approach could inherently learn robustness.
The goal of this work is to develop a learning-based robust controller for trajectory tracking of an ASV. Faced with uncertainties or disturbances, the proposed controller should track trajectories with less tracking error than the NMPC.
Similar problems have been considered in the past, for example one of the most prolific authors in vessel automation, Thor I. Fossen, considered the automation of tasks such as dynamic positioning, path following, line of sight algorithms, etc. in his book [2]. Furthermore, reinforcement learning approaches to vessel control have also been tested, namely the Proximal Policy Optimization algorithm [4] was used to learn dynamic positioning in [1], proving successful both in simulation and real world tests.

2. Background

2.1. ASV kinematics and dynamics

The movement of an ASV can be expressed using two reference frames, the inertial and the vessel's own reference frame. It can be condensed in 3 degrees of freedom (DOF) in the inertial reference frame, presenting the vessel's position and orientation:

$$\boldsymbol{\eta} = [x, y, \psi] \tag{1}$$

Likewise, the linear and angular velocities of the vessel's movement can be expressed in the ves-

sel's reference frame:

$$\boldsymbol{\nu} = [u, v, r] \tag{2}$$

The two vectors are connected through the following equation:

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

where $\boldsymbol{R}(\psi)$ is the transformation matrix from the vessel's own to the inertial frame.
The most popular form of propulsion for ASVs are propellers which, according to [2], produce thrust relative to the vessel's advance speed and the propeller RPM with the following equations:

$$\boldsymbol{\tau} = \rho D^4 K_T(J_0)|\boldsymbol{u}|\boldsymbol{u} \tag{4}$$

$$J_0 = \frac{(1-\omega)\boldsymbol{\nu}}{\boldsymbol{u}D} \tag{5}$$

where ρ is the density of the fluid, D the propeller diameter, K_T the thrust coefficient estimated for a hull-propeller pairing, J_0 the advance number, defined with the equation (5), and ω the wake fraction number.
The Roboat platform is actuated through a set of four thrusters in the "+" thruster configuration, with the main thrusters on the port and starboard side of the center of mass and two sideways thruster on the bow and stern, making the system overactuated.
The most significant uncertainties and disturbances for the Roboat platform are: varying payload, wind, current, and waves. These can be modelled in the following manner.
Firstly, since the magnitude of payload is significant with regards to the vessel's own weight, varying payload induces high trajectory tracking error. The effect of the distribution and magnitude of payload is reflected in a proportional change in the vessel's mass and damping matrices and the wake fraction number ω.
Secondly, the effect of wind can be approximated with a force and torque combination acting on the vessel with:

$$\boldsymbol{\tau_w} = \frac{1}{2}\rho_a V_{rw}^2 \begin{bmatrix} -c_x A_{FW} \cos(\gamma_{rw}) \\ c_y A_{LW} \sin(\gamma_{rw}) \\ c_z A_{LW} L_{OA} \sin(2\gamma_{rw}) \end{bmatrix} \tag{6}$$

where γ_{rw} is the apparent wind angle, ρ_a the density of air, V_{rw} the apparent wind speed, A_{FW} the frontal projected windage area, A_{LW} the lateral projected windage area, L_{OA} the vessel's overall length, and ω the wind's apparent angle of attack. c_x, c_y, and c_z are the wind coefficients estimated for the vessel at hand.
Thirdly, the current acts as a constant translation of the vessel's moving frame with a certain velocity. Its effect can be described by the following equation:

$$u_c = V_c \cos(\beta - \psi) \tag{7}$$

$$v_c = V_c \cos(\beta - \psi) \tag{8}$$

where u_c and v_c present the current's velocity in vessel's surge and sway directions respectively.
Waves are not considered in this work as the low frequency waves are never encountered in the urban waterways and the effects of high frequency waves are very weak.
Finally, the equations of motion for the entire system can be written as:

$$\boldsymbol{M\dot{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D_L}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\nu D_Q}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \tag{9}$$

$$\boldsymbol{M} = \boldsymbol{M_{RB}} + \boldsymbol{M_A} \tag{10}$$

$$\boldsymbol{\tau} = \boldsymbol{\tau_u} + \boldsymbol{\tau_E} \tag{11}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi) \cdot \boldsymbol{\nu} \tag{12}$$

where $\boldsymbol{M_{RB}} + \boldsymbol{M_A}$ are the rigid body and added mass terms, $\boldsymbol{D_L}(\boldsymbol{\nu})$ the linear damping matrix, $\boldsymbol{D_Q}(\boldsymbol{\nu})$ the quadratic damping matrix, $\boldsymbol{\tau_u}$ the vector of thruster generated torques, and $\boldsymbol{\tau_E}$ the vector of environmental torques.

2.2. Reinforcement Learning

Basic reinforcement learning (RL) problems can be described through the notion of discrete-time stochastic control processes called Markov Decision Processes (MDPs). They are defined by a (S, A, P_a, R_a) tuple where:

- S is the set of possible states called the state space,
- A is the set of possible actions called the action space,
- P_a is the probability of action a taken in state s leading to state s'
- R_a is the immediate reward for taking action a in state s.

The agent's behavior, or more specifically the choice of action in any given state, is defined

through the policy function $\pi(s, a)$. A policy that maximizes the reward function in the infinite horizon is called the optimal policy $\pi^*(s, a)$. The goal of RL algorithms is to obtain this optimal policy for a given agent-environment pair. This is usually done without knowledge of the environment dynamics P_a, which is where the name model-free RL comes from.
In the early stages of RL, the optimal policy was found by using the Q-learning algorithm which, inspired by the Bellman equation, extracts it from a table of approximated infinite horizon rewards for state-action pairings. To solve more complex problems, a neural network was used in place of the table in algorithms such as Deep Q-learning. Most recently, a new family of algorithms called Policy Gradient Methods showed promise allowing also continuous state and action values. These algorithms directly optimize the agent's policy, presented with a parameter vector θ usually in the form of a neural network, by performing gradient ascent on it.
The most recent breakthrough in the family is the Proximal Policy Optimization (PPO) from [4]. Presented in 2017, PPO is simple, more general than its predecessors, and has better sample complexity. To achieve this performance, the algorithm uses two tricks: trust region methods and a clipped surrogate objective. Currently, PPO outperforms similar algorithms in a number of tasks.

3. Simulation of the Roboat and design of the RL controller

To simulate the Roboat system in code, the equations from 2.1 were translated to Python code, parameterized based on uncertainties and disturbances, and wrapped in a Gym environment, following the interface architecture from the industry standard OpenAI Gym.
The following criteria must be met to use RL algorithms in an environment: define the observation and action space, and define the *step* and *reset* methods.
For the task of trajectory tracking the observation space was chosen to contain: the x and y error in the vessel's reference frame, the sine and cosine (for continuity) of the look-ahead heading error, the current and reference velocity values, and the actuation vector for the previous timestep.
The action space was defined as four values from -1 to 1, mapped to the minimum and maximum thruster range.
The reward function was defined as:

$$r_t = \begin{cases} r_{gauss} + r_{heading} - r_u - r_{\Delta u} & \text{if } \boldsymbol{\eta_{err}} \leq 5 \\ -100 & \text{if } \boldsymbol{\eta_{err}} > 5 \end{cases} \tag{13}$$

$$r_{gauss} = \exp\left(-k_1 * ((\hat{x}_t - x_t)^2 + (\hat{y}_t - y_t)^2)\right) \tag{14}$$

$$r_{heading} = \exp\left(-k_2 * \boldsymbol{\Delta\psi_t^2}\right) \tag{15}$$

$$r_u = k_3 * \left(k_5 * (u_{1,t}^2 + u_{2,t}^2) + k_6 * (u_{3,t}^2 + u_{4,t}^2)\right) \tag{16}$$

$$r_{\Delta u} = k_4 * \sum_{i=1}^{4} |(u_{i,t} - u_{i,t-1})| \tag{17}$$

where $\boldsymbol{\Delta\psi_t}$ the difference between the current and desired heading at time t, $u_{i,t} (i = 1, 2, 3, 4)$ are the actuation values at time t, and $k_i (i = 1, ..., 6)$ constants balancing the effect of each component on the learning process. The parameter values balancing the reward function are given in Table 1.

Parameter	k_1	k_2	k_3	k_4	k_5	k_6
Value	3	3	1	0.5	0.1	0.7

Table 1: Constant values in the reward function

The RL simulator was also compared to the proprietary Roboat simulation through step response tests and validated that it performs almost identically to the proprietary one.
To create a rich set of trajectories a trajectory generation module was built to output straight line, circular, and sine wave trajectories, parameterized with speed, radius, amplitude, and period values.
The training was performed for 1 million timesteps to avoid overfitting, as at that point the algorithm stagnated. The episode reward through the training process can be observed in Figure 1 and the learning process is visualized in Figure 2 on a sine wave trajectory.

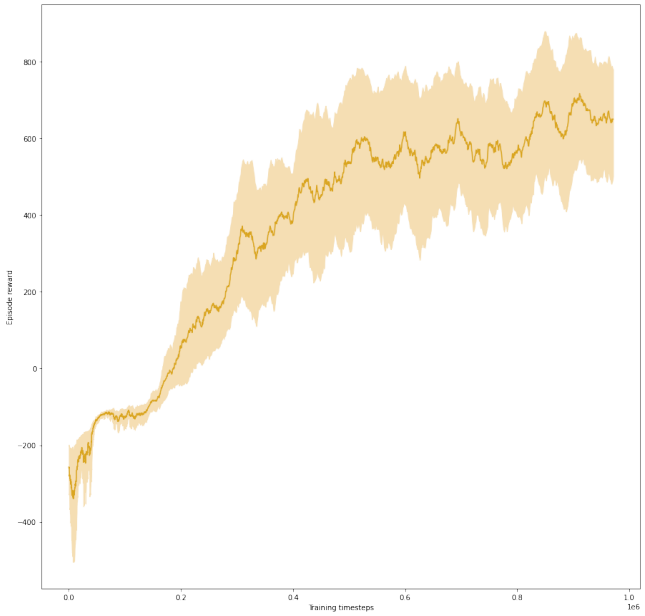

Figure 1: Plot of the episodic reward throughout one million steps of training. Orange line presents the moving average with a window size 50, light orange presents the standard deviation of the value.

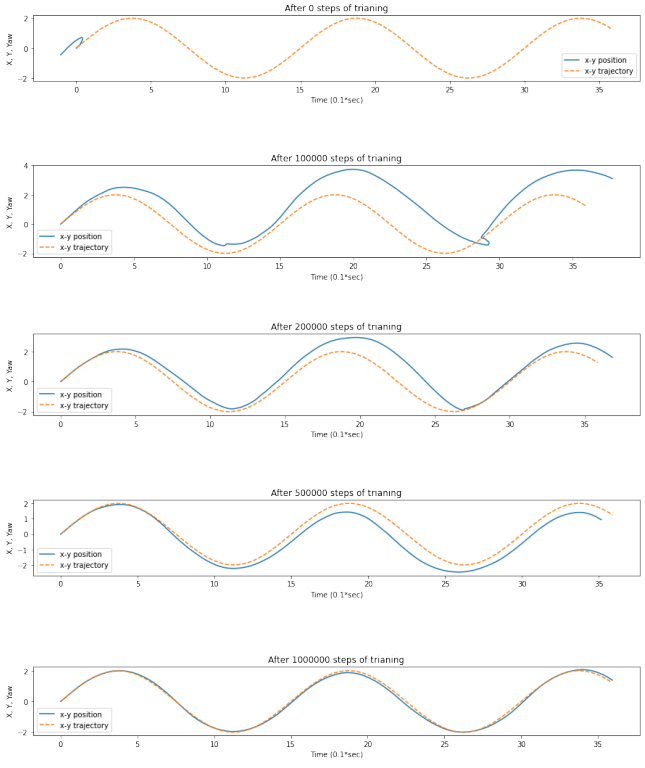

Figure 2: Visualization of intermediate trajectory tracking training results for the RL controller after initialization, 100000, 200000, 500000, and 1000000 training steps.

4. Results

Having trained a trajectory tracking controller it was time to compare it to the current control strategy. The performance of the NMPC and the RL controller was compared through two metrics. Firstly, the tracking precision was compared with the Euclidean distance between the current position and the desired position on the reference trajectory. In general, the performance was compared with the Root Mean Squared Error (RMSE) of that measure. Secondly, the average power usage of a given algorithm was measured with the Equation 18:

$$P_{average} = \sum_{i=1}^{N} \frac{(|f_1^i| + |f_2^i| + |f_3^i| + |f_4^i|) * |u^i|}{N} \tag{18}$$

where N is the number of data points, u^i is the surge speed of the vessel at time i and f_j^i is the force command for thruster j at time i.

4.1. Trajectory tracking comparison in simulation

The tests in simulation were done on a sine wave trajectory with amplitude 2 m, period 10 m, and forward speed 0.5 m/s. Figures 3 and 4 visualize the results of the comparison in simulation without the inclusion of uncertainties and disturbances. The position, the tracking error, and the force allocation are presented in the graphs. The two controllers perform similarly in terms of position error with RL being more reactive to its increase. Also, it is obvious that RL exhibits erratic and oscillatory actuation using up more power in the process.
Table 2 contains the comparison between the controllers with the inclusion of the disturbances and uncertainties.

Scenario	Undisturbed	Added payload	Current	Wind
Δ pos. NMPC (m)	0.3018	0.6979	1.6810	1.8032
Δ pos. RL (m)	0.2836	0.5836	0.5803	1.701
Difference	-6.03%	-22.83%	-65.48%	-5.69%
Δ P NMPC (W)	323.7134	559.3913	412.3100	184.7790
Δ P RL (W)	491.3406	742.1449	460.2448	300.3304
Difference	51.78%	32.67%	11.63%	62.53%

Table 2: Comparison between the NMPC and the RL algorithm in the evaluated metrics: average position error and power consumption.

The data clearly shows RL is capable of reducing the tracking error in simulation, however, this comes at a significant power cost. Furthermore, the erratic actuation hints at model exploitation.

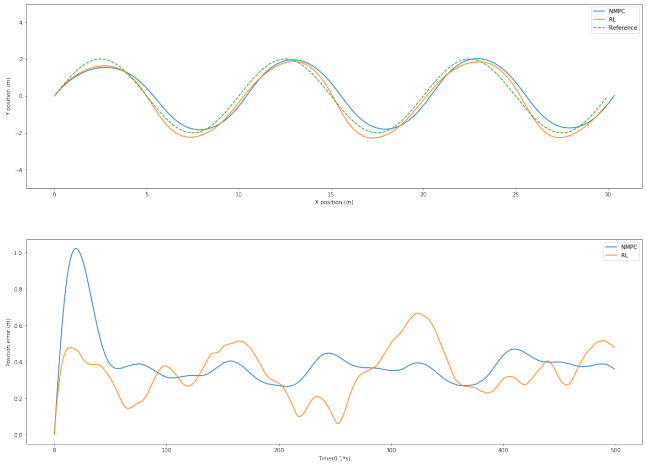

Figure 3: Comparison between the NMPC and RL algorithm trajectory tracking performance. Trajectories of NMPC, RL, and reference sine on the top graph. Tracking error of NMPC and RL on the bottom graph.

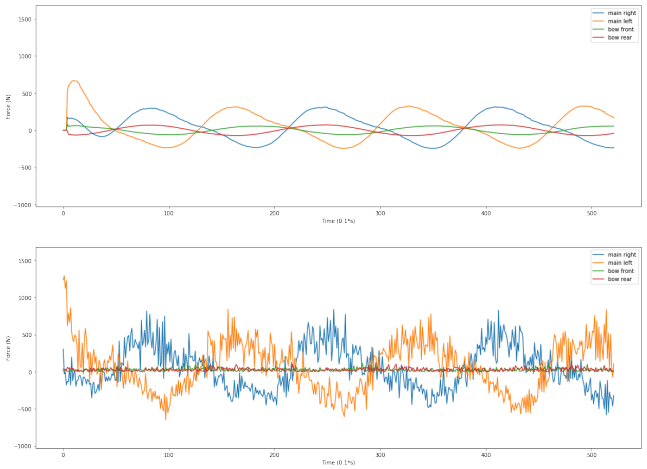

Figure 4: Comparison between the NMPC and RL algorithm trajectory tracking performance. Force allocation of NMPC (top) and RL (bottom) for tracking the sine wave trajectory.

4.2. Trajectory tracking comparison on the real system

After studying the controller in simulation, ROS framework for deployment to the real system was developed. Initially, a number of problems arose that caused failure. These problems range from unexpected software behavior, sensor noise and latency, to model discrepancies. Some of these were solved, others partially mitigated.

Firstly, to avoid badly formed trajectories, a module in the framework enforces certain structure by interpolating between points in the trajectory. Due to this, the reference trajectory is always slightly different at each timestep causing RL failure. This was fixed by overriding the said module.

Secondly, measurement noise and GPS delay caused choppy odometry signals and RL performance deterioration. The odometry is calculated with an Extended Kalman Filter from an IMU and two GPS sensors. To mitigate the problem its covariance matrix was re-tuned to obtain a smoother output.

The last and most potent problem discovered, was the discrepancy between the physical and thruster model between the real world and the simulation. Upon tedious inspection, the thruster behavior was discovered to be a compound problem formed of a 0.5s - 1s command signal delay, a proprietary RPM value PID controller, and thruster dynamics (as explained in 2.1). These effects are most obvious whenever changing speeds, which is why the sine waves are tracked much worse in real life than in simulation. Furthermore, this led to errors in parameter estimation for the physical model as reference force values do not align with true output. Approaches used to improve the model range from adding random noise, autoregressive and moving average processes, and thruster dynamic modelling. In the end, a combination of a random noise and a moving average with a window of size 5 applied to the actuation vector yielded best results. The resulting actuation was more robust and the algorithm learned the slower and unreliable nature of the actuation. Detailed modelling of the thruster from a control signal to the actual thrust output could be a very interesting study case in itself.

Figures 5 and 6 visualize the best results for RL of the comparison with two people on board amounting to around 160 kg of additional payload, facing wind of around 2 on the Beaufort scale, and no notable current or waves. Once again, the graphs present the position, the tracking error, and the force allocation.

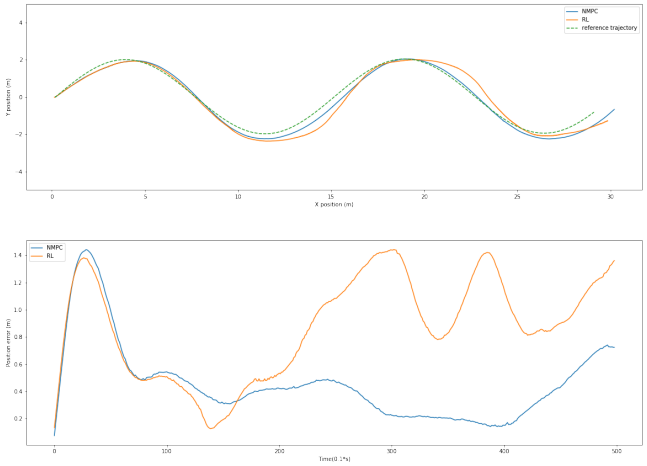

Figure 5: Comparison between the NMPC and RL algorithm trajectory tracking performance on the real Roboat vessel. Trajectories of NMPC, RL, and reference sine on the top, tracking error of NMPC and RL on the bottom.

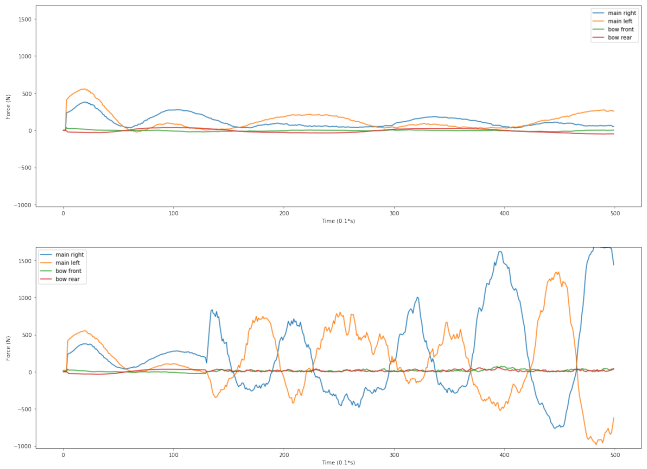

Figure 6: Comparison between the NMPC and RL algorithm trajectory tracking performance on the real Roboat vessel. Force allocation of NMPC (top) and RL (bottom).

5. Conclusions

The goal of this work was to train a learning-based controller to perform precise trajectory tracking for the Roboat platform. Not finding an adequate simulator, a new simulator of the vessel's dynamics, including the uncertainties and disturbances, has been developed to train agents through reinforcement learning algorithms. After the search for the best setup and hyperparameters, an agent was trained with the PPO algorithm in simulation to perform trajectory tracking successfully even when faced with uncertainties and disturbances. Compared to the NMPC, the controller showed better tracking performance, at a significant power consumption cost and an erratic actuation signal.
When deployed on the real system its performance deteriorated significantly proving not to be robust to differences between the real world and the simulated system, especially the thruster model. In the end, a study of problems causing the tracking to fail was performed with some of them being successfully solved and others needing further work.
Reinforcement learning still seems to be a bad fit for critical real world applications, however, recent developments hint it might become robust enough in the near future.

6. Acknowledgments

I would like to thank my supervisor, prof. Riccardo Scattolini, for leading me through this work. Furthermore, I would like to thank Jonathan and the entire AMS institute and Roboat team, for providing me with the opportunity to work on such a project. Finally, I would like to thank my family and Ivana for the endless love and support.

References

[1] Jens Balchen, Nils Jenssen, Eldar Mathisen, and Steinar Saelid. Dynamic positioning of floating vessels based on kalman filtering and optimal control. pages 852 – 864, 01 1981.

[2] T.I. Fossen. *Guidance and Control of Ocean Vehicles*. Wiley, 1994.

[3] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), oct 2020.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[5] Wei Wang, Luis Mateos, Shinkyu Park, Pietro Leoni, Banti Gheneti, Fabio Duarte, Carlo Ratti, and Daniela Rus. Design, modeling, and nonlinear model predictive tracking control of a novel autonomous surface vehicle. 05 2018.