Ocean Engineering 282 (2023) 114989

Contents lists available at ScienceDirect

Ocean Engineering

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

Unmanned surface vehicle navigation through generative adversarial imitation learning

Piyabhum Chaysri *, Christos Spatharis, Konstantinos Blekas, Kostas Vlachos *

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

ARTICLE INFO

Keywords:
Imitation learning
Generative adversarial
Unmanned surface vehicle navigation
Knowledge reusing

ABSTRACT

In the artificial intelligent and big data technology era, the marine industry among others is inevitably developing in this direction, aiming at becoming autonomous and completing tasks without relying on human involvement while providing safety. The technology of small unmanned surface vehicles (USVs) is relatively mature but with a large development potential and wide research interest expecting significant benefits such as safety and high efficiency in shipping and transportation systems. This article addresses these issues and utilizes an imitation learning algorithm to resolve autonomous navigation for USVs even in complex environmental conditions. We formulate the trajectory modeling as a data-driven imitation learning problem where we employ a state of the art imitation learning algorithm. Experiments are performed in a particular simulated environment tailored to match the specific weather conditions of the local area. The simulation results show the potential of the proposed imitation learning scheme to create advanced intelligent agents for USVs under real-world environmental settings, and USV actuation constraints that allow to predict trajectories with high accuracy and safety.

In addition, we evaluated the method's robustness in generating successful trajectories under environmental conditions that differed from those encountered during training, thereby promoting knowledge reusing without the need for retraining.

1. Introduction

During the last years we are witnessing a constantly increasing use and interest for Autonomous Systems in marine applications. The development of control techniques, based on artificial intelligence (AI), to generate autonomous marine vehicles has received a lot of attention from the research community and has become a lighthouse of the strategic research in this domain. In this perspective, AI manages to handle uncertain and heavily-constrained dynamic systems by providing its ability to adapt to changes in the environment and to implement efficient decisions.

Autonomous systems become more and more a core component of transportation systems (Rahman et al., 2017). Dealing with Unmanned Surface Vehicles (USVs) currently displays remarkable progress. There are plenty of benefits of making surface vehicles unmanned such as shipping flexibility, reducing costs, and minimizing the impact, limitation and cost of human operators (Qiao et al., 2022). USVs have been involved in military, research, and commercial applications, including surveillance, data collection, and sea, surface and space communication hubs (Liu et al., 2016).

Motion control of USVs is an essential part and constitutes an important challenge that can increase its autonomous operation. The marine environment is complex due to the appearance of environmental factors and stochastic weather conditions such as wind, waves and currents that lead to large disturbances. Therefore, it is necessary to build more comprehensive and accurate ship kinematic models able to achieve ship motion prediction and compensation in advance (Bai et al., 2022).

Ship navigation algorithms favor machine learning, reinforcement learning, and deep reinforcement learning, among other algorithms, which have a higher accuracy than the traditional algorithms because they can learn by themselves and search for an accurate path faster when facing a real situation. Most learning algorithms applied in marine platform navigation and modeling, set the ships' velocity to a fixed value and do not consider the influence of wind, waves, and currents during their process, see Bai et al. (2022). This constitutes a disadvantage of algorithms that might not allow a successful integration into the navigation in the real world environment. In addition, the paths created by many algorithms are not rounded and smooth enough, and lack a certain continuity, which is not in line with vessel motion trajectories.

* Corresponding author.
E-mail addresses: pchaysri@cse.uoi.gr (P. Chaysri), cspatharis@cse.uoi.gr (C. Spatharis), kblekas@cse.uoi.gr (K. Blekas), kostaswl@cse.uoi.gr (K. Vlachos).

https://doi.org/10.1016/j.oceaneng.2023.114989
Received 15 December 2022; Received in revised form 24 April 2023; Accepted 27 May 2023
Available online 13 June 2023

Existing trajectory prediction methodologies can be divided into mechanistic and data-driven. Mechanistic are model-based approaches that typically require a set of parameters to be tuned and are not precisely known at prediction time. This reduces their efficiency allowing to be used only for a limited prediction horizon. On the other hand, data-driven methods based on machine learning algorithms are more flexible, powerful and manage to produce ship transitions from state to state through time, achieving trajectory evolution models.

This paper follows a data-driven approach for the navigation of USVs by formulating the problem as an imitation learning problem. Under this prism, the aim is to learn models that imitate expert demonstrations offered by input USV trajectories and spatio-temporal evolution of transitions among states. Combining deep learning techniques with reinforcement learning (RL) has shown promising results. However, there are two well-known issues of Deep RL. From one hand there is the problem of specifying a suitable reward function for the agent to optimize and on the other hand another problem is that of time complexity, i.e. the model requires multiple trial-and-error episodes for learning a satisfactory behavior policy. The advantages of imitation learning is that it enables the agent to mimic an expert policy without usage of reward function, as well as it requires less training time than the state-of-the-art deep RL method, achieving improved performance in the navigation problem and good generalization capabilities. The latter has been also proved and measured in our experimental study.

More specifically, we apply an imitation learning algorithm known as Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016) that allows USVs to directly learn control policies from expert demonstrations in complex environments. Here, the desired trajectories used for the imitation learning process of the agent were created by an "expert" USV based on a virtual potential field technique. It must be noted that the choice of the technique is arbitrary, and any other technique could be used, as it does not affect the proposed imitation learning algorithm.

Our proposed method involves mimicking a predefined route between two ports situated in distinct geographic locations, with known and fixed obstacles. No moving or stationary obstacles along the path were considered. To evaluate the approach's efficacy, we created complex scenarios that incorporated environmental disturbances with varying degrees of variability and erratic behavior. Through these evaluations, we assessed the method's robustness and ability to generate successful trajectories under partially observable conditions, thereby promoting its knowledge reusing capabilities.

It must be mentioned that the scope of this work is not to propose a new navigation method, but rather to describe a method for imitating and to provide a way to replicate any existing method. Ideally, by imitating an existing navigation method, researchers or developers can evaluate its effectiveness in a controlled environment and make possible modifications or improvements if necessary. Imitation learning can also be used to improve the adaptability of USV navigation systems by allowing them to learn from experience and adjust their behavior accordingly. This approach can enable the USV to handle new or unforeseen situations that were not encountered during the training phase. This has the potential to promote effective, and adaptable navigation systems in complex environments.

Furthermore, the advantages of imitation learning is that it enables agent to mimic an expert policy without usage of reward function, as well as it requires less training time than the state-of-the-art deep RL method, achieving improved performance in the navigation problem and good generalization capabilities. The latter has been also proved and measured in our experimental study.

The specific contributions of this work are as follows:

- We are investigating the implementation of the cutting-edge imitation learning algorithm GAIL for navigating unmanned surface vehicles (USVs) in a specified geographic region with realistic environmental disruptions sourced from real-world datasets.
- The agent is trained using a range of values for the direction of disturbances, without any specific assumptions. This approach promotes the development of policies that can more effectively adapt to a range of environmental conditions offering generalization properties and resulting in improved performance.
- Our approach involves implementing the intelligent agent with a continuous action space, without imposing any constraints on the velocity of the USV. The agent's action space comprises both the USV's velocity and the desired heading, with both variables being continuous in nature.
- We conducted evaluations on multiple simulated scenarios that encompassed complex environmental conditions. Moreover, we assessed the efficacy of the agents' learned policies by measuring their capability to transfer their knowledge to new agents in unfamiliar situations, without necessitating retraining.

The structure of this paper is as follows. Section 2 reviews the literature on the research topic of RL based USVs navigation systems and also on imitation learning. Then, Section 3 specifies the problem to be solved and describes the proposed data-driven imitation learning algorithm for modeling USVs trajectories under variable environmental conditions. Finally, Section 4 presents the simulation cases and the obtained results, and Section 5 concludes the paper with findings and future directions.

2. Related work

In the recent literature, there are several methods that have been proposed for marine platform navigation using reinforcement learning (RL) schemes, see for example (Blekas and Vlachos, 2018; Tziortziotis et al., 2018). In the case of USVs, Deep Reinforcement Learning (DRL) approaches are also introduced. For instance, Gonzalez-Garcia et al. (2020) combined DRL algorithms such as the Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2015), with an adaptive sliding mode controller. Specifically, DDPG provides as action the desired heading, while an adaptive sliding mode scheme drives the heading and velocity achieving the USV path-following. Ma et al. (2020), presented a DRL algorithm for collision avoidance between multiple USVs in complex encounter situations, under the rules of International Regulations for Preventing Collision at Sea (COLREGS), which are imposed to the proposed method.

Moreover, Wang et al. (2021a) developed an actor–critic RL scheme to perform trajectory tracking of an unmanned ground vehicle based on optimal control and Wang et al. (2021b) proposed a prior knowledge-based USV RL method for obstacle avoidance in complex environments. In particular, they used an actor–critic architecture along with prior knowledge-based reward shaping for obstacle avoidance. Finally, Holen et al. (2022) studied the problem of autonomous docking and obstacles avoidance using DRL in a boat simulator environment for the development of USVs.

On the other hand, in Inverse Reinforcement Learning (IRL), the agent aims to approximate the underlying reward function based on the expert demonstrations, and then use it to find the optimal policy via RL. Nevertheless, through this process there can be found many reward functions that explain the same optimal policy. To tackle this problem, the maximum entropy principle can be utilized, assuming that the optimal probability distribution should be the one with the highest entropy. In the case of IRL, this means that a policy that imitates the expert state–action distribution must also have the maximum entropy among all policies. However, a RL optimization step is necessary after every update of the reward function, which renders this method inefficient.

The Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016) has been proposed to address the need for approximating a reward function. This method directly learns the expert optimal policy,

Fig. 1. The Pamvotis lake at Ioannina, Greece.

without obtaining the expert reward function, relying on the combination of IRL and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014).

A recent work presented in Vedeler and Warakagoda (2020) used the GAIL framework in order to steer a USV to avoid obstacles using expert demonstrations. Contrary to this work, our proposed method has two principal advantages: (a) the action space consists of the velocity (constant in the former work) and the heading angle of the USV, and (b) the dataset is based on realistic weather conditions of the local area which increases the difficulty. Moreover, in another recent study, Jiang et al. (2022) proposed an extension of GAIL, called GA2IL, in order to train an Autonomous Underwater Vehicle (AUV) agent to follow expert paths. This method builds on standard GAIL with a reward modification that allows the human expert to evaluate the generated trajectories. To the best of our knowledge, there are no other research works for imitation learning on navigating USVs under the presence of environmental disturbances.

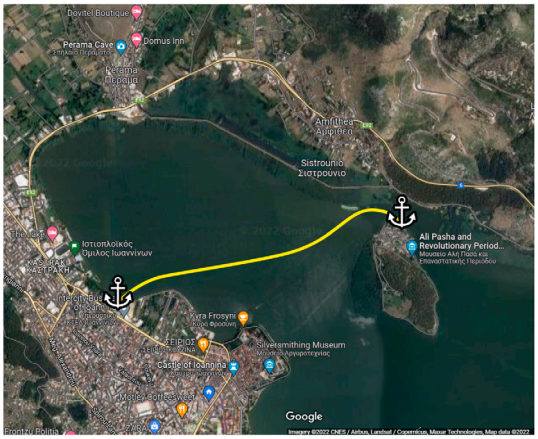


Fig. 2. The Pamvotis lake area and the connection of the ports of Ioannina city and the island by boat.

3. The proposed method

In this section the proposed method is introduced by describing the specific test environment and the way of designing the demonstrated trajectories under disturbances. Our goal is to train an intelligent agent to navigate through the lake, on demonstrated paths that connect two ports, in the presence of known stationary obstacles. This is achieved by using an imitation learning framework that constructs a navigation policy following expert trajectories under various weather conditions. The concepts of imitation learning and the generative adversarial scheme under the umbrella of reinforcement learning are described to further design and implement an efficient USV navigation policy.

3.1. Data acquisition-trajectories construction

The way of obtaining the surface trajectories used for the training and evaluation procedure is described next.

3.1.1. The lake environment

We chose the lake Pamvotis located in the central part of the Ioannina regional unit in north-west Greece, as the main test environment for this work, see Fig. 1. The island in Pamvotis lake is a popular tourist destination and is considered as one of Europe's few inhabited lake islands with no cars. There is a regular boat service from the city of Ioannina to the island that takes around 10 min[1] and (ideally) follows the path shown in Fig. 2 taken by Google maps.

In our study we used an image captured from Google Maps. The selected section of the map covers an area of 2074 × 2074 m. The start position, [166, 166] in m, and goal position, [1659, 1120] in m, are on the bottom left corner and upper right corner respectively, as depicted in Fig. 2.

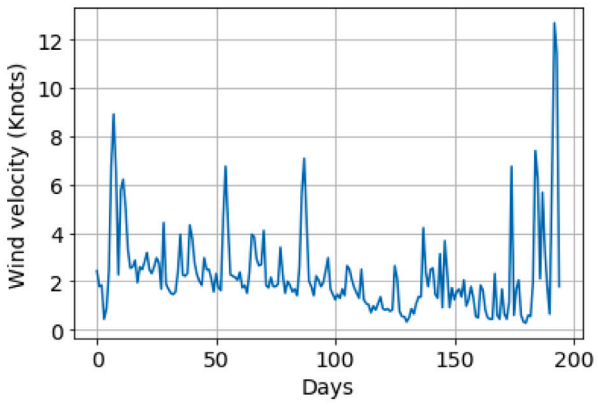
The objective is to implement an appropriate autonomous system by training a RL-based intelligent agent with imitation learning. Agent's learned policy will be used to efficiently control a USV so as to optimally complete the task of reaching the destination port of the island starting from the city port (starting point). The term optimal is referred to the ability of the intelligent agent to establish effective and robust motion planning policies that allow to design suitable paths for quick and efficient trips, under the presence of realistic environmental disturbances of different level and of significantly large variability. The study is focused on the proposed method's ability to imitate the demonstrated trajectories that are available in order to generate physically realizable paths at a reasonable computational cost under its motion constraints and the external disturbances.

We tried to provide a simulated environment tailored to match the specific weather conditions of the local area. For this reason, we used the wind velocity and wind direction information provided by the data from Hellenic Data Service, National Observatory of Athens Institute of Environmental Research and Sustainable Development,[2] for modeling the disturbances. These meteorological data provide several data points from the time period between 2010–2019.

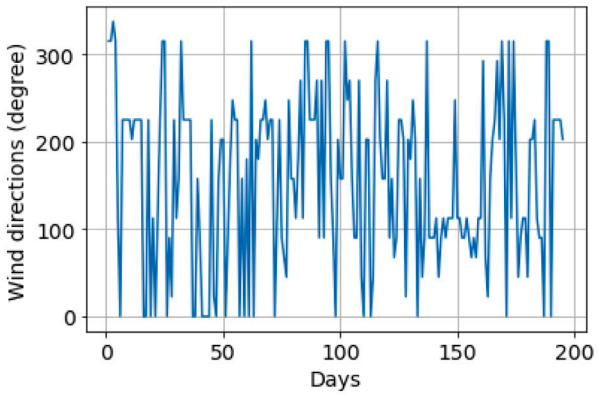
We selected the wind velocity and the wind direction data of the year 2019 as it is the most recent data available to simulate the environment according to the real-world data and to portrait the overall environmental conditions that the USV might encounter. Fig. 3 depicts the average wind velocity and the dominant wind direction recorded between June and December of 2019. It is interesting to observe the

[1] Since the distance between Ioannina and the port of the island is about 2 km, typical boats used in the lake have a mean velocity of 3.3 m/s.

[2] https://data.hellenicdataservice.gr/



(a) Wind velocity



(b) Wind directions

Fig. 3. Average wind velocity and dominant wind directions of the local area around lake Pamvotis during a time period of almost 200 days in the year 2019.

large variability of the wind direction, as well as the mean value of around 4 kn for the wind velocity.

3.1.2. The USv dynamic model

In this work, we consider the model and specifications of a typical USV model, like the WAM-V family,[3] with a twin hull, pontoon style with a length of 7 m, 3.66 m beam, and shallow draft at 0.56 m with a weight of 544 kg. The propulsion is provided by two thrusters, one on each side of the pontoon, and in this work we assume that the thrusters cannot be rotated. Each thruster can apply a maximum allowed force of 2000 N. The choice of the marine vehicle model is arbitrary, and does not affect the imitation learning algorithm.

A short description of the dynamic model of the USV is given next, but a detailed description and analysis of the WAM-V family USVs can be found in Klinger et al. (2013), Klinger et al. (2014), and Sarda et al. (2016).

A three DOF (surge, sway and yaw) dynamic model is used to develop the equations of motion, Fossen (1995). The model designates the origin of the body-fixed frame, $\{B\}$, at the center of gravity and assumes port/starboard symmetry. The equations of motion are the following:

$$\mathbf{M}^{B}\dot{\mathbf{v}} + \mathbf{C}^{B}\mathbf{v} + \mathbf{D}^{B}\mathbf{v} = {}^{B}\boldsymbol{\tau}_{E} + {}^{B}\boldsymbol{\tau} \tag{1}$$

and the kinematics equations of the plane motion are described by

$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\eta)^{B}\mathbf{v} \tag{2}$$

[3] https://wam-v.com/

where $\mathbf{M}$ is the mass and added mass matrix, $\mathbf{C}(\mathbf{v})$ is the Coriolis matrix, $\mathbf{D}(\mathbf{v})$ is the drag matrix, ${}^{B}\boldsymbol{\tau}_{E}$ is the vector of disturbance forces and moment caused by the wind and waves, and ${}^{B}\boldsymbol{\tau}$ is the vector of forces and moment generated by the propulsion system (thrusters), all w.r.t. the body-frame. The vector $\dot{\boldsymbol{\eta}}$ describes the vehicle's North ($\dot{x}$), East ($\dot{y}$) velocities and the angular velocity ($\dot{\psi}$) around the z axis in an inertial reference frame, $\boldsymbol{\eta} = [x, y, \psi]$, and the vector $\mathbf{v}$ contains the vehicle surge velocity (u), sway velocity (v) and yaw rate (r), in the body-fixed frame. $\mathbf{J}$ is the rotation matrix from body-fixed to inertial frame, see Sarda et al. (2016).

The wind induced forces and torque included in vector ${}^{B}\boldsymbol{\tau}_{E}$ are described by the following equations:

$$f_{x,wind} = 0.5 C_X(\gamma_R)\rho V_R^2 A_T \tag{3}$$

$$f_{y,wind} = 0.5 C_Y(\gamma_R)\rho V_R^2 A_L \tag{4}$$

$$n_{z,wind} = 0.5 C_T(\gamma_R)\rho V_R^2 A_L L \tag{5}$$

where C_X and C_Y are force coefficients and C_T is a moment coefficient. These coefficients are functions of the relative angle, γ_R, between the wind and the USV direction, and are taken from tables. ρ is the density of air, A_T and A_L are the transverse and lateral projected areas, and L is the overall length of the USV, see Fossen (1995), and Sarda et al. (2016). V_R is the relative wind speed, given in knots. The wind velocity magnitude and direction are time depended waveforms and, for simulation purposes, are produced by integrating white noise taking under consideration the real data in lake Pamvotis, presented in Fig. 3. In addition, the wave induced forces included in vector ${}^{B}\boldsymbol{\tau}_{E}$ are simulated assuming wind generated waves, where the Pierson Moskowitz wave spectrum was used, see Fossen (1995) and Perez and Blanke (2002). The wave induced forces are calculated using the mean wave drift force equation derived in Faltinsen (1990). Example wind velocities are depicted in Fig. 4. The induces wind and wave forces corresponding to the aforementioned wind velocities and direction are depicted in Fig. 5.

The disturbances of the water current are included in the dynamic equations of motion by representing (1) in terms of the relative velocity between water current and USV. Again, the water current velocity magnitude and direction used in the simulations are produced by integrating Gaussian white noise. Example current velocities are depicted in Fig. 4. A similar approach and a more detailed description can be found in Vlachos and Papadopoulos (2015).

The vector of actuation force and torque, ${}^{B}\boldsymbol{\tau}$, generated by the thrusters is described as:

$${}^{B}\boldsymbol{\tau} = \begin{bmatrix} {}^{B}f_x \\ 0 \\ {}^{B}n_z \end{bmatrix} \tag{6}$$

where f_x and n_z are the force and torque applied w.r.t. the body-frame of the USV.

3.1.3. Constructing the expert trajectories

Next, we need to construct the *expert* trajectories that the agent will learn to imitate. For this purpose we used a virtual potential field approach, Khatib (1985), Choset et al. (2005). We should point out that any other method could be used, and that it is irrelevant for the imitation learning methodology. For example, the desired paths could be alternatively constructed by observing and recording the actual routes of the boats for a sufficient number of days, but it would be time consuming. In our study we assumed the start and goal position of the USV (Ioannina and island port), and the obstacles in the area to be known. The original map presented in Fig. 2 was further converted into binary as shown in Fig. 6(a), where the darker area denotes the land (obstacles) and the lighter area denotes the water surface (free space). The USV is considered to have reached the goal when the distance to the destination point is less than 20 m.

The construction procedure of the trajectories involved three major steps:

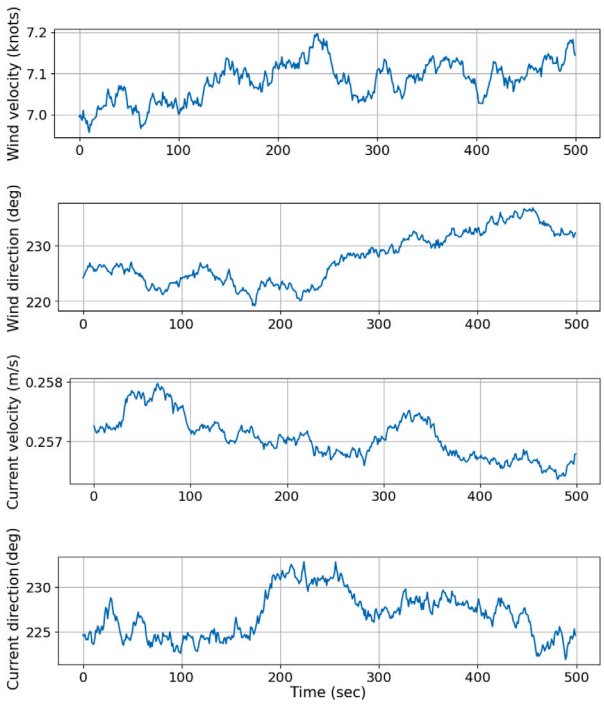


Fig. 4. Example wind and water current velocities.

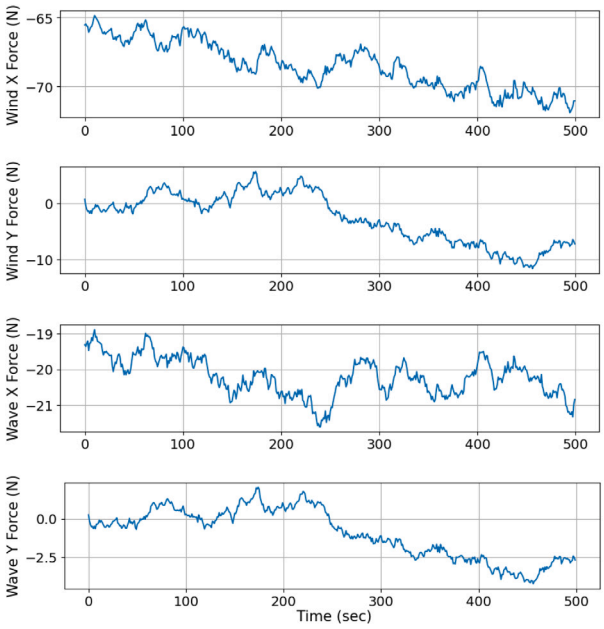


Fig. 5. Induced wind and wave forces corresponding to the wind velocities and directions depicted in Fig. 4.

(a) The first step was the construction of the suitable gradient vector field that directs the robot from the start position to the goal position while avoiding obstacles, resulting in the desired path, see Fig. 6(b).
The start and goal positions are equal to $\mathbf{q}_{start} = [166\ 166]^\top$ and $\mathbf{q}_{goal} = [1659\ 1120]^\top$ respectively as shown in Fig. 6. Virtual obstacles have been placed, in suitable positions on the map, along the coastlines. The number and position of the virtual obstacles is chosen by trial and error, so that the resulted path is about the same as the one followed by the local ferries.

Table 1
List of parameters and their values for constructing the virtual potential field.

Parameter	Symbol	Value
Gain of attractive potential	K_{att}	1450
Gain of repulsive potential	K_{rep}	620
Influence distance of each obstacle	ρ_0	15

A potential function is constructed using the following equation:

$$U(\mathbf{q}) = U_{att}(\mathbf{q}) + U_{rep}(\mathbf{q}) \tag{7}$$

where $U_{att}(\mathbf{q})$ represents the virtual attractive potential to the goal, and $U_{rep}(\mathbf{q})$ represents the sum of the virtual repulsive potentials from each virtual obstacle. Both, are functions of the position $\mathbf{q} = [x\ \ y]$. The potentials are calculated using the following equations:

$$U_{att} = \frac{1}{2} K_{att} (\mathbf{q} - \mathbf{q}_{goal})^2 \tag{8}$$

$$U_{rep} = \sum_{k=1}^{n} U_{rep_k} \tag{9}$$

$$U_{rep_k} = \begin{cases} \frac{1}{2} K_{rep} \left(\frac{1}{\rho_k} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho_k \le \rho_0 \\ 0 & \text{if } \rho_k > \rho_0 \end{cases} \tag{10}$$

where U_{rep_k} represents the virtual repulsive potential from the k virtual obstacle, and K_{att} and K_{rep} are gains used to scale the effect of the attractive and repulsive potentials respectively. ρ_k is the distance from obstacle k. The repulsive effect of obstacles at a distance greater than ρ_0 is ignored.
In theory, the gradient of the potential is a vector that can be viewed as a virtual force acting on the USV, so the negative gradient of $U_{att}(\mathbf{q})$ points to the goal, $\mathbf{q}_{goal}$, while the negative gradient of $U_{rep}(\mathbf{q})$ points away from the obstacles. However, in this work, we view the gradient as desired linear velocity vector, instead of force vector, that the low-level controller has to realize, as described in Section 3.1.4. Hence, The desired linear velocity vector, is equal to

$$\mathbf{v}(\mathbf{q}) = -\nabla U(\mathbf{q}) \tag{11}$$

The parameters are determined by trial and error and are listed in Table 1.

(b) The second step was to let the USV follow the desired linear velocity, from the start position to the goal position, avoiding the physical obstacles, without any environmental disturbances, see Fig. 6(b). To this end, a simple and effective low-level heading/velocity control scheme is employed, as it will be described later in Section 3.1.4.

(c) The third step was to repeat the same procedure under various environmental disturbances (see Fig. 8). This results into the construction of expert trajectories that will be next used as inputs to the imitation learning framework.

The resulting expert trajectories and environmental conditions are used for the training and evaluation procedure of the agent,

3.1.4. Low-level controller

The aim of the controller is to ensure that the USV maintain the desired velocity, i.e. analog to the negated gradient of the potential function, i.e. the surge velocity should follow the desired velocity vector magnitude, and the USV heading should be equal to the desired velocity vector angle as produced by the potential field method.

The control vector includes the force f_x in x_b axis, and the torque n_z about the z_b axis of the USV's body-frame. A velocity controller ensures that the USV's surge velocity is the desired, where the input is the

desired velocity of the USV, i.e. the desired velocity vector magnitude as produced by the potential field method, and the output is the control force according to

$$f_x = K_{p,f}(u_{des} - u) + K_{i,f}\int_0^t (u_{des}(t) - u(t))dt \tag{12}$$

where $K_{p,f}$, and $K_{i,f}$ are the controller gains related to the desired force, u_{des} is the desired surge velocity, and u is the actual surge velocity in x_b axis. The controller gains are equal to, $K_{p,f} = 0.01$, and $K_{i,f} = 0.15$.

A heading controller ensures that the USV's heading is the desired, where the input is the desired orientation of the USV, i.e. the angle of the velocity vector as produced by the potential field method and the output is the control torque according to

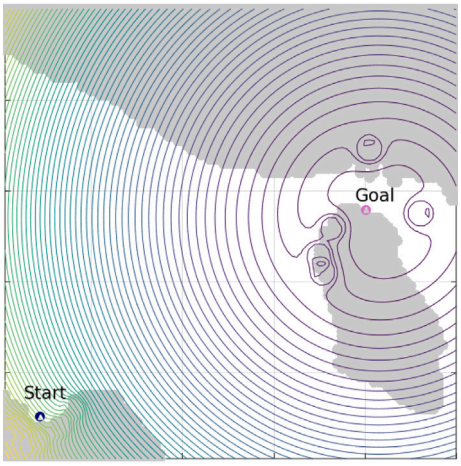
$$n_z = K_{p,n}(\psi_{des} - \psi) + K_{i,n}\int_0^t (\psi_{des}(t) - \psi(t))dt - K_{d,n}\dot{\psi} \tag{13}$$

where $K_{p,n}$, $K_{i,n}$ and $K_{d,n}$ are the controller gains related to the desired torque calculation, ψ_{des} denotes the desired orientation of the USV, i.e. the orientation of the negated gradient vector of the potential function, and ψ is the actual orientation of the USV in every time step. The gain values are the following, $K_{p,n} = 0.0085$, $K_{i,n} = 0.025$ and $K_{d,n} = 0.0003$.

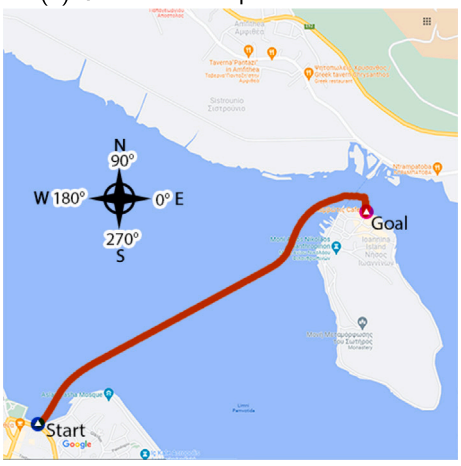
The control force, f_x, and torque, n_z, are implemented by the thrusters of the USV, port and starboard, according to

$$\begin{bmatrix} f_p \\ f_s \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{b} \\ \frac{1}{2} & \frac{1}{b} \end{bmatrix} \begin{bmatrix} f_x \\ n_z \end{bmatrix} \tag{14}$$

where b is the beam of the USV, and f_p and f_s are the actuation forces applied by the USV's port and starboard thruster respectively. The resulting path without any environmental disturbances is depicted in Fig. 6, where the USV reaches the goal avoiding the obstacles following a smooth path. The output thrust per thruster is depicted in Fig. 7. We choose the condition with the highest environmental disturbances at 10 knots wind to convey the efficiency of this low-level controller at its most difficult scenario. We can see that the thrusters are below the maximum allowed thrust (2kN), even in the worst scenario of 10 knot wind.



(a) Contour of the potential function



(b) The expert path that connects both ports

Fig. 6. Constructing the expert path without the presence of any environmental disturbances.

3.2. Generative adversarial imitation learning for USV navigation

In Inverse Reinforcement Learning (IRL), the objective is to deduce a reward function from a set of observed behaviors exhibited by an agent. The reward function serves as a representation of the underlying objective that the agent aims to achieve, and it provides guidance for the agent's decision-making process and behavior.

Generative adversarial imitation learning (GAIL) is an IRL approach that combines the power of Generative Adversarial Networks (GANs) with the imitation learning framework, in order to train an agent to imitate an expert's behavior. To accomplish that, GAIL tries to match the generated state–action distribution with the expert's state–action distribution by minimizing the Jensen–Shannon divergence.

GAIL is rooted in the principle of maximum entropy reinforcement learning, which involves maximizing the entropy of the policy while also ensuring the maximization of expected reward. This approach seeks to balance exploration and exploitation by promoting policy diversity through entropy maximization, while simultaneously optimizing the policy for high expected rewards.

The model consists of two components: a generator and a discriminator. Both networks are trained simultaneously in a zero-sum game, where the goal of the generator is to generate samples that are indistinguishable from the expert demonstrations, and the goal of the discriminator is to correctly identify the expert demonstrations. This process continues until the generator is able to generate samples that are sufficiently similar to the expert demonstrations, at which point the discriminator network will not be able to tell the difference between the two.

Once the generator is sufficiently trained, it can be used to generate new samples, that mimic the expert's behavior. This allows the policy network to learn from the expert demonstrations without directly observing the environment, which can be useful in cases where collecting expert demonstrations is expensive or difficult.

Formally, the objective of GAIL is denoted as:

$$\min_{\pi_\theta}\max_{D} E_{\pi_\theta}[log\,D(s,a)] + E_{\pi_E}[log(1 - D(s,a))] - \lambda H(\pi_\theta) \ , \tag{15}$$

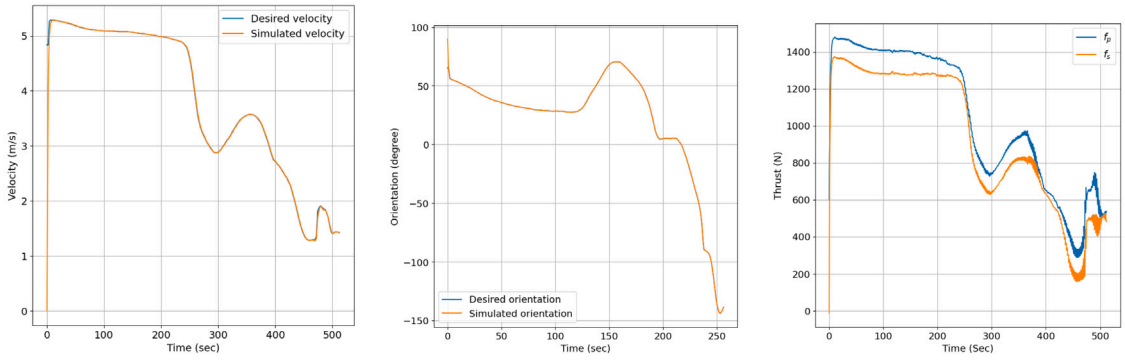
where π_E and π_θ are the expert and generated state–action distributions respectively, D is the discriminative network and $H(\pi)$ is the causal entropy of the policy π_θ, which plays the role of the regularizer. The first term in the objective function encourages the generator to produce trajectories that can be classified as expert-like by the discriminator, while the second term encourages the discriminator to correctly distinguish between expert and generated trajectories. The final term, encourages the generator policy to be diverse.

Finally, GAIL uses a surrogate reward:

$$r = -log\,D(s,a) \ , \tag{16}$$

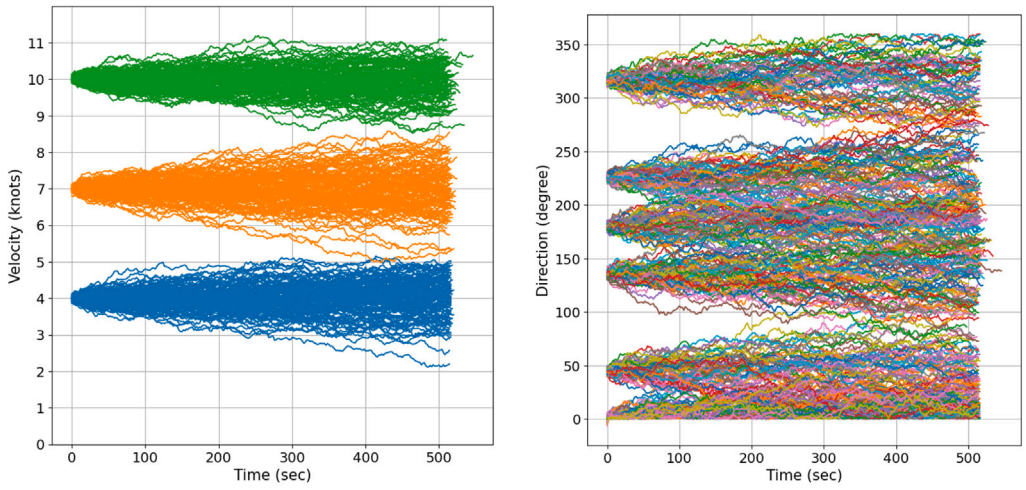
in order to update the policy π_θ, with either Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) or Proximal Policy Optimization (PPO) (Schulman et al., 2017).

In our work, the agent focuses on imitating expert USV trajectories that present similar patterns of navigational behavior. The trajectory,



(a) Output velocity (b) Output orientation (c) Output thrusts

Fig. 7. Simulated velocity, orientation and output thrusts produced by the PI and PID controller in the scenario with 10 knots wind.



(a) wind velocity (b) wind directions

Fig. 8. Demonstrated trajectories of the wind velocity and direction of the scenarios used in our experimental study. They include three different mean wind velocity values of 4 (blue), 7 (orange) and 10 (green) knots, correspondingly. All possible wind directions (b) were considered of range $[0, 360]$ degrees at every scenario of wind velocity (a).

T, is defined as the movement of the USV on the water, which is a chronologically ordered sequence of USV states, containing variables that include the 2D position on x, y axis and the orientation ψ. Moreover, the state space is enriched with environmental disturbances (wind velocity, wind direction, current velocity and current direction). Regarding the action set, A, it is a combination of the desired velocity (u_{des} in m/s) and the desired heading (ψ_{des} in degrees). The execution of the desired action, is then implemented by the low-level controller described in 3.1.4. Algorithm specifies the training process of GAIL for USV navigation.

In the testing phase of the method, the goal is to accurately predict the evolution of the trajectory until the USV reaches the destination port. To that end, we utilize the trained policy of the generator network as follows: Given an initial state from an expert trajectory of the testing set and the weather conditions for that trajectory, the agent rolls-out the whole trajectory. It must be noted that – in order to fairly compare the generated and expert trajectories – the weather conditions under which the expert trajectory was created, should remain the same.

Algorithm 1: GAIL for USV navigation

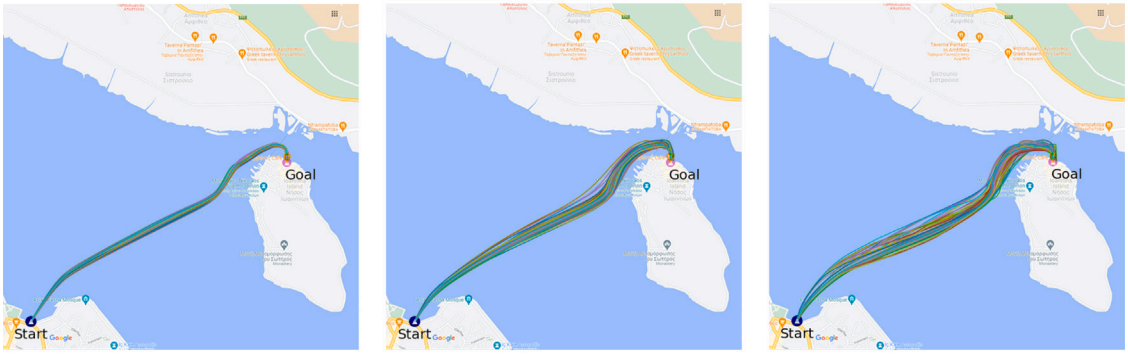
Input: expert trajectories, empty buffer ;
Initialize the policy weights θ using BC ;
for $i = 1, 2, \ldots$ **do**
  while *buffer is not full* **do**
    Sample an initial state from the expert demonstrations;
    Roll-out the trajectory:
      Sample action (u_{des}, ψ_{des});
      Apply action to the low-level controller (eq. (12), (13), (14));
    Store state-action tuples (s, a) to the buffer;
  Update discriminator's parameters with
  $\Delta_w = \mathbb{E}_{x_E}[\nabla_w log(1 - D_w(s,a))] + \mathbb{E}_{x_G}[\nabla_w log D_w(s,a)]$;
  Update θ using TRPO with the surrogate reward:
  $r = -log D_w(s,a)$

4. Simulation studies

We studied the performance of the proposed imitation learning approach using several simulated experiments. The simulation environment includes the kinematic and dynamic model of the USV, and simulated wind, wave, and current disturbances. Moreover, actuation limits on the thrusters are implemented.

The experiments are divided into three (3) scenarios by the intensity of the wind velocity. From the wind velocity data illustrated in Fig. 8, we categorized the intensity into low disturbance starting at 4 knots, medium disturbance at 7 knots and high disturbance at 10 knots. The recorded dominant wind directions show that the wind mostly

(a) Demonstrated trajectories obtained, by the construction procedure in 3.1.3, under various environmental disturbances.



(b) Predicted trajectories by the learned policy of the GAIL agent for every scenario.

(c) Predicted trajectories by the BC method for every scenario.

Fig. 9. Comparison between the demonstrated and the predicted trajectories. They are created under the same environmental disturbances shown in Fig. 8. (a) are the expert trajectories, obtained by the construction procedure described in Section 3.1.3. (b) and (c) are trained using the trajectories from (a) on each scenario.

approaches from the north eastern which translates to 225 degrees direction and the west which translates to 0 degrees direction in our simulation environment. The designated direction is depicted in Fig. 6(b). In order to cover all the wind directions we divided the starting wind directions between 0, 45, 135, 180, 225 and 315 degrees, then subsequently added integration of white noise on each time step. The wind direction white noise parameter is $(\sqrt{0.004} \times \epsilon_1)dt$, where ϵ_1 is normal distribution with $\mu = 0, \sigma = 8, dt = 0.2$ s and the white noise parameter for wind velocity is $(\sqrt{0.1} \times \epsilon_2)dt$, where ϵ_2 is normal distribution with $\mu = 0, \sigma = 0.86$. The simulated wind velocity and direction are shown in Fig. 8.

Following the above procedure, we constructed two sets of 100 expert trajectories each, one set for training the GAIL structure and the other for the agent's evaluation purposes (testing set). It must be noted that any scenario, regardless of the disturbance level (low, medium or high), has the same distribution of wind direction shown in Fig. 8(b) that simultaneously reflects the direction of the wave disturbances. Although, at first sight, this may bring major difficulties in the navigation problem with increased computational complexity, we expect that it will encourage the learning process and enhance the learned policies by incorporating generalization capabilities.

4.1. Experimental design and implementation issues

In this section we provide some implementation details about the architecture of GAIL and the design of the simulations.

The generator network follows an actor–critic architecture, with a policy and a value network. For the policy network, the states are given as input to a fully-connected layer of 100 nodes, followed by another hidden layer of 50 nodes that leads to an output layer of two nodes for estimating the USV velocity and direction that specify the action to be taken. It must be noted that the output of the final layer is the mean of a Gaussian distribution for each action. Moreover, the policy parameters are initialized using Behavioral Cloning in order to minimize the mean squared error between expert and estimated actions, using the Adam optimizer.

On the other hand, the critic and discriminator networks have the same architecture. They consist of two (2) layers of 100 nodes each, and

Table 2
Comparative results of the method in terms of five evaluation measurements in three scenarios.

Scenario	Method	Success rate (%)	$RMSE$ (m)	V (m/s)	T (s)	f_p (N)	f_s (N)
4 kn	Expert	–	–	3.82	514.67	1134.35	1118.39
	GAIL agent	100	14.99	3.93	500.69	1161.12	1142.11
7 kn	Expert	–	–	3.83	513.13	1153.82	1113.43
	GAIL agent	100	17.03	3.88	511.17	1171.19	1121.89
10 kn	Expert	–	–	3.84	509.60	1188.30	1109.37
	GAIL agent	100	23.63	3.87	514.55	1201.33	1106.29

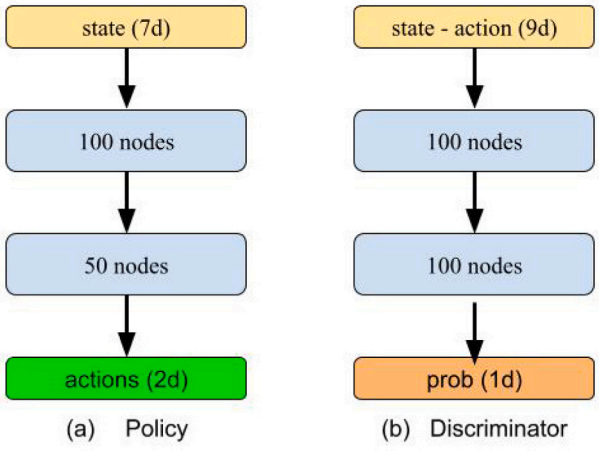


Fig. 10. The architecture of the policy and discriminator networks used in GAIL.

an output layer of one (1) node. In the case of critic, the output node is linear predicting the value of the state, while for the discriminator the output node uses a sigmoid activation function that indicates the probability of a state–action sample being real. The overall architecture of both policy and discriminator networks used in the proposed GAIL framework is presented in Fig. 10.

During training, the number of epochs was set to 500, where in each epoch the policy rolled-out 10000 samples. The collected samples were used in mini-batches of size 1000 for 100 training cycles.

The proposed imitation learning scheme was evaluated regarding its ability to predict the USV trajectories in all scenarios in terms of the following five (5) measurements:

- **Success rate (%)** of the agent calculated as the percentage of its ability to successfully reach the destination port.
- **Root Mean Squared Error (RMSE)** in meters (m) of the distance between the demonstrated by the expert and the generated trajectory using the two spatial dimensions. It must be noted that initially the actual trajectory points are matched to the closest predicted trajectory points, as measured by applying the dynamic time warping (DTW) distance between both trajectories. Then, measurements of error are calculated over the corresponding points.
- **Mean velocity (V)** in m/s of the USV across all trajectory points. We report the mean value of this metric over all trajectories used for evaluation.
- **Mean value of the actuation forces**, f_p and f_s, in N that are applied from the USV's port and starboard thruster (Eq. (14)) in every trajectory point. Again, we report its average over all tested trajectories.
- **Mean duration (T)** in s of all trajectories that take to reach the destination port.

All simulated results reported are the mean values of ten (10) independent runs per case. Finally, our experimental study was exclusively made on a PC with the following specification: AMD Ryzen 5 3600 CPU, 32 GB Ram and GeForce GTX 1660 Super with 6 GB Ram GPU.

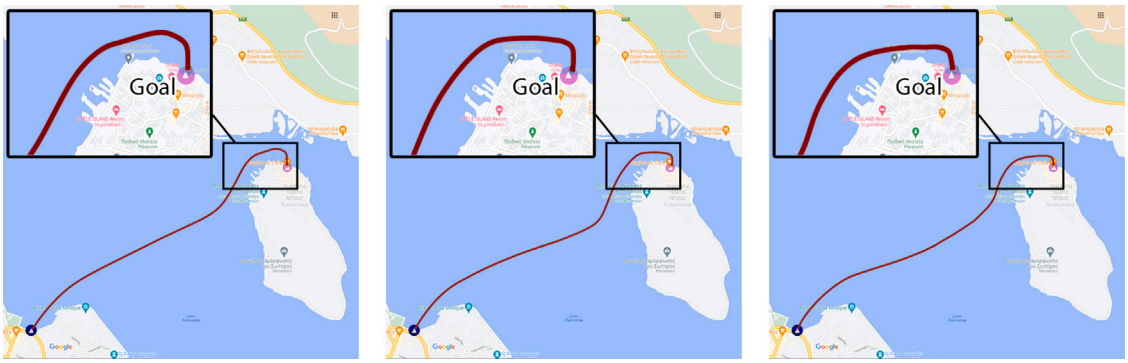
4.2. Results

The efficiency of the proposed approach was evaluated on the testing sets of three scenarios, after training the agents with the corresponding training sets. In all cases we used the Behavioral Cloning (BC) as a baseline and simultaneously as a way for initializing the weights of the policy network. According to the results we received, the performance of the BC was not satisfactory since it could not produce meaningful trajectories and imitate the expert (success 0%). Thus, we did not include its results in the remaining experimental analysis.

Table 2 presents the depicted results in terms of all evaluation criteria suggested. For comparison purposes, we also provide in the same table the values of the measurements (except for success rate) that were calculated by the demonstrated trajectories used for evaluation. As it can be observed, in all cases the trajectories of the USV following the agent's policies with GAIL managed to successfully reach the destination port (100% success), even under the most difficult and unusual weather conditions (i.e., scenario with wind velocity 10kn). Furthermore, the results seem to be intuitively consistent since all rest measurements are in accordance with the difficulty of the scenarios. For example, the predicted trajectories in the scenario with 4 kn wind velocity yielded the best RMSE value (14.99 m) followed by those of the 7 kn scenario (17.03 m) and the 10 kn scenario (23.63 m). We observe also the capability of the method to maintain the same average velocity (around 3.9 m/s) and traveling time (around 500 s) in every level of environmental disturbances. It is worth reminding that the typical boats – under ordinary weather conditions – reach a mean velocity of 3.3 m/s and traveling time of 600 s

Comparing the results between the predicted and the demonstrated trajectories, shown in Table 2, it is interesting to observe the ability of the proposed method to successfully imitate the expert demonstrations and can replicate the tendency of both average actuation forces to three levels of scenarios. However, there is a small difference on the mean velocity (V) and mean duration (T), where it seems that the predicted trajectories are faster mainly in the scenarios of 4 kn and 7 kn. To further investigate this finding, in Fig. 9(b) we plotted the predicted trajectories per scenario and compared them with the demonstrated ones shown in Fig. 9(a). Also, in the same figure we give the corresponding trajectories predicted by the BC approach, see Fig. 9(c). Looking carefully these diagrams we can view the tendency of our method to construct trajectories with less variability and closer to the mean values that constitutes an outcome of the GAIL. We believe that this finding can be also explained by the capability of the learning strategy we followed (use demonstrated trajectories of almost all directions) which significantly increases the robustness of the learned policies. On the other hand the BC completely fails to reach the destination port where either terminates earlier, or follows wrong direction.

It is crucial to emphasize that our work's objective is not to optimize the expert paths, but rather to develop the capacity to imitate them. Our focus is on learning from observed expert behaviors and generating policies that can imitate those behaviors effectively. The simulation



(a) Wind velocity 4kn (b) Wind velocity 7kn (c) Wind velocity 10kn

Fig. 11. Zooming the part of the generated trajectories near the destination port of the island. The ability of creating smooth solutions is obvious in all scenarios.

Table 3
The performance of the agent's learned policies of 7 kn wind velocity in unknown scenarios considering two different wind velocity values, 4 kn and 10 kn.

Wind velocity	Wind direction (degrees)			
	0	45	180	225
4 kn	100%	100%	100%	100%
10 kn	70%	60%	65%	90%

results obtained thus far indicate promising success in achieving this goal. By prioritizing imitation over optimization, we aim to leverage the expertise of human operators and transfer their skills to autonomous systems in a safe and efficient manner.

Finally, Fig. 11 shows examples of the generated trajectories in three different environmental cases. Obviously the learned policy is able to generate smooth and dynamically feasible trajectories that reflects the method ability to offer robust and accurate navigation solutions and smoother journeys to the island independent on the level of environmental disturbances. This can be seen clearly by taking a closer look to the part of the generated trajectories towards the goal (destination port), as shown in Fig. 11.

4.3. Knowledge reusing

One of the key challenges in designing intelligent agents to a task is to alleviate the burden of learning and allow the exploitation, sharing and *reusing of the knowledge* generated throughout decision-making process (Taylor and Stone, 2009; Lazaric, 2012). Transfer learning focuses on storing obtained knowledge from the solution of one problem and applying it to a different but related problem. It can significantly reduce learning time and create more solid intelligent agents. Knowledge reuse becomes a core technology in agent-based learning systems that can establish relationships with other agents that allow implicit or explicit knowledge sharing, and integrate the received information with its previous experience to improve learning.

In our study, we tried to investigate the capability of the proposed method to offer knowledge reusing for USV autonomous navigation tasks. Knowledge is offered through the learned policies of the RL agent which can be (re)used in unknown environments. More specifically, we have taken as basis the agent's policies that have been learned by the scenario of wind velocity equal to 7 kn. To measure their effectiveness, we have tested them to different (unknown) scenarios created using wind velocity equal to 4 kn (small level of disturbances) and 10 kn (large level of disturbances). A number of 100 test trajectories were created, by the construction procedure as described in 3.1.3, for any scenario, and all learned policies of 7 kn scenario were applied for reconstructing them starting from the same initial point and having the same environmental conditions.

Table 3 gives the results where we report the ability of policies to successfully reach the destination (port of island) in terms of the percentage of the success. An interesting observation concerns the ability of all agents to not decrease their performance when using scenarios with smaller wind velocity (4 kn) reaching always the destination port. This shows the capability of the method to efficiently offer knowledge reusing and maintain its decision-making policies to unknown environments. The percentage of success becomes lower when examining scenarios of larger wind velocity (10 kn) as shown in Table 3 with a success rate of more than or equal to 60%. The best behavior (90%) was obtained in the case of the direction of 225 degrees even if this is opposite to the main boat travel direction from the initial port. As shown in Fig. 9(b) that illustrates the predicted trajectories, the ship must turn a half-circle around the top of the island in order to dock successfully. As a result, this will change the orientation of boat and now the wind direction would be relatively modified for the motion. Thus, the direction of 225 degrees will be favorable for approaching the destination port, while the direction of 0 degrees may get the boat away from the destination port. When the wind velocity is strong, counteracting the forces of the wind become a difficult task.

5. Conclusions

In this work, we presented a USV navigation system based on GAIL which learns a policy that imitates expert trajectories. This approach trains the policy to map states to desired velocity and heading angles. We tested our system on three (3) scenarios of increased difficulty, where the agents showed their capability to produce realistic trajectories for the USV under diverse environmental disturbances. We further evaluate the agents' learned policies and measure their knowledge reusing capability where we displayed a way of transferring the agent's knowledge to unknown scenarios.

Based on the encouraging results, there are a number of directions in which we plan to extend our work:

- Consider more complex scenarios with
  - trajectories of multi-modal behaviors and/or of longer distance,
  - moving or (unknown) stationary obstacles and alternative collision avoidance algorithms during training, and
  - marine traffic in the port environment.
- Study alternative schemes of learning algorithms (such as Offline RL) to imitate expert trajectories.

CRediT authorship contribution statement

Piyabhum Chaysri: Conceptualization, Methodology, Software, Writing – original draft. **Christos Spatharis:** Conceptualization,

Methodology, Writing – original draft. **Konstantinos Blekas:** Supervision, Conceptualization, Methodology, Writing. **Kostas Vlachos:** Supervision, Conceptualization, Methodology, Writing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Bai, X., Li, B., Xu, X., Xia, Y., 2022. A review of current research and advances in unmanned surface vehicles. J. Mar. Sci. Appl. 21, 47–58.

Blekas, K., Vlachos, K., 2018. RL-based path planning for an over-actuated floating vehicle under disturbances. Robot. Auton. Syst. 101, 93–102.

Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., 2005. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press.

Faltinsen, O.M., 1990. Sea Loads on Ships and Offshore Structures / O.M. Faltinsen. Cambridge University Press Cambridge, New York, URL http://www.loc.gov/catdir/toc/cam031/90043346.html.

Fossen, T., 1995. Guidance and Control of Ocean Vehicles. Wiley, New York, NY.

Gonzalez-Garcia, A., Castañeda, H., Garrido, L., 2020. USV path-following control based on deep reinforcement learning and adaptive control. In: Global Oceans 2020: Singapore – U.S. Gulf Coast. pp. 1–7. http://dx.doi.org/10.1109/IEEECONF38699.2020.9389360.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. Adv. Neural Inf. Process. Syst. 27.

Ho, J., Ermon, S., 2016. Generative adversarial imitation learning. In: Advances in Neural Information Processing Systems (NIPS). 29.

Holen, M., Ruud, E.-L.M., Warakagoda, N.D., Goodwin, M., Engelstad, P., Knausgård, K.M., 2022. Towards using reinforcement learning for autonomous docking of unmanned surface vehicles. In: Engineering Applications of Neural Networks. Springer International Publishing, pp. 461–474.

Jiang, D., Huang, J., Fang, Z., Cheng, C., Sha, Q., He, B., Li, G., 2022. Generative adversarial interactive imitation learning for path following of autonomous underwater vehicle. Ocean Eng. 260, 111971.

Khatib, O., 1985. Real-time obstacle avoidance for manipulators and mobile robots. In: 1985 IEEE International Conference on Robotics and Automation. pp. 500–505.

Klinger, W.B., Bertaska, I., Alvarez, J., von Ellenrieder, K.D., 2013. Controller design challenges for waterjet propelled unmanned surface vehicles with uncertain drag and mass properties. In: 2013 OCEANS - San Diego. pp. 1–7. http://dx.doi.org/10.23919/OCEANS.2013.6741200.

Klinger, W.B., Bertaska, I.R., von Ellenrieder, K.D., 2014. Experimental testing of an adaptive controller for USVs with uncertain displacement and drag. In: 2014 Oceans - St. John's. pp. 1–10. http://dx.doi.org/10.1109/OCEANS.2014.7003032.

Lazaric, A., 2012. Transfer in reinforcement learning: A framework and a survey. In: Reinforcement Learning. Adaptation Learning and Optimization. Vol. 12, pp. 143–173.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N.M.O., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. CoRR arXiv:1509.02971.

Liu, Z., Zhang, Y., Yu, X., Yuan, C., 2016. Unmanned surface vehicles: An overview of developments and challenges. Annu. Rev. Control 41, 71–93.

Ma, Y., Zhao, Y., Wang, Y., Gan, L., Zheng, Y., 2020. Collision-avoidance under COLREGS for unmanned surface vehicles via deep reinforcement learning. Marit. Policy Manag. 47 (5), 665–686. http://dx.doi.org/10.1080/03088839.2020.1756494.

Perez, T., Blanke, M., 2002. Simulation of ship motion in seaway. Computer Science; Technical Report, the University of Newcastle, Callaghan, Australia, pp. 1–13.

Qiao, Y., Yin, J., Wang, W., Duarte, F., Yang, J., Ratti, C., 2022. Survey of deep learning for autonomous surface vehicles in the marine environment.

Rahman, A., Hamid, U.Z., Chin, T.A., 2017. Emerging technologies with disruptive effects: a review. Perintis e-Journal 7 (2), 111–128.

Sarda, E.I., Qu, H., Bertaska, I.R., von Ellenrieder, K.D., 2016. Station-keeping control of an unmanned surface vehicle exposed to current and wind disturbances. Ocean Eng. 127, 305–324. http://dx.doi.org/10.1016/j.oceaneng.2016.09.037.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P., 2015. Trust region policy optimization. In: Proceedings of the 32nd International Conference on Machine Learning. Vol. 37, pp. 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. CoRR.

Taylor, M.E., Stone, P., 2009. Transfer learning for reinforcement learning domains: A survey. J. Mach. Learn. Res. (JMLR) 10, 1633–1685.

Tziortziotis, K., Tziortziotis, N., Vlachos, K., Blekas, K., 2018. Motion planning with energy reduction for a floating robotic platform under disturbances and measurement noise using reinforcement learning. Int. J. Artif. Intell. Tools (IJAIT) 27 (4).

Vedeler, A., Warakagoda, N., 2020. Generative adversarial imitation learning for steering an unmanned surface vehicle. In: Proceedings of the Northern Lights Deep Learning Workshop.

Vlachos, K., Papadopoulos, E., 2015. Modeling and control of a novel over-actuated marine floating platform. Ocean Eng. 98, http://dx.doi.org/10.1016/j.oceaneng.2015.02.001.

Wang, N., Gao, Y., Zhao, H., Ahn, C.K., 2021a. Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle. IEEE Trans. Neural Netw. Learn. Syst. 32 (7), 3034–3045. http://dx.doi.org/10.1109/TNNLS.2020.3009214.

Wang, W., Luo, X., Li, Y., Xie, S., 2021b. Unmanned surface vessel obstacle avoidance with prior knowledge-based reward shaping. Concurr. Comput.: Pract. Exper. 33 (9), http://dx.doi.org/10.1002/cpe.6110.

Piyabhum Chaysri received B.Sc. in Computer Science & Engineering and M.Sc. degree in the area of Navigation of Robotics platform using Reinforcement Learning from the University of Ioannina, Greece in 2016 and 2018 respectively. He is now a Ph.D. student of the University of Ioannina. Research interests: navigation and control of robotic platforms, autonomous systems, design and construction of robotic mechanisms, human–robot interface, human assistant robotic systems.

Christos Spatharis acquired his B.Sc. in Computer Science & Engineering and his M.Sc. in Technologies – Applications from the University of Ioannina. He is currently a PhD Student in the same department. During his academic career so far, he took part in the "DART" research program, the "Data-Driven Trajectory Imitation with Reinforcement Learning" project and is currently working under "Combining Simulation Models and Big Data Analytics for ATM Performance Analysis (SIMBAD)" project. Moreover, he participated in the publication of eight (8) research papers and presented four (4) of them at SETN(2018), IISA(2019), SETN(2020) and ITSC(2020) conferences. Research Interests: Artificial Intelligence, Machine Learning, Deep Reinforcement Learning, Autonomous Agents, Multi-Agent Systems, Collaborative Environments, Robotic Applications.

Konstantinos Blekas received the Diploma degree in Electrical Engineering in 1993 and the Ph.D. degree in Electrical and Computer Engineering in 1997, both from the National Technical University of Athens. He is currently full professor at the Department of Computer Science and Engineering, University of Ioannina, Greece. He has co-authored more than 90 refereed journal and conference articles. His research interests include machine learning, pattern recognition, intelligent agents and reinforcement learning with applications to robotics and autonomous systems, computer vision, and medicine. He has served as a co-chair of the 8th Hellenic Conference of Artificial Intelligence, SETN 2014, as a member of program committee member on numerous conferences, including AAAI, NIPS, ECML/PKDD, ECAI, IJCAI, AAMAS, ICTAI and CVPR, as well as reviewer in well-respected journals for artificial intelligence, machine learning and intelligent systems. Details on his publications, work and academic activities are provided at http://www.cs.uoi.gr/~kblekas/.

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