Development of Onboard Decision Supporting System for Ship Docking Operations

Luman Zhao, Guoyuan Li, Knut Remøy, Baiheng Wu, and Houxiang Zhang
Department of Ocean Operations and Civil Engineering
Norwegian University of Science and Technology
Postboks 1517, N-6025, Aalesund, Norway
{*luman.zhao, guoyuan.li, knut.remoy, Baiheng.wu, hozh*}*@ntnu.no*

Abstract—**Maritime operations are inevitably influenced by the wind, wave, sea currents, and other perturbations at sea. Providing decision support for these operations based on historical and real-time data of ship status is thus of great concern in terms of ship safety. However, it is challenging for collecting and analysis large quantities of ship data in real operations. Moreover, the development of an onboard decision support system (DSS) will be a gradual and iterative process subject to extensive testing and simulation. Consequently, the paper presents an integrated simulation framework which provides testing and simulation environment for the DSS development. The system enables navigation data transmission from a well-designed simulator and automatic determining of the safe maneuver of a ship within the framework. The development of DSS is divided into three steps. The ship maneuvering data from simulator is collected; the data is then classified and fed into an imitation learning (IL) algorithm to learn an initial policy; the result is further applied to an reinforcement learning (RL) algorithm for safe decision making of the operation. In this way, it could speed up the learning efficiency by extracting more information from available experience. To verify the effectiveness of the proposed integrated simulation framework, in this study, we implemented the proposed DSS in ship docking operation under various environmental disturbances. It is interacted with the simulator to obtain data. By processing these data, it provides the shipmaster with the information about the consequences of the ship maneuvering decisions. The simulation results demonstrate that the proposed DSS could assist the shipmaster in deciding policies and increase the efficiency of decision making.**

Index Terms—**ship docking, decision supporting system, imitation learning (IL), reinforcement learning (RL)**

I. Introduction

In congested sea conditions, the marine operation becomes difficult due to much consideration of positioning and heading requirements in a short time. Besides, the presence of uncertainty, in the form of environmental disturbances like wind, wave, and sea current, further increases maneuvering complexity. Analysis based on the Norwegian Maritime Directorate incident database shows risk influencing factors in maritime accidents could relate to weather, geographic, visibility, technical failures, and human errors [1]. Notably, human-based error is the main reason for that. For instance, for ship docking application, current knowledge of practices, and risk management on board the ships is primarily based on experience. Therefore, to reduce human-based errors in ship operations, a more intelligent decision support system (DSS) based on fact-based knowledge and circumstances concerning the ship-environment status are needed to increase ship safety [2]. The DSS can be used to observe the surrounding, predict the ship's responsive motions and supply the shipmaster with information on suitable control command. There are already existing works about DSS development in several fields such as UAV [3], and autonomous ships [4, 5]. However, the challenge in developing such a system comes from two respects: 1) how to obtain the motions of the ship dynamically, and 2) what data should be included and how to use the collected motion data to extract the suitable information for decision making.

To achieve this, at first, we propose an integrated simulation framework for DSS by integrating a well-designed simulator. The simulator can offer extensive scale data that can be collected much faster than in the real world. Based on this, the DSS can analyze the real-time data on sea conditions alongside historical information about a ship's performance to set its optimum action. From the simulator point of view, the DSS will be designed for use in crew training, supporting the crew with information on how the ship will perform in real life. From the DSS point of view, it will take advantage of the significant amount of data available for analysis, to provide onboard support, such as visualizing complex surroundings and illustrating future prediction of the situation for either achieving autonomy or remote control.

Secondly, we use these dataset as input to an reinforcement learning (RL) algorithm that decides what action to take. Even though RL has been widely applied in autonomous ships and showed a good performance in applications such as auto docking [6], autopilot [7, 8], collision avoidance [9, 10], and so on, it remains relatively limited primarily due to 1) extremely high time cost of exploring and sampling. 2) highly rely on the problem definition, especially the reward function design. An alternative data-driven method is called imitation learning (IL). It learns an optimal policy by imitating the expert's decision from demonstrations data instead of learning from a specified reward function [11]. One of the limitations of IL is the requirement of plenty of demonstrations data. Moreover, the obtained policy works efficiently in specific simple applications but hard to scale very well. Consequently,

The project is financially supported by grants from the Research Council of Norway with the MAROFF KPN "Digital Twins for Vessel Life Cycle Service" (Project NO. 280703).

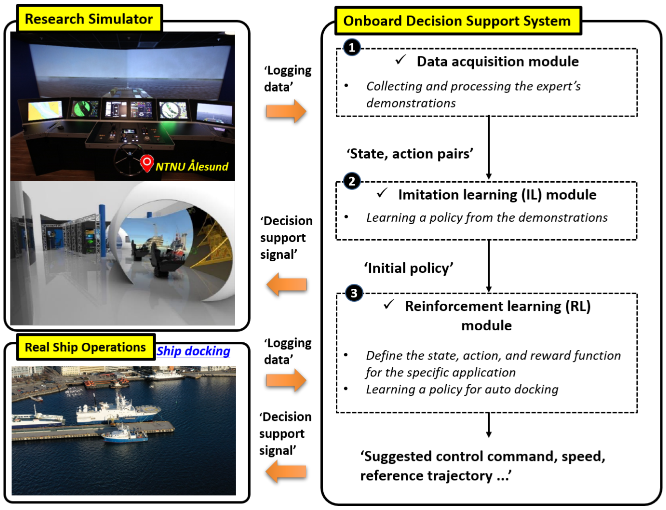
Research Simulator
NTNU Ålesund
Real Ship Operations
Ship docking
'Logging data'
'Decision support signal'
'Logging data'
'Decision support signal'
Onboard Decision Support System
1
✓ Data acquisition module
• Collecting and processing the expert's demonstrations
'State, action pairs'
2
✓ Imitation learning (IL) module
• Learning a policy from the demonstrations
'Initial policy'
3
✓ Reinforcement learning (RL) module
• Define the state, action, and reward function for the specific application
• Learning a policy for auto docking
'Suggested control command, speed, reference trajectory ...'

Fig. 1. Overview of integrated simulation framework for DSS.

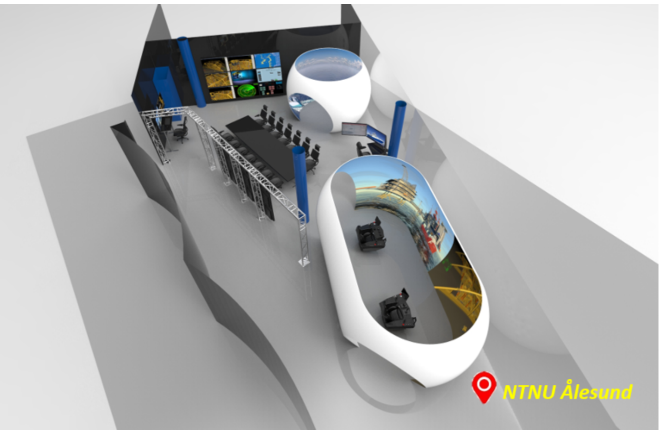

Fig. 2. Layout of the NTNU research simulator.

combining IL and RL can be an efficient way for complex marine operation in the context of shipmaster's demonstrations are available from a simulator. Given such demonstrations, IL algorithm can be used to train a policy that maps perceptual inputs to action, then use the initial policy to fine-tuning the RL policy. [12, 13]

In this paper, we propose an integrated simulation framework to provide the test environment for the DSS. Section II introduces each component for the framework. In Section III, we present the implementation for ship docking operation using the intelligent DSS, which combines the IL and RL modules inside. The simulation result will be shown in Section IV. Finally, the conclusion and future works will be discussed in Section V.

II. Components for the Integrated Simulation Framework

This section presents the definitions and theoretical background of the integrated simulation framework. Fig. 1 illustrates the scheme of integrated simulation framework, which involves a well-designed simulator and the DSS. The simulator used in this study is from Norwegian University of Science and Technology (NTNU) research laboratory located at Ålesund campus; the DSS is a system where logging data obtained from the research simulator, making informed decisions regarding ship performance under environmental conditions and sending the suggested information such as reference trajectory, velocity, and control command back.

A. NTNU research simulator

As shown in Fig. 2, NTNU research simulator is a center that aims to test technology, methodology, and procedure for remote control of various functions on ships. The research area is included all work stations on a ship or in a control center for manually, autonomous or semi-autonomous control. It includes navigation bridge, operation bridge (aft bridge), engine room, crane, ROVs, winch, and operation manager (on-shore and offshore). All interfaces are real interfaces commonly used in the industry. In addition, all the work stations are flexible to perform various navigation scenarios. Control operation can be replaced for customized set-up, and workplaces can be modularized, and a minimum of interfaces to simplify testing of equipment of different tasks.

The main objective of the research simulator is to simulate the different expected conditions and train the navigators to make decisions based on real-time data. It also provides high-quality and updated weather forecasts to help them to understand the ship's behavior better, and take the appropriate action due to different environmental conditions. The navigators have to plan a route based on their experience during the training process. At the same time, a significant amount of simulation data could be stored.

B. Configuration of DSS

Based on the stored dataset, we use the proposed DSS to learn a policy that will support the onboard decision making. It contains three modules: data acquisition module, IL module, and RL module.

1) Data acquisition module: The data acquisition module is responsible for collecting and pre-processing demonstrations data from the research simulator.

2) IL module: Given such demonstrations data, the IL module can be used to learn a maneuvering policy that achieves similar performance compared to the expert. As there already exists the simulator platform built through system engineering principles, we could collect the expert's demonstrations instead of implementing expensive real ship experiments.

Behavior cloning (BC) is one of the imitation learning techniques, which treats imitation learning as a supervised learning problem. In BC, we have the labeled state-action pairs from the expert's demonstrations, and we use the IL to train a policy by using the loss function between the expected output variable and the labeled target data (action). The goal is to approximate the policy so that we can mimic the behavior model of expert's demonstrations. As a result, IL is used to learn the sequential decision-making policy that maps

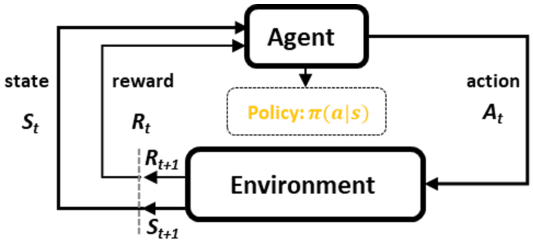

Fig. 3. Overview of reinforcement learning.

perceptual inputs to outputs. The learning result can either serve for the initial policy for RL or formulate the references as prior knowledge for prediction of future operation. However, it is sometimes hard for adequate expert demonstrations. More importantly, the learned policy can only perform at most as well as the guiding expert's policy and it may have poor performance in new unseen situations due to a shift from the training data distribution.

3) RL module: Unlike other approaches to machine learning such as supervised learning, where a batch of data for training is made available for the agent, RL methods depend on gathering this data in a process, where an agent interacts with an environment (e.g., an autonomous ship) by following a policy, as illustrated in Fig. 3. Note that the policy here is always initialized stochastic.

RL aims to deal with teaching the agent the connection in between states and actions, known as state-action pairs $\{(s, a)\}$, with the aim of maximizing a user defined reward r. Subsequently, with environments of strong interconnections of such pairs and future rewards possible RL is rendered a complex problem. In each state of the environment, it takes action based on the stochastic policy $\pi(a|s)$, and as a result, receives a reward r and transitions to a new state. The goal of RL is to learn an optimal policy that maximizes the long-term cumulative rewards [14].

RL has shown great potential in improving system performance autonomously by learning from iterations with the environment. It has achieved significant progress in solving this sequential decision problem, from autonomous driving cars like Google Waymo [15], to playing video games [16]. The RL algorithm for solving these problems generally can be categorized into model-free and model-based approaches. Model-free RL algorithms are capable of solving a wide range of control problems. However, it typically requires a huge number of samples to achieve good performance. Also, it can suffer from vast and high dimensional possible state and action space, which can result in an insufficient exploration or prohibitively long training time. Therefore, direct application model-free RL is not only sampling insufficient, but costly and dangerous in our applications. Although model-based RL may require fewer samples, it can lead to suboptimal, potentially unstable results [17, 18]. Considering these limitations, we use the initially trained policy from the IL module to initialize the model-free RL policy. In this way, the model-free RL could learn complex tasks using relatively small samples when compared to purely model-free RL. Thus we combine the benefits of IL and model-free RL by using the initial policy to initialize a model-free RL.

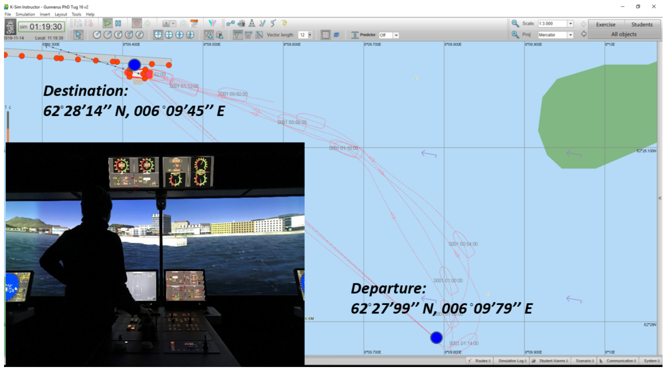

Fig. 4. A ship docking scenario manoeuvred by a shipmaster in the research simulator.

C. Onboard decision support for ship operation

Based on the integrated simulation framework (see Fig 1), the proposed DSS could offer various decision supports for shipmasters or automatic maneuvering in different operational scenarios. These include a reference trajectory, control comment, and guidance speed. Firstly, the RL module could compute a reference trajectory before the ship prepares to depart. Second offerings could be the control command like rudder angle and propeller revolution during the ship maneuvering. In addition, as the ship speed is a significant factor to affect the ship safety, it could be regarded as a decision support element.

III. Implementation of the proposed DSS

A. Ship docking scenarios

An expert's demonstration dataset is generated by the operation from shipmasters in the integrated simulator, as shown in Fig. 4. The simulator offers extensive data about the ship and environment that can be collected much faster than in the real world. To collect expert's demonstrations, experienced shipmasters are asked to perform ship docking operation to Ålesund harbor. Here we assume that the human operators perform perfectly, which ensures that these demonstrations data could be used to learn a good docking policy representation.

To generate a good initial policy from the IL module, we have to make sure that the data used for training must consist of situations with enough diversity, such as various initial heading angles. How much the training dataset explores the environment would play a vital role in the learned policy's performance. As shown in the table I, we perform the ship docking operations from a defined starting point to the Ålesund harbor for eight scenarios, and repeat three times for each scenario to get more demonstrations. Here, the initial heading angle is set from 0 degrees to 315 degrees with a 45-degree interval. The wind direction and wind speed are set as 100 degrees and 10.5 knots respectively. After the collection of

TABLE I
SIMULATION SETUP FOR SHIP DOCKING SCENARIOS.

Num.	Wind direction and speed [deg] [knots]	Start and end position [deg]	Initial heading [deg]
1	100 10.5	62°27'99" N, 006°09'79"E 62°28'14"N, 006°09'45"E	0
2			45
3			90
4			135
5			180
6			225
7			270
8			315

demonstrations, a dataset $\mathcal{D}$ including information about ship position, ship velocity, control command is saved for training an initial policy with IL.

B. Problem definition of IL module

The goal of the imitation learning is to learn a basic policy that imitates the behavior of the shipmaster. Given the expert's demonstration dataset $\mathcal{D}$ of states and actions collected during the 24 sets scenarios over T time steps, the loss function for learning a policy parametrized by θ by supervised learning can be written as

$$\mathcal{L}(\theta) = \frac{1}{2}(a_{\pi_\theta}(s_t) - a_t)^2 \tag{1}$$

where $a_{\pi_\theta}(s_t)$ is the action predicted by the policy-network π_θ at state s_t and a_t is the labeled action from the demonstrations. As a result, taking the expert's docking policy as a supervision signal, we can get a relatively good policy from the IL module. Then we can use it as an initial policy for the subsequent reinforcement learning phase, which can significantly reduce the training time, stabilize the training process, and produce better results than training from scratch.

C. Problem definition of RL module

1) RL setup: The sequential decision-making for the auto-docking problem can be formulated as a Markov decision process (MDP) in an RL framework illustrated in Fig. 3. The decision-maker, i.e., the ship, also called as an agent π_θ, parametered by θ at each time step t, executes an action $a_t \in \mathcal{A}$, at state $s_t \in \mathcal{S}$ in the environment; the environment, in turn, yields a new state s_{t+1} and a reward $r_t \in \mathcal{R}$. The definition of the state, action, and reward function for the ship docking task is described as follows:

- As illustrated in Fig 5, the ship uses the reference system with velocities being surge u, sway v and yaw r. The ship position P_t is saved in Universal Transverse Mercator (UTM) coordinates in the raw dataset, and we translate it to a Cartesian coordinate with the origin at start point of a docking demonstration. P_{goal} represents the goal position near the port. We define a state g_t as a ship's relative goal position, i.e. the coordinates of the goal in the ship's local polar coordinate frame. ψ_t and χ_t refer to the heading angle and course angle of the ship. $\tilde{\phi}_t$ is the relative angle between the course angle of the ship and angle pointing to the destination from the ship. α_t is the relative angle between the heading angle of the ship and the quay. In addition, the previous predicted action a_{t-1} is also included in the state vector. Thus, the state space is represented as follows:

$$s_t = [P_t,\, g_t,\, \psi_t,\, \chi_t,\, \tilde{\phi}_t,\, \alpha_t,\, a_{t-1}] \tag{2}$$

- During ship maneuvering, the shipmaster steers a ship by controlling the rudder angle. The rudder angel δ can create a moment about the centre of gravity of the ship, and thus change the ship's orientation by giving a drift angle correspondingly. As a result, we set the action space as $a_t = \delta$ in this study.
- The reward function is computed as the sum of the rewards accumulated in each episode. It is a measurement of the quality of the action. At first, the reward function can be specified to reward the ship for approaching the destination. It is designed to constraint the ship to reach the docking position next to the quay. The goal reward R_{goal} is designed to guide the ship to achieve the destination. This can be expressed mathematically as:

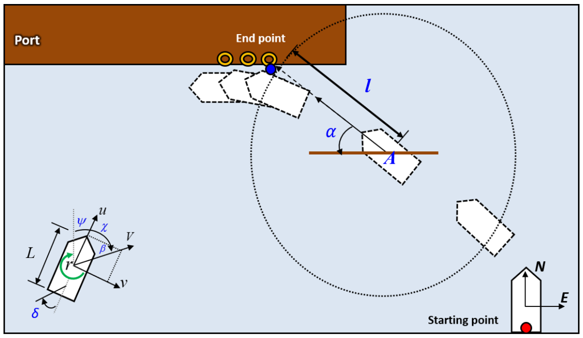

Fig. 5. State and action space definition for the RL problem.

$$R_{\text{goal}} = \begin{cases} r_{\text{goal}} & \text{if } \|P_{\text{goal}} - P_t\|_2 < 10.0 \\ \lambda_{\text{g}}(\|P_{\text{goal}} - P_{t-1}\|_2 - \\ \|P_{\text{goal}} - P_t\|_2) & \text{otherwise} \end{cases} \tag{3}$$

where λ_g refers to a hyper-parameter. When the ship is directly approaching the destination, the more substantial goal reward value is imposed on the agent. Second, the reward function can be specified to reward the ship to arrive with the correct heading. Consequently, $R_{heading}$ will help training converge in guiding the ship towards and parallel to the quayside. The angle error between the heading angle of the ship and the quay α_t should be small than a predefined parameter α_ϵ.

$$R_{\text{heading}} = \begin{cases} \lambda_h(\alpha_\epsilon - \alpha_t) & \text{if } \|P_{\text{goal}} - P_t\|_2 < 3 * L \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The reward received by the ship at timestep t is a sum of the two terms above.

2) Training process: Given the input s_t and output a_t, we create the a policy neural network mapping s_t to a_t and a critic network to predict a state value function for each state.

To represent the policy network, we use a fully-connected multilayer perceptron with two hidden layers consisting of 64 and 64 hidden units respectively with $tanh$ nonlinearities predicting the probability over the action space. In the process of training, the state is transmitted to the neural network, and the agent selects and executes an action according to the predicted result with the highest probability.

Training of the critic and policy networks is performed by defining the surrogate loss functions for each network. Then, back-propagate gradients computed with the surrogate loss function are used to update the weights of the network. We refer to the network trained with this approach as PPO [19], as shown in Algorithm 1. Note that the policy network training was initialized by the policy based on expert's demonstrations. This enables a transition from IL to RL without performance degradation and improves RL in terms of overall performance and reduces training time.

Algorithm 1 Combining IL with RL for ship docking.

```
Input: Demonstrations dataset $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t)\}_{t=1}^{N}$.
First: Train the policy network $\pi_\theta$ by IL
for $iteration = 1, 2, \ldots, N_{IL}$ do # imitation learning
    Update $\pi_\theta$ with the Equation 1 by gradient descent
end for
Second: Initialize value network $V_\phi(s_t)$, train $\pi_\theta$ and $V_\phi(s_t)$ by RL
for iteration $= 1, 2, ..., N_{RL}$ do
    Run policy $\pi_\theta$ for $T$ timesteps, collecting $\{\mathbf{s}_t, r_t, \mathbf{a}_t\}$, where $t \in [0, T]$
    Estimate advantages, $\hat{A}_t = \sum_{l=0}^{T} (\gamma\lambda)^l \delta_t$, where $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
    break, if $T > T_{max}$
    $\pi_{old} \leftarrow \pi_\theta$
    for $j = 1, ..., E_\pi$ do
        $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$
        $L^{PPO}(\theta) = \sum_{t=1}^{T_{max}} \mathbf{min}(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$
        Update $\theta$ with $lr_\theta$ w.r.t $L^{PPO}(\theta)$
    end for
    for $k = 1, ..., E_V$ do
        $L^V(\phi) = -\sum_{t=1}^{T} (\sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t))^2$
        Update $\phi$ with $lr_\phi$ by Adam w.r.t $L^V(\phi)$
    end for
end for
```

IV. Simulation result for ship docking operation

As described in the previous sections, we collect docking trajectories from the expert's maneuvering in the integrated simulation framework. Then we use the dataset to train an initial policy for RL using BC.

In the beginning, we train a ship from a starting position to converge to a defined quayside. The general description of the ship is presented in II. The final heading will parallel with the quayside in 90 degrees. During the training process, the starting position and initial heading are generated randomly. The starting position is selected within a 5-meter range, and the initial heading is selected within 360 degrees range.

TABLE II
General description of the ship.

Model: **Ship Assist Tug**	**Value**
Length O.A.	30.8 [m]
Beam	11.1 m
Draft	3.5 [m]
Displacement	615 [ton]
Speed ahead	12.5 [kn]

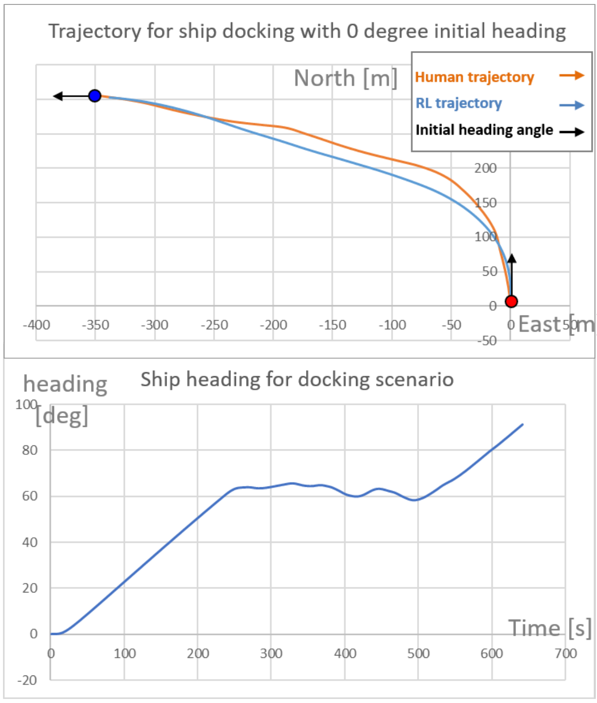

Fig. 6. Ship trajectory and heading with the 0 degree initial heading: compare with the expert's demonstrations

When the ship achieves reliable performance, we save the trained policy and test it using a defined starting position and heading angle. To validate the feasibility of the proposed algorithm, we compare the ship trajectory using the RL with the expert's demonstration in the simulator. As shown in Fig. 6, we can find that the behavior of the training result is very similar to the expert. When we arbitrary set the initial heading as -37 degrees, the ship can arrive at the quay smoothly with an approximately 90 degrees heading. The simulation result is illustrated in Fig. 7.

Fig. 8 shows the average reward during the training of the RL policy. We can find that the reward function increases along with the training process.

V. Conclusion and future works

In this study, we proposed an integrated simulation framework, which aims to create a reliable decision support system (DSS) that will assist the shipmaster in deciding action and efficient ship maneuvering. We take the docking operation as

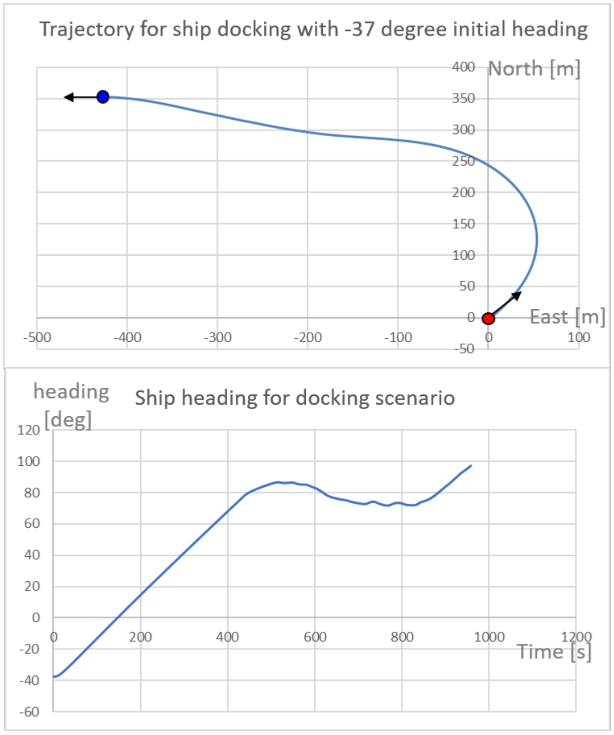

Fig. 7. Ship trajectory and heading with the -37 degree initial heading.

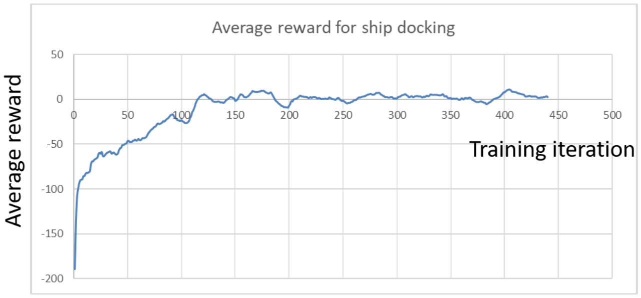

Fig. 8. Average reward for ship docking.

an example to verify the effectiveness of the DSS. In such scenarios, ships should be carefully aware of their surroundings and make decisions based on the sensor input. Under harsh environmental conditions, the ship docking operation mainly relies on the human experience and knowledge of the past to make decisions. Consequently, an intelligent DSS for docking operation is developed. The demonstrations dataset is collected from a research simulator then transmitted to an IL module, which can efficiently obtain intelligence of decision making for docking scenarios. It can be treated as guidance to accelerate the training efficiency of RL. RL can calculate the corresponding action under various environmental conditions and initial heading. Thus combining IL and RL can be an efficient and promising method for the ship docking application.

The experimental results show that the resulting imitation policies perform favorably compared to those generated by existing imitation learning approaches that do require access to demonstrator actions. We can conclude that the DSS has the potential to provide decision support and predictions for the shipmaster.

The proposed DSS will validate by the real ship operation in the future. We believe that the dataset from real ship operations can be used to further research in this area with multiple and complex tasks. Furthermore, it can be considered to be applied to achieve an auto-docking operation.

Acknowledgment

Thanks to Arnt Håkon Barmen at Department of Ocean Operations and Civil Engineering in NTNU for data support and insightful comments.

References

[1] P. Stornes, "Risk influencing factors in maritime accidents: an exploratory statistical analysis of the norwegian maritime authority incident database," 2015.

[2] E. Jokioinen, J. Poikonen, R. Jalonen, and J. Saarni, "Remote and autonomous ships-the next steps," *AAWA Position Paper, Rolls Royce plc, London*, 2016.

[3] I. Panella, "Artificial intelligence methodologies applicable to support the decision-making capability on board unmanned aerial vehicles," in *2008 Bio-inspired, Learning and Intelligent Systems for Security*. IEEE, 2008, pp. 111–118.

[4] L. Perera, J. Rodrigues, R. Pascoal, and C. G. Soares, "Development of an onboard decision support system for ship navigation under rough weather conditions," in *Sustainable Maritime Transportation and Exploitation of Sea Resources*. ROUTLEDGE in association with GSE Research, 2011, vol. 837, no. 844, pp. 837–844.

[5] P. A. Sarvari, E. Cevikcan, M. Celik, A. Ustundag, and B. Ervural, "A maritime safety on-board decision support system to enhance emergency evacuation on ferryboats," *Maritime Policy & Management*, vol. 46, no. 4, pp. 410–435, 2019.

[6] B. E. W. Mothes, "Reinforcement learning for autodocking of surface vessels," Master's thesis, NTNU, 2019.

[7] Y. Cui, S. Osaki, and T. Matsubara, "Reinforcement learning ship autopilot: Sample-efficient and model predictive control-based approach," *arXiv preprint arXiv:1901.07905*, 2019.

[8] Y. Zhang, P. Sun, Y. Yin, L. Lin, and X. Wang, "Human-like autonomous vehicle speed control by deep reinforcement learning with double q-learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1251–1256.

[9] L. Zhao and M.-I. Roh, "Colregs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Engineering*, vol. 191, p. 106436, 2019.

[10] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," *arXiv preprint arXiv:1808.03841*, 2018.

[11] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[12] W. Du and Y. Ji, "An empirical comparison on imitation learning and reinforcement learning for paraphrase generation," *arXiv preprint arXiv:1908.10835*, 2019.

[13] T. Wang and D. E. Chang, "Improved reinforcement learning through imitation learning pretraining towards image-based autonomous driving," *arXiv preprint arXiv:1907.06838*, 2019.

[14] R. S. Sutton and A. G. Barto, "Reinforcement learning an introduction–second edition, in progress (draft)," 2015.

[15] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[17] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, “Agile autonomous driving using end-to-end deep imitation learning,” in *Robotics: science and systems*, 2018.

[18] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, “Model-based reinforcement learning via meta-policy optimization,” *arXiv preprint arXiv:1809.05214*, 2018.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.