Batch Reinforcement Learning of Feasible Trajectories in a Ship Maneuvering Simulator

José Amendola[1] **, Eduardo A.Tannuri**[1] **, Fabio G. Cozman**[1] **, Anna H. Reali**[1]

[1]Universidade de São Paulo, Escola Politécnica

{jose.amendola,eduat,fgcozman,anna.reali}@usp.br

Abstract. *Ship control in port channels is a challenging problem that has resisted automated solutions. In this paper we focus on reinforcement learning of control signals so as to steer ships in their maneuvers. The learning process uses fitted Q iteration together with a Ship Maneuvering Simulator. Domain knowledge is used to develop a compact state-space model; we show how this model and the learning process lead to ship maneuvering under difficult conditions.*

1. Introduction

Machine learning is now getting significant attention in the context of marine systems research: indeed, several projects are working on autonomous ships to be delivered in the next decade or so [Laurinen 2016, MUNIN 2016]. While automated marine navigation systems are well established for open sea navigation, the process of ship berthing has resisted automation (that is, the process of taking a ship to its final position in a port). The sea bottom and the margins interfere with vessel dynamics in complex ways that are still under research [Berg and Ringen 2011]. Furthermore, the maneuver of a vessel in restricted waters relies on the knowledge of maritime pilots regarding port and local weather conditions. Another important restriction is the loss of rudder efficiency as ship velocity slows down, so a ship cannot simply be steered in low speed along a channel.

All in all, autonomous ship berthing remains an open topic that is not effectively approached by ongoing autonomous, even large, ship projects. One difficulty in developing such autonomous navigation systems is that there are no full-scale prototypes deployed; given the high costs of real marine operation, testing is difficult. Hence simulators with accelerated time steps are often employed for engineering analysis and design [Tannuri et al. 2014]. The development of autonomous navigation systems based on simulators is a promising strategy to avoid the costs of real operation testing.

The goal of this application-oriented paper is to describe the development of an automated port navigation system that relies on batch reinforcement learning; the learning process is based on data collected from a high precision marine simulator, together with domain knowledge provided by dynamic modeling. We present a system that benefits from simulated fast-time trajectories using reinforcement learning, by exploring compact representation of ship dynamics. We contribute in two directions, the first related to ship control, and the second related to machine learning techniques as applied to this setting:

- We deal with discrete inputs (rudder and propeller) that are similar to real-world scenarios, and we use a detailed model of the relation between rudder efficiency and vessel velocity.

- We exploit efficient learning from random trajectories sampled from a simulator through a batch algorithm; moreover, we apply heuristics based on the domain in order to improve significance of samples, state-space compactness and reward design, allowing the problem to be efficiently tackled by conventional batch RL.

Before critical conditions are selected and tested in real-time simulations with local pilots control, engineers execute a large number of fast-time simulations using a trajectory computer control algorithm. The fast-time simulation process, however, is very time-consuming, as it requires a trial and error procedure for the definition of the way-points and the correct gains for the control algorithm. The trajectory output of the fast-time simulations does not necessarily take profit of the environmental forces (as the pilot does) and does not reflect the constraints imposed by real-world piloting, such as discrete lever commands in discrete intervals. As a first application, our techniques can replace the trajectory control algorithm used in the fast-time simulations. It can speed up the feasibility analyses since does not require the engineer to test a large number of way-points and controller gains to obtain an acceptable trajectory.

The paper is organized as follows. Sections 2 and 3 summarize key concepts respectively in reinforcement learning and in dynamic modeling and simulation. Section 4 gives a short overview of previous work applying machine learning to vessel berthing and port channel navigation. In Section 5 we describe our reinforcement learning solution, and in Section 6 we describe experiments that validate our solution. Finally, section 7 shows our conclusions and indicates future work.

2. Background: Reinforcement Learning and Fitted Q-Iteration

Reinforcement Learning (RL) uses sequential decisions of an agent to learn how to behave [Sutton et al. 2018]. At each time step, the agent observes a state, takes an action that results in a transition to another state and receives a reward signal. RL is based on the framework of Markov Decision Processes (MDPs); an MDP is described by a set of states S, a set of possible actions A, transition probabilities $p(s, a, s')$, a reward function $S \times A \times S \rightarrow R$ and a discount factor $0 \leq \gamma \leq 1$. The latter factor attenuates future rewards and is applied in infinite horizon tasks to guarantee convergence of accumulated rewards. The lower the discount factor, the more short-sighted the agent. A Markov property is assumed by MDPs: every transition depends only on s and a.

RL algorithms look for optimal policies maximizing cummulative reward (a policy is a function from states to actions). The Q value is the expected accumulated reward if action a is taken at state s as specified by a policy π; we have that $Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right]$.

The Q-Learning algorithm is widely used for RL. The main idea of the algorithm is to update the Q-value of a given state-action pair during an exploration (online) by incrementing the Q value from the subsequent state s' with a factor α:

$$Q_t(s, a) \leftarrow (1 - \alpha) \cdot Q_{t-1}(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q_t(s', a')).$$

Updated this way, Q values converge to the optimal policy if all states are sufficiently explored. Q-learning can be extended to continuous state-spaces, using for example supervised learning techniques to produce an estimate for the Q function.

Data: Set S of four-tuples $(s_t^l, a_t^l, s_{t+1}^l, r_{t+1})$
Initialization: $\hat{Q}_0(s,a) \leftarrow 0$ and $i \leftarrow 1$;
while $i < N$ **do**
  foreach *tuple* l *in* S **do**
    Train approximation $\hat{Q}_i(s,a)$ using supervised learning with
    $input : s_t^l, a_t^l$
    $target : r_{t+1} + max_a \hat{Q}_{i-1}(s_{t+1}^l, a)$
  end
  $i \leftarrow i + 1$;
end

Algorithm 1: Fitted Q Iteration.

While online RL focuses on algorithms that learn and adapt the policy during exploration, Batch Reinforcement Learning processes previously acquired transitions in order to obtain a policy. A Batch RL algorithm with excellent performance is Fitted Q Iteration (FQI) [Ernst et al. 2005, Riedmiller 2005], as it tends to be less susceptible to instabilities than other algorithms because the Q updates are not performed online for a given transition [Berlink et al. 2015]. In its most popular version, Neural Fitted Q Iteration [Riedmiller 2005], the regression technique used to store the Q function is a Multilayer Perceptron. Algorithm 1 describes the pseudo-code for the FQI algorithm.

Because it is not always possible to obtain transition samples that cover the whole state-space, the practical use of FQI resorts to a technique known as "Growing Batch" Learning [Lange et al. 2012] This technique alternates between learning iterations and episodes using the last policy learned; transitions of episodes are then added to the batch, thus incorporating representative transitions into the learning process.

3. Dynamic Modeling and Simulation

We start this section by summarizing the mathematical model that encodes the motion of a floating vessel at low speed in 6 DOFs (degrees of freedom), subjected to the external forces due to the environment and tugboats, and to the control forces provided by thrusters, propeller and rudder. We give here the main ideas behind the model; more detailed discussion can be found elsewhere [Queiroz Filho et al. 2014, Tannuri et al. 2014]. For the sake of simplicity, this section will only present the equations of motion for the horizontal plane. We adopt two different coordinate systems to derive the ship equations of motions, as shown in Figure 1. The system $OXYZ$ is Earth-fixed (inertial system) and the system $oxyz$ is ship-fixed, with the origin at the central point of the keel midship section. The center of gravity G is at distance x_G ahead from the point o, ox is the longitudinal axis of the vessel directed to the bow, and oy is the transversal axis, pointing to port. The heading of the vessel ψ defines the angle between the ox and OX axes.

The horizontal 3 DOF equations of motion referred to the body-fixed $oxyz$ coordinate system, considering symmetry with respect to the axis ox, are [Fossen 2011]:

$$(M + M_{11})\dot{u} - (M + M_{22})vr - (Mx_G + M_{26})r^2 = X_{ext},$$

$$(M + M_{22})\dot{v} + (Mx_G + M_{26})\dot{r} + (M + M_{11})ur = Y_{ext},$$

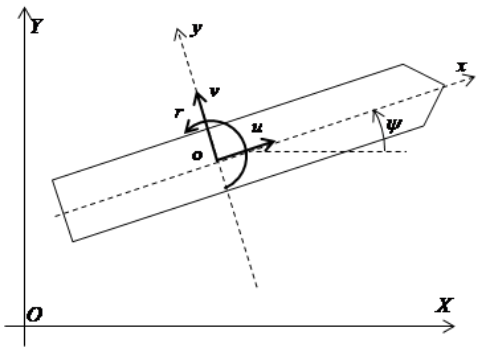

Figure 1. Ship coordinate system.

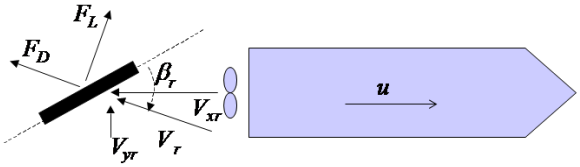

Figure 2. Forces on the rudder.

$$(I_z + M_{66})\dot{r} + (Mx_G + M_{26})(\dot{v} + ur) + (M_{22} + M_{11})uv = N_{ext},$$

where M is the vessel mass, I_z is the yaw moment of inertia of the ship, u and v are the longitudinal and transversal velocities respectively and r is the yaw angular velocity. The terms M_{11} and M_{22} are the ship added masses in the ox and oy directions, M_{66} is the ship added moment of inertia and M_{26} is the coupled added inertia. The last term on the right side of the yaw equation is the Munks moment. The subscript ext represents the external loads that may be expressed in terms of different factors:

$$X_{ext} = X_h + X_w + X_{wv} + X_p + X_{tug} + X_M,$$

where X_h represents the hydrodynamic non-potential forces, including the current and maneuvering forces, X_w, X_{wv} represent the wind and wave forces, respectively X_p represents the thrusters, propeller and rudder forces, X_{tug} represents the external action of the tug boats, either in contact with the hull or connected by a cable and X_M represents the forces due to mooring lines, fenders or anchor lines. In the present paper, the vessel navigates along a channel without any external action (wind, waves and current), controlled only by its own propeller and rudder (no tugs and mooring lines).

The rudder lift forces F_L are given by [Molland and Turnock 2007]:

$$F_L(\beta_r) = 0.5\rho A_r C_L(\beta_r) V_r^2,$$

where ρ is the water density, A_r is the area of the rudder, C_L is an dimensionless coefficient, β_r the effective rudder angle and V_r the relative velocity of the fluid onto the rudder (Figure 2). Rudder forces are dependent on the speed of the water on the rudder. This water jet is mainly generated by the propeller rotation; thus the vessel is more controllable (maneuverable) when the propeller is operating. With the engine stopped, the water flow on the rudder and the rudder forces are reduced, thus making the vessel less controllable.

We now describe the simulator we used in our experiments. As noted in Section 1, high precision simulation is routine in the feasibility analysis of ports. We have relied on the TPN-USP Maneuvering Simulation Center, shown in Figure 3, the largest Brazilian Ship Maneuvering Center. This center consists of six simulators, three of them classified as full-mission (immersive system with more than 270° of projection). The simulation can be executed in real-time mode in one or several cabins simultaneously (single or multiplayer simulation) as depicted in Figure 3. The simulator is used for evaluation of new ports and operations, risk analysis, pilot and captain training. The same simulation software can also be executed in fast-time mode, used in the early stages of port design, using control algorithms to maneuver the vessel.

Figure 3. TPN-USP Full maneuvering simulator.

The 6 DOF vessel dynamics differential equations presented before are solved by the simulator using explicit 4rd order Runge-Kutta integration, assuming interaction with the fluid and external forces acting on the hull of the vessel.

4. Previous Work

As indicated previously, there have been very few attempts to automate vessel berthing; we start by reviewing the relevant literature on this topic. A neural controller was used by [Ahmed and Hasegawa 2013]; in that work previously validated berthing trajectories were assumed available. In a different direction, [Goh and Lim 2000] used genetic algorithms and tabu-search to automate the generation of way-points used in berthing. A negative aspect of both approaches is that they require a great deal of validated data and/or area expertise.

Even though RL is clearly a possible approach to vessel berthing without previous trajectory samples, little has been done in this direction. Exceptions can be found in the work of [Stamenkovich 1992], where a simple neuron-like actor-critic agent navigated a ship through a channel through two sensors signal; more recently, [Lacki 2008] and [Rak and Gierusz 2012] employed online RL for ship handling in restricted waters assuming constant speed in a broad area with small obstacles (tabular algorithms with discretized state and function approximation for continuous states space were respectively used). Finally, [Tuyen et al. 2017] combined actor critic methods with neural networks to

control rudder with continuous levels also using constant speed. One common aspect of those efforts is that actions taken by the agent are considered continuous in time, meaning it operates more like a controller and not much in a human style with the already described limitations.

5. An RL Solution for Automated Port Channel Navigation

The task of the agent is to keep the ship at the center of the channel, aligned with the guidance line. The main control that affects a mission is the rudder steering to avoid a collision with the limits of the channel. Our learning strategy consists of:

- Collecting data from fast-time simulations for a given port channel using a sampling strategy;
- Learning from data using FQI with growing batch using a compact state representation.

The novelty here is the use of RL in batch mode with velocity fluctuations that represent the critical conditions of a real navigation in a port channel by a human pilot; this makes the policy learned more adequate for port engineering analysis. The overall goal of the agent is to maneuver the ship along the channel reaching the other end and with propulsion fixed at a certain level that suffices to prevent the ship from stopping or completely losing its steering capacity. Although simulation steps are constrained proportionally to the length of the channel, the mission can be modeled a control task infinite in time if the ship does not collide.

Sample tuples were obtained by running the numerical simulator in fast-time mode, with numerical integration time step $t = 0.5$s and interval between rudder commands $t_{control} = 10$s. As the channel is usually narrow and does not have high curvature, a guidance line was set considering the longest straight line formed by the boundary buoys at the channel. Such line splits the channel almost symmetrically for generating reflected tuples, as discussed in Section 5.1

5.1. Tuples Sampling

Once the guidance line is set, a trajectory following this line is generated. As sampling strategy, at every 50 steps of this first state trajectory the ship state is taken and stored as a starting point. For each starting point added, noise is added to the variables creating three additional starting points. At the end, trajectories are generated using all points stored. The actions taken in these trajectories are chosen randomly considering the discretization adopted in action space.

An additional technique used for attenuating the bias of samples was to mirror the transitions using the guidance line as symmetry line. This is only possible since the ship is not subjected to environmental conditions. If the reflected ship positions in the tuple fall inside the channel, velocity vectors are also reflected and the new resulting tuple is added to the batch (see Figure 4). Given the straight topology of the channel, almost all tuples are reflected.

5.2. State space Representation

Even though a ship is a complex system, and the propeller has its own internal states, prior experiments show that a simplification can be used. The time step adopted is large enough

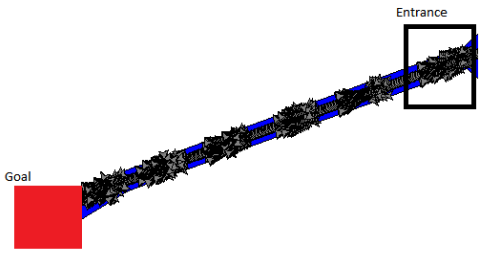

Figure 4. Layout of port channel. Small triangles represent the tuples sampled and reflected by symmetry.

to ensure that internal states are very close to equilibrium and ship position in relation to the channel length can also be disregarded since it does not affect the task. A simple three-dimensional state was then adopted with the following variables (see Figure 5):

- $v_{longitudinal}$: Speed over ground in the longitudinal direction of ship.
- $misalignment$: Ship heading angle subtracted from the guidance line heading angle.
- $bank\_balance$: Difference between ship and the banks (port side and starboard side) divided by the sum of both distances. Ship should ideally follow through the centerline of the channel, where the variable is null. The signal of the variable will reveal which bank the ship is getting closer to.
- $\dot{\psi}$: Variation rate of ship orientation. Also known as rate-of-turn.

The variable $v_{longitudinal}$ was included in order to take into account the agent based on the known loss of rudder steering capacity with lower velocity, mentioned in Section 3. And $\dot{\psi}$ was included following conventions of marine control systems, which usually require a derivative term for stabilizing ship orientation.

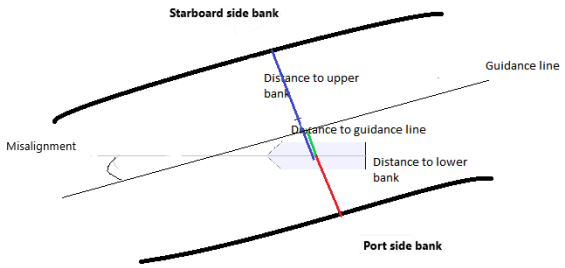

Figure 5. Distances for state and reward representations.

5.3. Actions

The action values represent rudder levels between -35 and +35 degrees, discretized as multiples of 5 (resulting in 13 possible action values). This discretization represents the commands usually given by human pilots in similar situations. The actual value sent by the agent, however, is normalized between -1 and 1.

5.4. Reward

Reward design is a heuristic task given knowledge about the domain of interest [Ng et al. 1999, Randløv and Alstrøm 1998]. We adopted a reward structure composed of two terms:

$$R(s, a, s') = R_{state} + R_{action}.$$

The term R_{state} reflects the desire of keeping the agent stable at the center of the channel, aligned with the guidance line:

$$R_{state} = -K_{balance} bank\_balance^2 - K_{misalign} misalignment^2 - K_{rot}\dot{\psi}^2.$$

where $bank\_balance$, $misalignment$ and $\dot{\psi}$ are the state variables already introduced in 5.2 and $K_{balance}$, $K_{misalign}$ and K_{rot} are positive factors to be tuned empirically. An additional constant $-C_{collision}$ can be added R_{state} when ship collides in order to propagate the danger of collision. The term R_{action} represents a form of policy shaping. As it is desirable to control the ship smoothly, using rudder levels as small as possible, the term penalizes the normalized action taken by the agent as $R_{action} = -K_{rudder} a^2$,, where K_{rudder} is a positive tunable parameter and a is the normalized rudder level given as command.

Using this reward structure, terms are never larger than 0 as negative rewards prevent problems in policy such as agent loops described by [Randløv and Alstrøm 1998].

5.5. Q function Approximation

A neural network with 2 hidden layers with 20 perceptrons each was employed to approximate the Q function. The ReLu (Rectifier Linear Unit) [Maas et al. 2013] activation function was used for hidden layers and a linear function was used in the output layer. ADAM [Kingma and Ba 2014] was chosen as the optimizer algorithm with learning rate $\eta = 0.001$. The choice of topology was based on other RL applications to control problems with non-linear dynamics and equivalent number of state variables [Hafner and Riedmiller 2011].

6. Experiments

The port channel for Suape Port, located in northeastern Brazilian coast, was taken as a testing scenario. The vessel model used in the experiments represents a real vessel and the propeller was set at 60% of its maximum rotation. The speed at the entrance point was set to 3.0 m/s, assuring the ship enough velocity to steer through the channel.

The agent was trained with discount factor $\gamma = 0.8$ and at every 10 iterations of FQI the agent run 12 episodes using the policy learned. For every FQI iteration 300 training iterations ("epochs") were run in the MLP and the loss function was verified along the iterations. Some error spikes occur at the beginning of new FQI iterations and when more batches are added as expected, but the process soon stabilizes, as in Figure 6. These trajectories, along with the reflected pairs generated through symmetry, were added to the batch.

To assess performance, 6 cases were considered with the ship starting at the region near the entrance of the channel. In three of them, the ship was placed at the guidance line ($bank\_balance = 0$) and $misalignment = \{0, 1.0, 2.0\}$. In other three cases, the

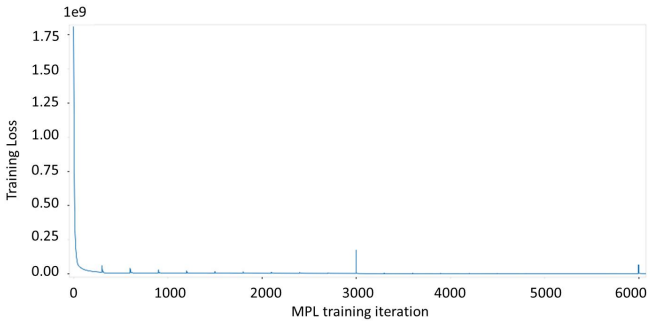

Figure 6. Loss function over each MLP training iteration.

Table 1. Results with ship starting at the guidance line, where $K_1 = K_{misalign}$, $K_2 = K_{rot}$, $K_3 = K_{rudder}$, $C = C_{collision}$.

	Initial state								
	Bank_balance:0; Misalignment:0			**Bank_balance:0; Misalignment:1**			**Bank_balance:0; Misalignment:2**		
Reward Parameters	$\bar{\sigma}$	$\|\bar{bal}\|$	$S_{\|bal\|}$	$\bar{\sigma}$	$\|\bar{bal}\|$	$S_{\|bal\|}$	$\bar{\sigma}$	$\|\bar{bal}\|$	$S_{\|bal\|}$
$K_1 = 100$; $K_2 = 1000$; $K_3 = 100$; $C = 100000$	0.85	73.85	3009.73	0.85	72.13	3158.82	0.85	65.85	3017.30
$K_1 = 1$; $K_2 = 10$; $K_3 = 10$; $C = 1000$	0.86	50.71	3574.6	0.86	52.77	3512.93	0.86	55.31	3567.55
$K_1 = 1$; $K_2 = 1$; $K_3 = 1000$; $C = 10000$	0.20	50.17	3095.43	0.29	43.23	2837.36	0.31	42.05	2676.55

ship was placed closer to the port side bank ($bank\_balance = 30$) and $misalignment = \{0, -1.0, -2.0\}$. The policy learned was evaluated based on smoothness of trajectory and commands; the parameters found to be appropriate in assessing a given trajectory were: the average magnitude of the rudder command $\bar{\sigma}$ (value normalized between $\{0, 1\}$), the average bank_balance magnitude $|\bar{bal}|$ and the variance for the bank_balance magnitude $S_{|bal|}$. The simulations were performed for policies learned with different reward function configurations and the results are displayed in Tables 1 and 2. Figure 7 illustrates the trajectories obtained using $K_{balance} = 1$, $K_{misalign} = 100$, $K_{rot} = 1000$, $K_{rudder} = 10000$, $C_{collision} = 100000$.

For a better evaluation of relative impact of the terms, the factor $K_{balance}$ was fixed at 1. $K_{misalignment}$ not only affects how the policy will attempt to realign the ship, but also counterbalance the urge for the ship to return to the middle of the channel totally unaligned. This prevents unstable situations. Factors such K_{rudder} and K_{rot} impact on how the agent is punished for attempting more abrupt orientation changes. This is reflected for the third reward configuration, where K_{rudder} is significantly higher than the other factors

Table 2. Results with ship starting with displacement from guidance line, where $K_1 = K_{misalign}$, $K_2 = K_{rot}$, $K_3 = K_{rudder}$, $C = C_{collision}$.

	Initial state								
	Bank_balance:20; Misalignment:0			**Bank_balance:20; Misalignment:-1**			**Bank_balance:20; Misalignment:-2**		
Reward Parameters	$\bar{\sigma}$	$\|\bar{bal}\|$	$S_{\|bal\|}$	$\bar{\sigma}$	$\|\bar{bal}\|$	$S_{\|bal\|}$	$\bar{\sigma}$	$\|\bar{bal}\|$	$S_{\|bal\|}$
$K_1 = 100$; $K_2 = 1000$; $K_3 = 100$; $C = 100000$	0.85	66.79	2850	0.85	67.92	2730	0.85	69.15	2637.63
$K_1 = 1$; $K_2 = 10$; $K_3 = 10$; $C = 1000$	0.85	55.18	2683.32	0.85	53.98	2203.60	0.85	56.60	2092.05
$K_1 = 1$; $K_2 = 1$; $K_3 = 1000$; $C = 10000$	0.15	38.38	1458.86	0.11	50.15	1611.13	0.12	52.46	1694.66

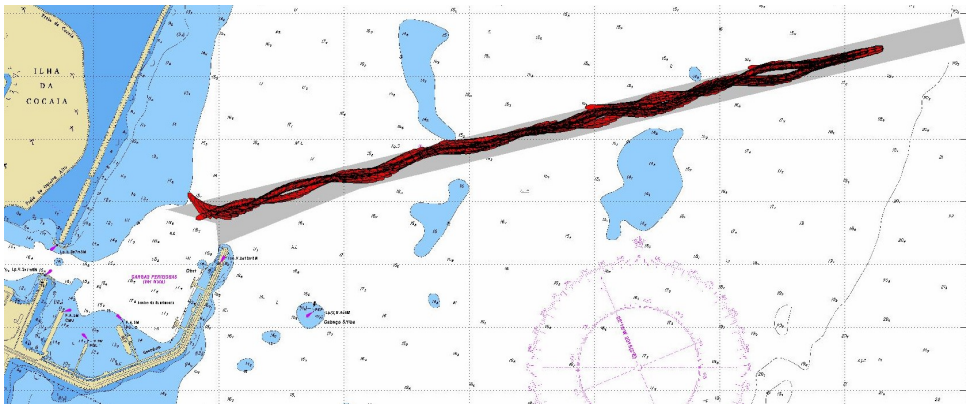

Figure 7. Trajectories for $K_{balance} = 1$, $K_{misalign} = 100$, $K_{rot} = 1000$, $K_{rudder} = 10000$, $C_{collision} = 100000$.

from R_{state}. A high rudder magnitude punishment, however, makes it impossible for the agent to escape from extreme misalignment and displacement situations.

One interesting behavior observed in these experiments was the unexpected actions taken by the agent in cases where the ship dangerously approached the margins with a large misalignment or with very low velocity. Instead of setting the rudder to the maximum angle in an attempt of dodging from collision, it simply took the action that anticipates it. Two possible explanations for that are: the lack of samples with low punishment or a reward function which causes the discounted accumulated reward to be as negative as the collision punishment. Since the horizon appears to have a highly negative accumulated reward, the agent goes for the the "unavoidable" collision state faster in order to minimize the accumulated punishment.

7. Conclusion and Future Work

In this paper we have presented a batch reinforcement learning solution for ship maneuver control. Channel navigation for berthing was modeled using a compact state-space, and no way-points were required; ship behavior was simulated so as to reflect the limitations of human steering and ship maneuverability. Experiments show that human-style controls can be generated this way, using discrete command levels with continuous propulsion.

As future work, propulsion level changes should be considered as actions to achieve higher controllability. Also, more complex port topologies and robustness against conditions such as wind and current, should be investigated. In both cases, additional state variables may be necessary. As a long term goal, it is desirable to include commands to tugs that tow the ship until the berth point to attain full emulation of autonomous ship berthing. In the scope of channel navigation, there are further directions for research: improvement of controllability through more action levels, robustness against environmental forces, and use of varied port topologies.

Acknowledgements

Thanks to Petrobras for supporting the development of the maneuvering simulation center. The second author is partially supported by CNPq grant 304784/2017-6. The third author, by CNPq grant 308433/2014-9, and FAPESP grant 2016/18841-0. The fourth author, by CNPq grants 425860/2016-7, 307027/2017-1 and FAPESP grant 2016/21047-3.

References

Ahmed, Y. A. and Hasegawa, K. (2013). *Implementation of automatic ship berthing using artificial neural network for free running experiment*, volume 9. IFAC.

Berg, T. E. and Ringen, E. (2011). Validation of Shiphandling Simulation Models. In *Volume 1: Offshore Technology; Polar and Arctic Sciences and Technology*, pages 705–712. ASME.

Berlink, H., Helena, A., and Costa, R. (2015). Batch Reinforcement Learning for Smart Home Energy Management. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, (Ijcai):2561–2567.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6(1):503–556.

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

Goh, K. S. and Lim, A. (2000). Combining various algorithms to solve the ship berthing problem. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 2000-Janua:370–373.

Hafner, R. and Riedmiller, M. (2011). Reinforcement learning in feedback control : Challenges and benchmarks from technical process control. *Machine Learning*, 84(1-2):137–169.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Lacki, M. (2008). Reinforcement Learning in Ship Handling. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, 2(2):157–160.

Lange, S., Gabel, T., and Riedmiller, M. (2012). Batch Reinforcement Learning. In *Reinforcement Learning: State-of-the-Art*, pages 45–73.

Laurinen, M. (2016). Remote and Autonomous Ships: The next steps. Available at: http://www.rolls-royce.com/˜/media/Files/R/Rolls-Royce/documents/customers/marine/ship-intel/aawa-whitepaper-210616.pdf. Technical report.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *International conference on machine learning*.

Molland, A. and Turnock, S. (2007). *Marine Rudders and Control Surfaces*. Elsevier/Butterworth-Heinemann.

MUNIN (2016). Research in maritime autonomous systems project results and technology potentials. Available at: http://www.unmanned-ship.org/munin/wp-content/uploads/2016/02/MUNIN-final-brochure.pdf. Technical report.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations : Theory and application to reward shaping. *Sixteenth International Conference on Machine Learning*, 3:278–287.

Queiroz Filho, A. N., Zimbres, M., and Tannuri, E. A. (2014). Development and Validation of a Customizable DP System for a Full Bridge Real Time Simulator. In *International Conference on Ocean, Offshore and Arctic Engineering - OMAE 2014*, volume 1A, page V01AT01A047.

Rak, A. and Gierusz, W. (2012). Reinforcement learning in discrete and continuous domains applied to ship trajectory generation. *Polish Maritime Research*.

Randløv, J. and Alstrøm, P. (1998). Learning to Drive a Bicycle using Reinforcement Learning and Shaping. *Proceedings of the International Conference on Machine Learning (ICML)*, pages 463–471.

Riedmiller, M. (2005). Neural fitted Q iteration - First experiences with a data efficient neural Reinforcement Learning method. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3720 LNAI:317–328.

Stamenkovich, M. (1992). An application of artificial neural networks for autonomous ship navigation through a channel. In *IEEE PLANS 92 Position Location and Navigation Symposium Record*, pages 346–352, Monterey, CA, USA. IEEE.

Sutton, R. S., Barto, A. G., and Bach, F. (2018). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press.

Tannuri, E. A., Rateiro, F., Fucatu, C. H., Ferreira, M. D., Masetti, I. Q., and Nishimoto, K. (2014). Modular Mathematical Model for a Low-Speed Maneuvering Simulator. In *Proceedings of the 33th International Conference on Ocean, Offshore andArctic Engineering (OMAE2014)*, pages 1–10, San Franscisco, USA.

Tuyen, L. P., Layek, A., Vien, N. A., and Chung, T. (2017). Deep reinforcement learning algorithms for steering an underactuated ship. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*.