frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 20 September 2021
doi: 10.3389/frobt.2021.739023

Robust ASV Navigation Through Ground to Water Cross-Domain Deep Reinforcement Learning

Reeve Lambert[1], *Jianwen Li*[2], *Li-Fan Wu*[2] *and Nina Mahmoudian*[3]*

[1]MS Student, School of Mechanical Engineering, Purdue University, West Lafayette, IN, United States, [2]PhD Student, School of Mechanical Engineering, Purdue University, West Lafayette, IN, United States, [3]Associate Professor, School of Mechanical Engineering, Purdue University, West Lafayette, IN, United States

This paper presents a framework to alleviate the Deep Reinforcement Learning (DRL) training data sparsity problem that is present in challenging domains by creating a DRL agent training and vehicle integration methodology. The methodology leverages accessible domains to train an agent to solve navigational problems such as obstacle avoidance and allows the agent to generalize to challenging and inaccessible domains such as those present in marine environments with minimal further training. This is done by integrating a DRL agent at a high level of vehicle control and leveraging existing path planning and proven low-level control methodologies that are utilized in multiple domains. An autonomy package with a tertiary multilevel controller is developed to enable the DRL agent to interface at the prescribed high control level and thus be separated from vehicle dynamics and environmental constraints. An example Deep Q Network (DQN) employing this methodology for obstacle avoidance is trained in a simulated ground environment, and then its ability to generalize across domains is experimentally validated. Experimental validation utilized a simulated water surface environment and real-world deployment of ground and water robotic platforms. This methodology, when used, shows that it is possible to leverage accessible and data rich domains, such as ground, to effectively develop marine DRL agents for use on Autonomous Surface Vehicle (ASV) navigation. This will allow rapid and iterative agent development without the risk of ASV loss, the cost and logistic overhead of marine deployment, and allow landlocked institutions to develop agents for marine applications.

Keywords: autonomous surface vehicle (ASV), navigation and control, reinforment learning, autonomous vehicle navigation, marine robot navigation, cross-domain deep reinforcement learning

OPEN ACCESS

Edited by:
Alberto Quattrini Li,
Dartmouth College, United States

Reviewed by:
Navid Nourani-Vatani,
FleetSpark, Germany
Yugang Liu,
Royal Military College of Canada, Canada

***Correspondence:**
Nina Mahmoudian
ninam@purdue.edu

Specialty section:
This article was submitted to Robotic Control Systems, a section of the journal Frontiers in Robotics and AI

Received: *09 July 2021*
Accepted: *24 August 2021*
Published: *20 September 2021*

Citation:
Lambert R, Li J, Wu L-F and Mahmoudian N (2021) Robust ASV Navigation Through Ground to Water Cross-Domain Deep Reinforcement Learning. Front. Robot. AI 8:739023. doi: 10.3389/frobt.2021.739023

1 INTRODUCTION

Mobile robots are emerging as powerful tools for scientific exploration in response to societal needs. Robots such as Autonomous Surface Vehicles (ASV), Autonomous Ground Vehicles (AGV), and Unmanned Aerial Vehicles (UAV) have pushed the boundaries of autonomous activity in the water, ground, and air domains respectively. Many applications of autonomous vehicles are mission specific and require retasking, recharging, and reprogramming between missions or activities. For a system to be truly autonomous it must be able to solve navigational problems and avoid obstacles by thinking, planning, and acting and do so across different missions and environments without human intervention. This issue is present across all domains where mobile autonomous robots are deployed.

ASVs are no exception to the rule. While ASVs have been deployed on missions requiring complex navigation such as monitoring coastal ecosystems (Nicholson et al., 2018), exploring coastal waterways (Han et al., 2019), and inland waterways (Karapetyan et al., 2021) they struggle to generalize between these environments and require retasking between deployments in dynamic, rapidly changing, and challenging environments. In general, ASVs have historically struggled to complete missions in such environments like rivers and obstacle ridden littoral waters. For example, an ASV traversing inter-tidal zones will require the ability to sense and rapidly react to obstacles that are exposed with an outgoing tide that it did not experience traversing the area during high-tide. This requires that a vehicle have the ability to robustly process high dimensional and complex environmental data to rapidly make decisions to avoid unknown obstacles that exist in or enter a mission area.

Recent advances in computational power, sensor quality, and algorithms have enabled the application of Deep Learning (DL) to solve the navigation and obstacle avoidance problem in dynamic environments (Giusti et al., 2016; Kahn et al., 2017; Woo and Kim, 2020). A common approach for training DL models is through supervised learning, where a model is fitted to a set of inputs and desired outputs either by classification or regression on predetermined data sets. However, this requires existing data of vehicle navigation. Unlike supervised learning methods, Reinforcement Learning (RL), specifically Deep Reinforcement Learning (DRL), was developed to enable an agent to continuously improve itself through experienced interactions with its environment and thus does not need preexisting datasets.

Large strides in DRL have been realized over the last decade with breakthroughs in an agent's ability to learn and complete complex tasks such as playing video games (Bellemare et al., 2013), playing board games such as Chess and Go (Silver et al., 2018), and robot body simulations (Todorov et al., 2012). Recently work has shifted towards applying DRL to autonomous vehicle control in real-world situations. The application of deep reinforcement learning to mobile robot navigation and obstacle avoidance has been explored for air (Singla et al., 2021), ground (Lei et al., 2018), and water domains (Cheng and Zhang, 2018; Woo and Kim, 2020; Zhou et al., 2020).

DRL implementations require a significant amount of experiential learning to generalize, which creates a training bottleneck in challenging domains such as water surface and subsea. Access to such domains for training can be exceedingly expensive, logistically challenging, require extensive human presence, be weather dependant, and be potentially dangerous for the vehicle. In contrast, deployment of AGVs and UAVs is comparatively easy, inexpensive, and poses little hazard to vehicles. This has lead to a disparity in data quantity (dataset sizes) and quality (annotated). Many large publicly available datasets are available for terrestrial applications (Cordts et al., 2016; Kellenberger et al., 2018; Perera et al., 2018), while only smaller and more sparse datasets are available for the marine environment (Bovcon et al., 2019; Taipalmaa et al., 2019). Leveraging data rich environments to train DRL agents for data poor environments provides an opportunity to drastically cut developmental times and costs.

Despite the promising application of deep reinforcement learning to autonomous navigation problems, there is limited real-world implementation, especially in the marine environment. The lack of real-world implementation is due to multiple factors, in particular: 1) the significant environmental interaction data required for agent generalization is hard, expensive, and logistically challenging to collect, 2) the constraints of agent training in environments where failure can be detrimental to the vehicle, and 3) implementations can suffer from differences in agent prediction, and vehicle actuation frequencies (Dulac-Arnold et al., 2019).

This paper proposes a methodology for building and implementing well studied DRL agents into mobile robots in such a way that the agent can solve navigation problems and generalize across domains by 1) using an action space that is present in all domains, 2) integrating with a vehicle at a high level through path planning by generating waypoints, and 3) leveraging low-level control for vehicle dynamics and actuation. Following such a methodology that separates the DRL agent from robotic dynamics, an agent can generalize across simulation to real and real to real domain shifts utilizing inexpensive Commercial Off-The-Shelf (COTS) vehicles for initial risky training, can predict at a slower rate than vehicle actuation, and can use accessible domains such as ground to complete training before implementation in hazardous and costly domains such as water. To test the proposed method's ability to allow cross-domain generalization, a simple Deep Q Network (DQN) is assembled and trained to avoid obstacles on an AGV in a virtual environment and then implemented without alteration or retraining in a simulated water surface environment, on a real-world AGV, and on a real-world ASV for further validation, **Figure 1**. Both real-world robotic platforms are low-cost COTS systems that operate autonomously using an in-house hardware and software setup developed for multi-domain use.

The success of the proposed methodology will help alleviate domain disparities and decrease the time required for DRL agents to generalize on ASVs by enabling ground and virtual training to be directly applied to the water domain. Knowing that agents trained in simulation and on the ground have a high likelihood of real-world generalization enables confidence in rapid DRL agent iteration on said domains. Lastly, utilizing this methodology landlocked institutions can develop autonomous agents for deployment in marine domains without the need for relocation to coastal areas.

The remainder of this work covers a review of related RL implementations for obstacle avoidance and other related cross-domain ML work in **Section 2**. Theoretical review of deep neural networks and reinforcement learning methodologies utilized are presented in **Section 3**. The methodology utilized in DRL agent implementation, the software and hardware package utilized in real-world testing, and a description of the robotic platforms used is detailed in **Section 4**. The results of agent-testing in simulated and real-world environments are presented in **Section 5**. Finally a

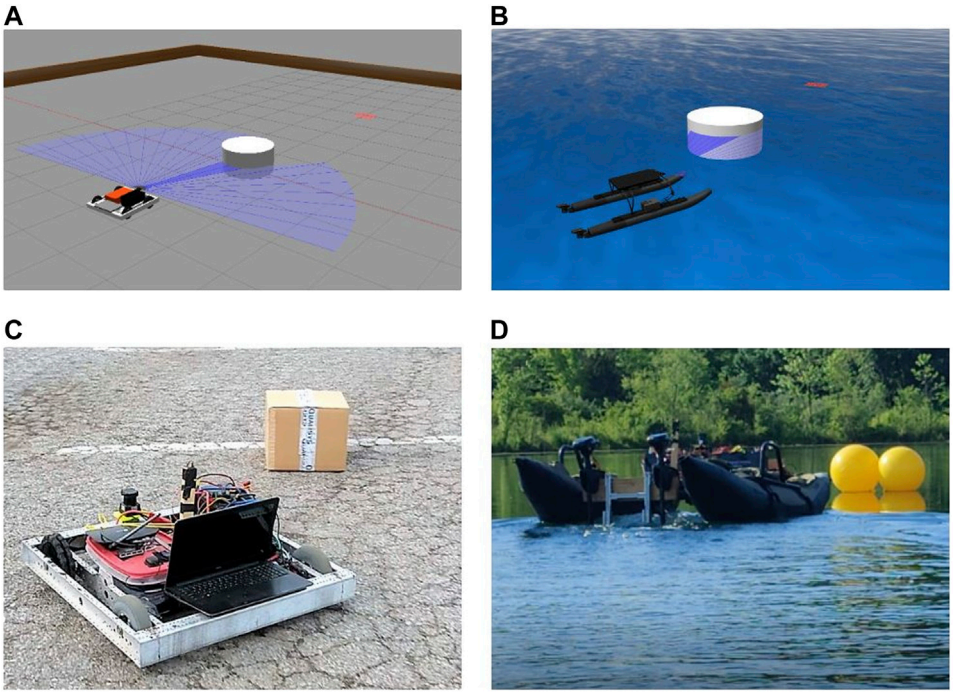

FIGURE 1 | Trained DQN model performing cross-domain obstacle avoidance during **(A)** an AGV simulation, **(B)** ASV simulation, **(C)** AGV real-world mission, and **(D)** ASV real-world mission. All images show execution of the same trained DRL agent.

conclusion of the paper and future work are presented in **Section 6**.

2 RELATED WORK

Traditional methods for navigation, obstacle avoidance, and control such as SLAM (Yu et al., 2018), multi-layer controllers (Ferreira et al., 2007), PID control (Caccia et al., 2008), and integral line-of-sight (Fossen et al., 2015) have been extensively studied. While these methods generalize well across domains, customized implementations are required for different vehicles (dynamic differences) and missions (environmental differences). Traditional navigation and control strategies are able to perform adequately in their design scenarios but after customization encounter issues in dynamic environments, incursions by other agents, and upon encountering random obstacles (Gupta, 2013).

Recently there has been increasing effort to solve the issues that traditional controllers experience in dynamic environments with machine and reinforcement learning. Both supervised machine learning and reinforcement learning have typically learned control policies that govern vehicle actuation based on high dimensional sensor data (Mancini et al., 2017; Zhu et al., 2017; Woo and Kim, 2020). These types of ML agents utilize an observation space encompassing the vehicles location, orientation, and its surrounding environment and typically an action space of vehicle actuation commands such as yaw rates and translational speed. Such implementations are considered to be End-to-End.

End-to-End ML and RL implementations think, plan, and act. An agent interfaces with all levels of control. This methodology constrains a trained agent to the specific vehicle it was trained on as it has learned not only how to avoid obstacles and adapt to a changing environment, but also a vehicle's unique specific dynamics (Cheng and Zhang, 2018; Codevilla et al., 2018; Zhang et al., 2020; Zhou et al., 2020). These implementations would likely struggle generalizing in any cross-vehicle or cross-domain implementation without re-training of the agent.

Some work on DRL implementation has been done in a way that leverages traditional navigational controllers and trains an agent in only the think and plan aspects of autonomous control. Implementations can have the DRL agent switch between path planners to avoid discrete obstacle avoidance scenarios (Woo and Kim, 2020), utilize large action spaces to add 3-D path sub-tasks for tracking (Sun et al., 2020), and utilize higher level Convolution Neural Networks (CNN) to prescribe an observation space for path DRL agents (Yan et al., 2019). These works provide methodologies for indirectly altering the path of a vehicle using typical low-level control, thus providing an agent which could generalize across vehicles. However, they are not made for and are not built to specifically facilitate generalization across domains.

Cross-domain training is not new to the machine learning field. There are many examples of using data gathered in one domain to train an agent designed for implementation in another environment. For example, images and video can be recorded from a car or bike and used in off-line training of convolution neural networks and image recognition nets for a drone

(Loquercio et al., 2018). Furthermore, significant work has been done to allow training of agents in a simulated environment with the goal of implementing a trained agent in the real-world (Zhu et al., 2017; Cheng and Zhang, 2018; Osinski et al., 2020; Zhou et al., 2020). However, such an approach typically suffers from the shift between simulation and reality and may still require data gathering in the real-world for full generalization (Zhu et al., 2017).

In addition to cross-domain training, generalizing neural networks between simulation and real-world domains has seen significant work recently under the broad ideology of sim2real. Sim2real implementations do this through training neural networks in randomly generated simulated environments (Tobin et al., 2017), approximating real-world dynamics and textures (Kaspar et al., 2020), or learning high-level control policies (Hong et al., 2018; Müller et al., 2018). Unlike this work, such implementations are typically either end-to-end, or take place in environments with well established environmental dynamics and are used to generate mission paths in a local space rather than augment traditionally globally generated ones.

In this work, a neural network implementation methodology is proposed whereby any high-level dynamic-independent neural agent can be trained in simulation and readily accessible real-world areas (ground and air) and then integrated into existing and proven marine systems (surface and subsea) without altering low level systems. The goal is not to provide another approach to either create realistic simulations that approximate the variances of real-world interaction or generalize neural networks from simulation to the real world through end-to-end visually identified waypoint navigation. Integrating into existing and expensive marine systems to extend their capabilities without training in real and dangerous environment saves time and protects robotic hardware. This work uses the implementation of a DQN trained to avoid obstacles in simulation, on an AGV, and ASV as a case study to prove the viability of the methodology.

3 REINFORCEMENT LEARNING AND DEEP Q NETWORK IMPLEMENTATION

The problem of sensing and avoiding obstacles can be represented as a Markov Decision Process (MDP). Q-Learning is a form of reinforcement learning that maps the vehicle's reward through all possible state transitions and actions in such a Markov Decision Process. Deep Q-Learning Networks (DQN) (Mnih et al., 2015) are a form of Q reinforcement learning that utilize Q-Learning and Deep Q-Networks to learn the optimal Q-value function and solve the MDP problem in a stable fashion. Policy-based methods like policy gradient (Silver et al., 2014) solve the MDP problem by finding the optimal policy. In this paper a static obstacle avoiding DQN is used to study and demonstrate the ability of a neural net trained in one domain to be implemented and validated for functionality in another domain. Policy gradient methods require trajectories generated by the current policy, thus are more sample inefficient and are require more training steps to converge and were thus not utilized for implementation in this work. Furthermore, While more advanced DQN implementation strategies such as Double DQN (van Hasselt et al., 2016), Dueling DQN (Wang et al., 2016), and Rainbow (Hessel et al., 2018) exist, a simple DQN was utilized for a proof of concept of cross-domain use on mobile robots.

In the DQN (Mnih et al., 2015), each action (a) enacted at an agent's current state (s) has an unknown probability to go to a new state (s') that has associated rewards (positive feedback) and penalties (negative feedback). This positive and negative reinforcement of the action leading to the realized state is called the reward function [$R(s, a, s')$] and is calculated with respect to the old and new state based on a user defined system that incorporates states to achieve (objectives) and states to avoid (obstacles). A very basic example reward function is shown in **Eq. 1** and shows the reward for transitioning from state s to state s' through action a.

$$R(s,a,s') = \begin{cases} +50 & \text{(Goal reward)} \\ -100 & \text{(Collision penalty)} \\ R_\theta R_d & \text{(Position reward)} \end{cases} \tag{1}$$

R_θ and R_d are the heading reward and distance reward respectively and are defined in **Eq. 2**. θ is the yaw offset from the goal, D_c is the current distance to the goal, and D_g is the initial distance to the goal.

$$R_\theta = 5\left(1 - \frac{2}{\pi}|\theta|\right) \tag{2}$$

$$R_d = 2^{\frac{D_c}{D_g}} \tag{3}$$

Each such unique transition and reward of all future transitions is called a Q-value and can be calculated iteratively using the Q-Value Iteration Algorithm in **Eq. 4** (Aurelien, 2019).

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \times [R(s,a,s') + \gamma \max_{a'} Q_k(s',a')] \quad \text{for all } (s',a) \tag{4}$$

Where $T(s, a, s')$ is the unknown probability of achieving a new state (s') from an old state (s) through action (a), γ is a discount factor to discount potential future rewards caused by the current state transition, and a' is the optimal action to take after achieving s'. By calculating the potential vehicle reward of every possible Q-value, an optimal vehicle policy simply selects the state action pair (s, a) that deliver the highest possible sum of all discounted future rewards and enacts it. The learning of the unknown probabilities occurs during vehicle deployment as the vehicle either enacts its optimal policy or randomly explores state action pairs and then chooses between the two with an epsilon (ϵ) greedy policy.

While Q-learning creates a very stable prediction agent for learning completion, it is impractical in real-world applications with high numbers of state action pairs (Aurelien, 2019). Thus, to improve real-world applications a Deep Neural Network is used as a function approximator [$Q_\theta(s, a)$] to give a Q-value for any input state action pair, such a Deep Neural Network is referred to as a DQN. Thus the goal of agent training is to recursively train the neural network to accurately approximate Q-values.

$$Q_{target}(s,a) = r + \gamma \cdot \max_{a'} Q_\theta(s,a) \tag{5}$$

During agent operation the DQN follows the same principles as Q-Learning: it observes the state, chooses an optimal or uniform random action as dictated by ϵ greedy policy, transitions to the next state, and receives a reward. The agent stores its interactions with the environment in an experience replay buffer. During training, these interactions—in the form of $[s, a, s', R(s, a, s')]$–are randomly extracted in batches and replayed for the DQN, which predicts the reward of the state action pair $[Q_\theta(s, a)]$. With the predicted value and a target reward (**Eq. 5**, with r being recorded rewards) Mean Squared Error (MSE) loss and a RMSprop optimizer the DQN model can be iteratively trained. The better trained the DQN network becomes, the more able it is to select an optimal action that maximizes the sum of all future rewards **Eq. 4**.

For basic obstacle detection and localization we used LiDAR to provide environmental state sensing, and GPS, digital compass, and IMU to provide vehicle state sensing. The entire state input for our DQN is given in **Eq. 6** where θ_1 is the desired heading to the mission objective (as dictated by mission path), d_1 is the distance to the mission objective, θ_2 is the heading to an observed obstacle, d_2 is the distance to said obstacle, and L_1 through L_{24} are ranges provided by an on-board LiDAR at 24 linear spaced intervals in the 120° field of view in front of the vehicle. All of the measurements are normalized on [0, 1] before being fed into the DQN neural net.

$$s = [\theta_1, d_1, \theta_2, d_2, L_1, L_2, \ldots, L_{24}]^T \tag{6}$$

$$a = [HLT, LT, S, RT, HRT]^T \tag{7}$$

The DQN's action space is comprised of five discrete actions ($a \in \mathbb{R}^5$). These actions are discrete vehicle commands (**Eq. 7**) which can be qualified as Hard Left Turn (HLT), Left Turn (LT), Straight (S), Right Turn (RT), and Hard RightTurn (HRT). This small subset of actions are chosen to reduce complexity of the neural network and are interpreted as five vehicle heading augmentations between ± 80° with 40° of separation. These augmentations when combined with vehicle heading are used to define a new waypoint. Larger action spaces will provide smoother paths at the cost of more complex networks. As described earlier action selection is done by selecting the action that has the highest approximated Q-value. While a large continuous action space such as linear and rational rates ($a \in \mathbb{R}$) is common in DRL applications it does not work in this instance as it requires the DRL agent to directly control the agents movements at a low level. Furthermore the base DQN utilized within this work, and other value-based DRL agents, are not capable of outputting continuous actions.

4 METHODOLOGY

To facilitate cross-domain training and implementation we have created a DRL agent that interfaces with the a given robotic platform at a high level and enables the agent to operate without regard or understanding of the underlying vehicle dynamics. To achieve this, the DRL action space includes path augmentation commands instead of steering and throttle control commands. Simplification of the observation space through sensor state normalization also enables cross-domain generalization. This freedom allows agent abstraction from both vehicle and domain allowing cross-vehicle and cross-domain learning and implementation.

4.1 Agent Implementation

In our methodology, the DRL agent described in **Section 3** is implemented as a tertiary level in a multi-level control strategy. This tertiary level serves as both a path augmentation agent for obstacle avoidance and a watchdog for control state switching when an obstacle is sensed in the surrounding environment. As shown in **Figure 2**, the DRL agent as well as the traditional path planning controller receive commanded mission waypoints (shown in red). While the mission planner always provides a path for the entirety of the mission utilizing a traditional path assembly method, the DRL agent does not provide any input until an obstacle is sensed.

Algorithm 1. Path Augmentation and Waypoint Insertion for Obstacle Avoidance initialization;

Algorithm 1: Path Augmentation and Waypoint Insertion for Obstacle Avoidance

```
initialization;
mission = [[initialization point], [n waypoints of type (lat, long)]];
current_Path = path between mission waypoints;
while running do
    take in DRL observation space (Eqn. 6);
    if minimum returned LiDAR Distance ≤ safety threshold then
        DRL Agent Prediction (Eqn. 7);
        temporary waypoint = destination_point(action (a), distance (D), vehicle_heading (θ));
        inject temporary waypoint into mission affix to temporary waypoint utilizing LOS;
    else
        update look ahead distance;
        calculate ILOS for path following (Eqn. 11);
    low-level vehicle actuation;
```

In the event that an obstacle in the environment is sensed in the observation space by the trained DRL agent (**Eq. 6**), the agent will provide an action from the action space (**Eq. 7**) which will be given to the path planner. The path planner interprets the DRL agent action into a temporary waypoint which is injected into the mission path. The path planner and follower ceases following the mission path and begins providing low-level control references towards the temporary waypoint, thus navigating away from the nominal path and onto a dynamic path. The temporary waypoint used to build the new path is updated upon every prediction created by the DRL agent (1–2 Hz). This process is shown in **Algorithm 1**, where the function destination_point is represented by **Eq. 8** through **Eq. 10**, (Veness, 2010). These function will translate DRL actions (**Eq. 7**) into waypoints. Subscripts of $_1$ represent the location of the autonomous agent, subscripts of $_2$ represent the new waypoint being created, λ represents radians of longitude, Ψ represents radians of latitude, Θ_1 is the current heading of the autonomous agent in the global reference frame, D is the user set distance away from the boat to create the waypoint (considered 2 m in this work), and R is the radius of the earth if represented as a perfect sphere.

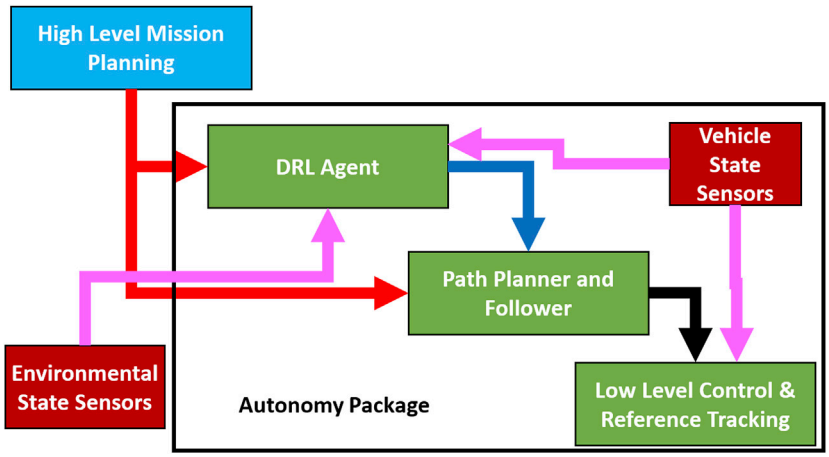

FIGURE 2 | Illustration showing methodology of tertiary controllers integration of the DRL agent and lower level controllers. All processing components are shown in green, autonomy package items are within the black bounding box, mission waypoint flow is shown in red, flow of state information from sensors is shown in magenta, and DRL actions/predictions are shown in blue. Reference headings are shown in black and are based on the DRL action when an action is provided and on the mission path when one is not.

$$\phi = ((3 - a_i) \times 40) + \Theta_1 \quad \text{where } a_i \text{ is the action index with respect to Eqn. 7} \tag{8}$$

$$\Psi_2 = asin\left(\sin(\Psi_1) \cdot \cos\left(\frac{D}{R}\right) + \cos(\psi_1) \cdot \sin\left(\frac{D}{R}\right) \cdot \cos(\phi)\right) \tag{9}$$

$$\lambda_2 = \lambda_1 + atan2\left(\sin(\phi) \cdot \sin\left(\frac{D}{R}\right) \cdot \cos(\Psi_1), cos\left(\frac{D}{R}\right) - \sin(\Psi_1) \cdot \sin(\Psi_2)\right) \tag{10}$$

The DRL agent essentially acts with a carrot and stick strategy, leading the lower control levels where it deems the vehicle should go by redrawing the vehicle path through waypoint creation, but not directly controlling vehicle movement. By focusing on high-level processes the DRL agent leaves real-world dynamic responses to lower level controllers. A PID switching function is implemented on the low level controller to change the aggressiveness of heading error tracking when an obstacle is observed versus not.

The heading reward (**Eq. 2**) is calculated with respect to the prescribed path, in addition the DRL agent has no insight into the total mission path. This means the DRL is rewarded for following the prescribed path (θ_1 in **Eq. 6**) when no obstacle is present, which creates a redundant value for vehicle control at the expense of power consumption and heat production. Thus, the predictions of the DRL agent ceases when either no obstacle is perceived or the mission is completed. Once prediction stops, the vehicle re-affixes to the original mission path to resume a nominal trajectory.

The low-level controller receives its reference tracking values from the path planner and has no insight if the reference value is from a nominal mission path or an augmented one. This methodology is implementable in any domain under the assumption that all actuation and path generation is 2-dimensional, as such any 2-D path generation can be utilized with this methodology that allows for dynamic re-planning. This is the case for ground and water vehicles as well as aerial and subsea vehicles operating at constant depth/altitude. The only aspect of this methodology that does not carry between vehicles and domains is the low-level controller. However, such controllers are typically already implemented in commercial autonomous vehicles or can be easily created and tuned.

4.2 Autonomy Package

To facilitate the rapid development of low-cost machine learning capable vehicles across multiple domains a generic and open sourced autonomy package has been developed. The package consists of vehicle and environmental state sensing equipment, communication equipment, and a distributed processing center, **Figure 3**. The entire package is contained within a splash proof container that can be seamlessly integrated into any AGV or ASV with a 12 V power supply. Implementation in Autonomous Underwater Vehicles or other vehicles is also possible with minimal component packaging modifications.

The distributed processing equipment comprises two low-cost single board computers and a computer capable of running the DQN serves as a computational engine operating together in a distributed processing environment managed by the Robotic Operating System (ROS). Each processing module operates at a different level of vehicle abstraction. The package's two Raspberry Pi's operate as a frontseat-backseat duo, while the DRL agent provides prediction and insight into the vehicle's changing environment. The ROS implementation also allows simulation of the frontseat and backseat by including their respective nodes into a simulated environment.

The backseat (Path Planner and Follower in **Figure 2**) uses a commanded mission profile (GPS waypoints and times) to

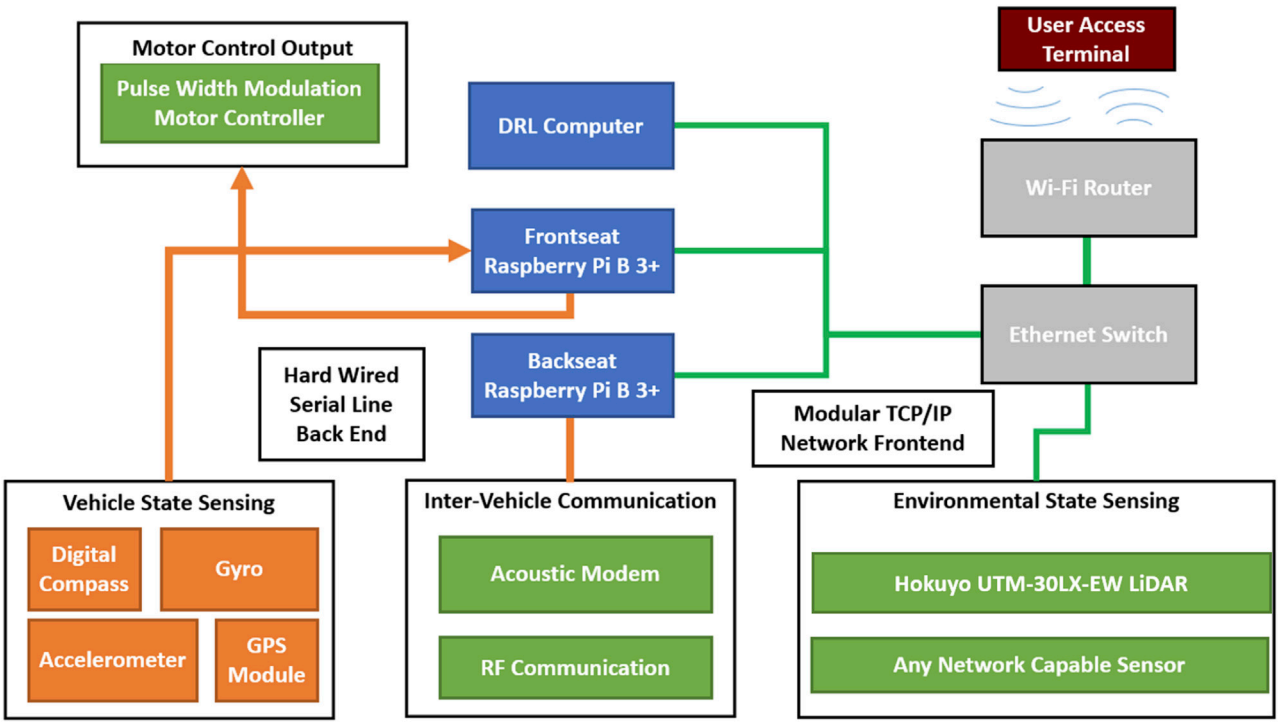

FIGURE 3 | Illustration showing components which comprise the autonomy package. Blue items comprise computational processing items, orange items comprise fixed back-end sensors that are utilized for mid and low-level control, green items are mission specific modular components that can be integrated with computational components over USB (purple lines) or ethernet (green lines), grey items represent network management items that create a local network for distributed computation and sensing.

generate a path for completion of a delivered mission. This mission path can be a smooth trajectory such as a Dubins path (Dubins, 1957) or straight line between way-points. For this specific work, a line of sight path between waypoints was used for simplicity and construction of missions that force vehicle avoidance around obstacles.

The backseat controller monitors path progression and provides corrected desired bearings to follow the path with an ILOS control strategy (Fossen et al., 2015) as given in **Eq. 11**.

$$\phi_D = \gamma_P + tan^{-1}\left(-\frac{1}{\Delta}y_e - \hat{\beta}\right) \tag{11}$$

Where ϕ_D is the desired heading, γ_P is projected heading obtained from the desired path, y_e (cross-track error) is the length of the vector from the vehicle to the prescribed path which intersect the path at a right angle, $\hat{\beta}$ is the estimate of sideslip angle, and Δ is the lookahead distance, which can be set by the user or dynamically updated.

The frontseat (Low-Level Control & Reference Tracking in **Figure 2**) interfaces with vehicle state sensing and localization using the on-board digital compass, and GPS receiver. The frontseat takes the desired compass bearing and desired velocity provided from the backseat to calculate a desired rotational and linear velocity with two PID controllers that are tuned to the dynamics of the vehicle equipped with the autonomy package.

A graphical representation of autonomy package items, connections, and modularity is shown in **Figure 3**. The core components and associated cost of the autonomy package are listed in **Table 1**. The low-cost of the package allows reproduction and repairs to be made quickly and efficiently in the event of a catastrophic failure due to improper agent predictions. This alleviates real-world training risks of vehicle loss. To provide observation space (**Eq. 6**) values, LiDAR measurements were obtained from a Hokuyo UTM-30LX-EW LiDAR module that was connected to the autonomy package's network for this series

TABLE 1 | Itemized list of Autonomy Package components.

Item	Number
Raspberry Pi B+	2
NVIDIA Jetson Nano	2
Gumstix Pre-Go GPS Module	1
GPS Patch Antenna	1
HMC6343 Digital Compass	1
Adafruit BNO055 IMU	1
DMC60C Motor Controllers	2
Splash-Proof Enclosure	1
TP-Link N450 WiFi Router	1
TP-Link TL-SG105 Switch	1
5x Ethernet Cables	1
BESTEK 500W Power Inverter	1
LM2596 DC-DC Converter	1
DROK DC-DC Converter	1
6AWG Power quick disconnect cabling	1

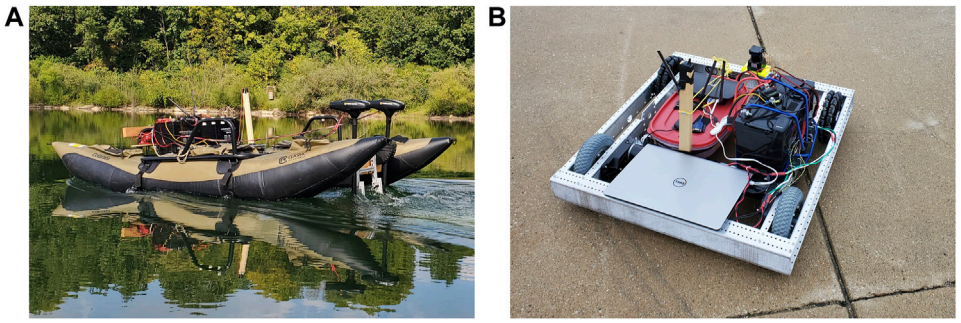

FIGURE 4 | ASV and AGV low-cost COTS vehicles built for use in cross-domain DRL development. Both vehicles utilize the Autonomy Package detailed in **Section 4.2**.

of tests. No other sensors were utilized for these tests, but could be integrated with the autonomy package due to its modular design. Any USB capable or network capable sensor can be integrated without any physical package modifications, only a ROS wrapper is required to make sensor data accessible to the rest of the autonomy package. Thus more complex sensors such as 3D LiDAR, monocular cameras, sonar, and even stereo cameras can be easily incorporated into the architecture to increase the environmental state information available to the DRL agent.

The presented methodology for autonomous vehicle control creates a layered architecture with increased levels of abstraction from the vehicles state while also providing layers of interrupt priority ensuring vehicle safety from obstacles while still fulfilling mission priorities. The DRL agent implementation has scope into the entire state of the vehicle and its surrounding environment allowing sensing of obstacles. The frontseat-backseat controllers in the autonomy package have scope of all environmental and mission variables to manage mission objectives and planned paths.

4.3 Robotic Platforms

To validate the DRL agent's ability to generalize across domains two robotic platforms were chosen to utilize the autonomy package to perform autonomous obstacle avoidance dependent missions on the water and ground domains. Both platforms were chosen such that control was achieved through differential steering and had a base platform cost under $900.00 USD. The autonomy package was utilized with both systems to deliver autonomous navigation capability.

For the water domain BREAM (Boat for Robotic Engineering and Applied Machine-Learning) (Lambert et al., 2020) was chosen as an ideal platform. BREAM is a low-cost and versatile Autonomous Surface Vehcile (ASV). The low-cost and modular vehicle is driven by two external trawling motors providing differential steering controlled by the Autonomy Package. All components are mounted on a commercial off-the-shelf (COTS) inflatable catamaran hull that provides a total payload capacity of 100 kg in its current configuration. The autonomy package and motors are powered by two 12 V AGM deep cycle marine batteries connected in parallel. **Figure 4A** shows BREAM with the autonomy package in action. The vehicle has over 60 h of open water operational time completing missions prior to this work.

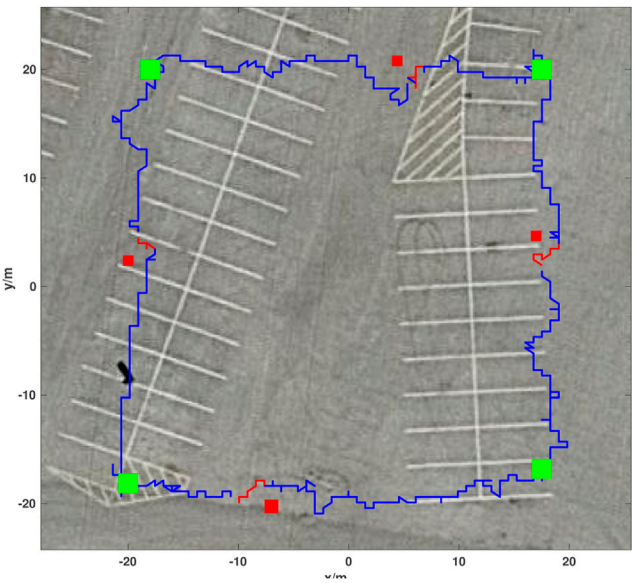

FIGURE 5 | Real-world AGV obstacle avoidance mission showing mission waypoints as green squares, placed obstacles as red squares, DRL agent augmented paths for avoidance in red, and the vehicle trajectory following the mission path in blue. The sample mission shown took 250 s and covered 120 m. The mission trajectory starts at the bottom left mission waypoint and progresses counter-clockwise.

An Autonomous Ground Vehicle (AGV) (**Figure 4B**) comprised of a COTS chassis and differential drive system was utilized for ground based DRL training and actuation. The base vehicle consists of a chassis, two omni-directional wheels, two pneumatic drive wheels, two brushed DC motors, and a single AGM deep cycle battery. By controlling the vehicle through the autonomy package the vehicle was made fully operational without any modification to the COTS chassis and was able to complete missions autonomously after 15 h of tuning and adjusting low-level control parameters through full deployment in the real-world

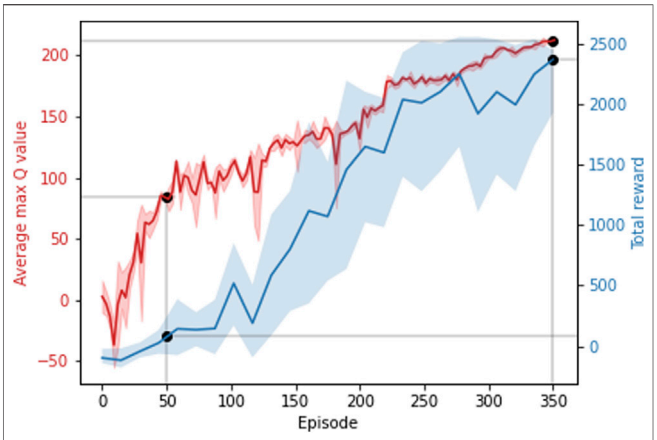

FIGURE 6 | Average max Q-value and total reward across training episodes. Showing the Average max Q-value and the total reward at 50 and 350 episodes.

environment. Further development allowed completion of missions as shown in **Figure 5**.

In the event that the DRL agent fails to ensure vehicle safety during training/testing both of the high-value low-cost robotic platforms will not put undue financial stress on users. These systems serve as low-risk tools for the rapid development and testing of DRL agents in the real-world. The modularity, inexpensiveness, and simplicity of the autonomy package means that a lost or damaged platform can be easily rebuilt or repaired rapidly for continued testing. Once training is completed and agent confidence is high then the agent can be transplanted to a final platform capable of operation in a specific environment that is desired (off-road, blue water, etc.).

5 EXPERIMENTAL RESULTS AND DISCUSSION

To test the proposed methodology's abilities to allow cross-domain generalization a simple DQN is assembled and trained to avoid obstacles on an AGV and then implemented without alteration in a simulated water surface environment and in the real work for further validation. In each case a mission path was constructed by connecting goal waypoints by line-of-sight bearings that connect subsequent waypoints, either in open or closed paths. To quantitatively categorize the amount of deviation from the optimal path (line-of-sight between goal waypoints) cross-track error was utilized as a metric. Initial DRL training was done with a simulated AGV and environment within the Gazebo simulation software. The DRL agent was also implemented on an ASV simulation and on an AGV for operation in urban environments. A comparison between trajectories at different training episodes is utilized to evaluate the DRL agent's performance. The results for each simulated and real implementation are then discussed.

The DQN model itself is implemented in Python using Tensorflow (Abadi et al., 2016) and Keras. The network itself is contained with a ROS node. Training is done on a desktop computer with a GeForce 2080 GPU and an Intel Core i7-8700 CPU. The AGV testing is done on a laptop with an UHD 620 GPU and an Intel Core i5-5200U CPU. To make the system more compact and protect the computer from water damage, the ASV testing utilized a Nvidia Jetson Nano with a 128-core Maxwell GPU and a Quad-core ARM A57 CPU. The Nano is the end-use computational device for the Autonomy package.

The DQN model has 3 ReLu dense layers, a dropout layer with a dropout rate of 0.2, and a output layer using linear activation function. The learning rate for the DQN is chosen as 2.5×10^{-4}. The greedy policy used for state exploration ϵ starts at 1 and updates every episode. ϵ decay rate is set to 0.99 for greater exploration of state spaces. RMSprop is used for optimization. Models will be saved every 10 episodes. An episode is terminated after a collision or if 6,000 steps have occurred (a step takes 0.1 s).

5.1 Ground Domain Implementation

To test the autonomy package and train the DQN model, we created a simulation environment using Gazebo. **Figure 1** shows the simulated vehicles and obstacles with LiDAR data and target position visualization. The simulated robot receives its position and heading from the gazebo environment. The Gazebo environment contains obstacles and walls that enclose the test area. An AGV model based on the real-world test AGV platform was created using SolidWorks to better simulate the behavior of the AGV's dynamics. The vehicle consists of an off-the-shelf chassis, four wheels, and two motors which operate in a differential drive configuration. Video of AGV obstacle avoidance can be seen in https://youtu.be/IDkI6rXP1r8.

The obstacles have different shapes and sizes and are placed between mission waypoints. The DQN model is trained during execution of the waypoint navigation tasks. The AGV starts at the first waypoint, and begins moving to all subsequent mission waypoints that are present. If a collision occurs, the AGV model will reinitialize at the origin and another simulation begins.

It can be seen in **Figure 6** that after more than 350 training episodes, the average max Q-value reaches 220 and the total reward reaches approximately 2,400, showing that the average max Q-value and reward increases with respect to training episodes. A reward value of 2,400 contains goal reward values from 1,500 to 2000 (30–40 achieved waypoints) and step reward values of 900–400. A model with such a total reward can avoid obstacles until simulation timeout at 6,000 steps. **Figure 7** compares the behavior of the DQN model at different episodic training intervals. At 50 training episodes collisions occurred three times. The average cross-track error was 1.10 m the average deviation across DRL agent avoidance was 1.14 as shown in **Table 2**, while at 350 episodes the AGV avoided all obstacles and the DQN gave a more optimized trajectory around the obstacles. The average cross-track error was 0.68 m and the average cross-track error across DRL agent avoidance was 0.89 m. When the AGV executed the mission without obstacles, the average cross-track error was 0.16 m. The simulated ground environment formed a baseline model for cross-domain testing.

To show that the model trained in the simulated environment is valid in the real-world, the trained DQN model was integrated into an experimental AGV platform, **Figure 4B** and tested. The AGV utilizes the mid and low-level control framework of the autonomy package but a different mission to the simulation. The course utilized

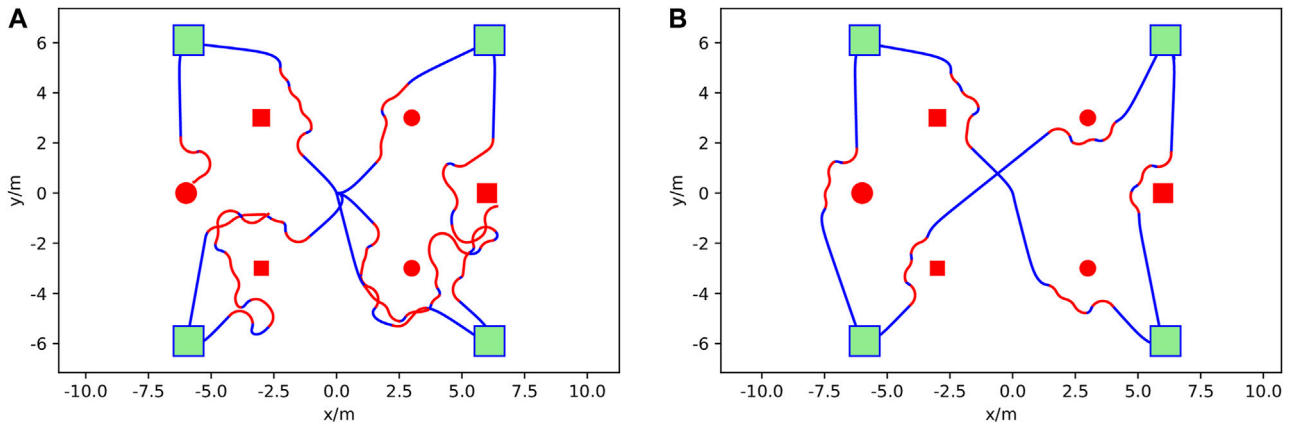

FIGURE 7 | Obstacle avoidance mission comparison at different training episodes in the AGV simulation. **(A)** is at 50 episodes and **(B)** is at 350 episodes. Mission waypoints are shown as green squares, obstacles are shown in red, DRL agent augmented paths for avoidance in red, and the vehicle trajectory following the mission path in blue. The AGV starts from the top right waypoint and then goes to the bottom right, top left, bottom left, and top right in sequence.

TABLE 2 | Cross-track error of different tests. $\overline{y_e}$ column lists the average cross-track errors across the entire mission, $\overline{y_e}(activated)$ column lists the average cross-track errors when the DRL agent is activated, and max(y_e) column lists the max cross-track error observed during the entire test. Tests with no obstacles are not shown within this work.

Test	$\overline{y_e}$	$\overline{y_e}$(activated)	max(y_e)
agv_sim_50_eps	1.10	1.14	3.40
agv_sim_350_eps	0.68	0.89	1.85
agv_sim_no_obstacle	0.16	N/A	0.34
agv_real	1.80	1.93	2.70
agv_real_no_obstacle	0.56	N/A	2.91
asv_sim_with_wind	5.99	7.78	15.59
asv_sim_with_wind_no_obstacle	3.01	N/A	4.98
asv_real_square	1.39	4.88	7.75
asv_real_triangle	1.81	5.66	7.80
asv_real_no_obstacle	0.75	N/A	7.90

for testing is comprised of four waypoints that make up the vertices of a 40 by 40 m^2.

Four corrugated cardboard boxes were utilized as obstacles for this test. Each boxes is 35 long × 35 wide × 30 cm tall. To better test the model's performance, obstacles were purposefully placed along the AGV's prescribed mission path, with one obstacle on each side of the prescribed course during operation to ensure that DRL agent avoidance occurred. Obstacle positions were obtained and overlaid with the AGV's resulting path, **Figure 5**. As presented in **Table 2**, the trajectory has a mean cross-track error of 1.80 m and an average cross-track error from obstacle avoidance of 1.93 m.The trajectory plot shows that the AGV is able to navigate through an environment with static obstacles without collision and find a path to its goals. In contrast, the mean cross-track error is 0.56 when no obstacles were within the mission environment.

5.2 Water Domain Implementation

To validate the ability of the DQN model implementation methodology to generalize across different domains, a simulated ASV was tested in a marine environment created in the Virtual RobotX (VRX) simulator (Bingham et al., 2019). To facilitate rapid development of water simulation an open source ASV model of the WAM-V was utilized.

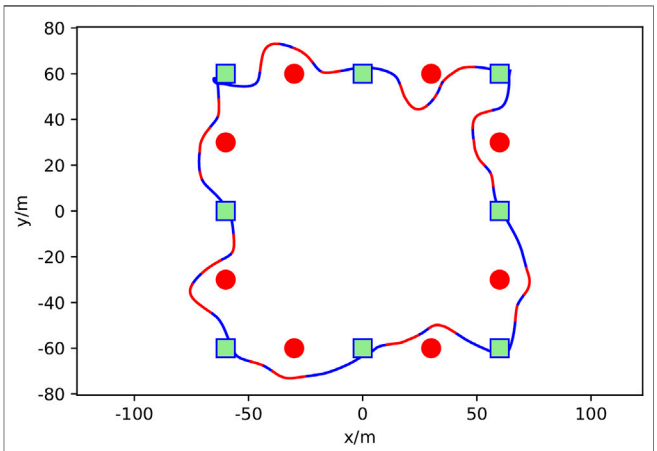

FIGURE 8 | Simulated ASV obstacle avoidance mission showing mission waypoints as green squares, obstacles in red, DRL agent augmented paths for avoidance in red, and the vehicle trajectory following the mission path in blue. The trajectory runs counter-clockwise from the top center mission waypoint.

The WAM-V dynamics model was coupled with the DQN agent trained in the AGV simulation and tested on the real-world AGV, as well as the mid and low-level controllers of the autonomy package. The simulation was also given appropriate sensors for environmental and position feedback, specifically a 2-D LiDAR, GPS, and IMU. Gazebo "buoyancy" and "ASV dynamics" plugins were used to simulate the marine environment and dynamic behavior of the vehicle. The simulation used a constant wind disturbance of 5 m per second along the minus *y* axis. Due to the different physical properties of AGV and ASV, larger course and obstacles were setup. All obstacles have a radius of 4 m. The max LiDAR scan range was changed from 3 to 15 m for the ASV to sense and decide whether to

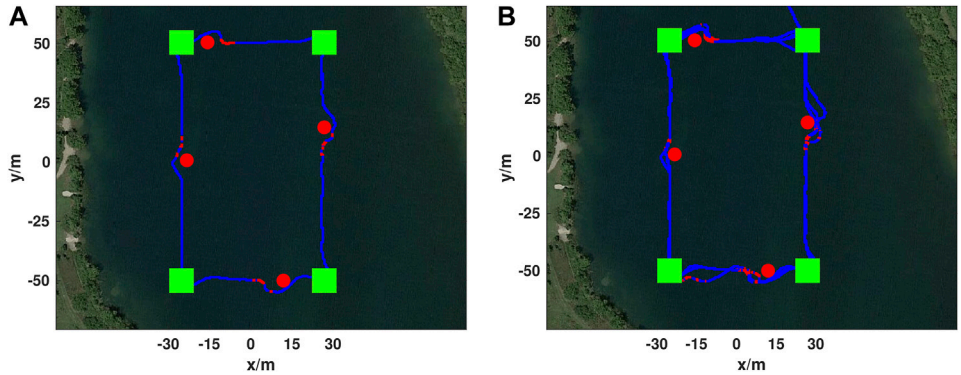

FIGURE 9 | Real-world ASV obstacle avoidance test. Mission waypoints as shown as green squares, placed obstacles as red circles, DRL agent augmented paths in red, and the vehicle trajectory following the mission path in blue. An example trajectory from the test is shown **(A)** and took 252 s and covered 320 m. In total this mission ran for 19 min without collision **(B)** and only stopped due to a human command to stop actuation. The augmented path shown close to the bottom left waypoint (–30, –50) was caused by a passing boat. The mission trajectory starts at (20, 70) and first goes to the top right mission waypoint (30, 50) and progresses counter-clockwise.

react to obstacles and augment the prescribed mission path earlier in the path. However, this did not affect the agent's trained DQN weights as the observations space values are normalized.

After loading the agent trained from **Section 5.1** and tuning the low level PID controller of the autonomous package to the WAM-V dynamics, the ASV is able to finish the waypoint navigation task without any collisions. The trajectory is shown in **Figure 8**. As listed in **Table 2**, the average cross-track error was 5.99 m, a maximum deviation from the optimal path of 15.59 m, and an average cross-track error across DRL agent avoidance was 7.78 m. This validation shows that the DQN agent is able to rapidly generalize across domains for the obstacle avoidance task that it was trained to do with only extremely minimal low level controller changes regarding how rapidly the vehicle alters course to path augmentations. The PID tuning in this case only took approximately 2 h before the ASV was avoiding obstacles as well as the AGV did in real-world implementations. A mission without obstacles was utilized to test the controller's performance. The average cross-track error was 3.01 m.

To test the DRL agent's ability to generalize across domains without additional training in a real water environment the BREAM ASV described in **Section 4.3** was used. Like the WAM-V model used in the water surface simulation, BREAM is a differentially actuated catamaran ASV. However, unlike the WAM-V (Pandey and Hasegawa, 2015), BREAM is able to be deployed from shore, and is small enough for transportation without a trailer and testing in small lakes or ponds. Furthermore grounding of a WAM-V due to DRL agent failure could prove costly and repairs could be difficult or take a significant amount of time.

To verify obstacle avoidance a mission and associated course was set up at Fairfield Lakes Park in Lafayette, with the resulting course being rectangular in shape with side lengths of 60 and 100 m. This can be seen in **Figure 9A**. The DRL agent that was trained in the AGV simulation and tested on the real-world AGV and ASV simulation was utilized for this test without any further training on the real-world ASV.

Eight inflatable open water marker buoys were utilized as obstacles for this test, four red and four yellow. Each buoy is spherical with an inflated diameter of 51 cm and is constructed of durable plastic. Each obstacle placed on the course was composed of two such buoys tied together and anchored to the lake bottom with a weight in such a way that minimal slack was present in the anchor line to prevent obstacle drift.

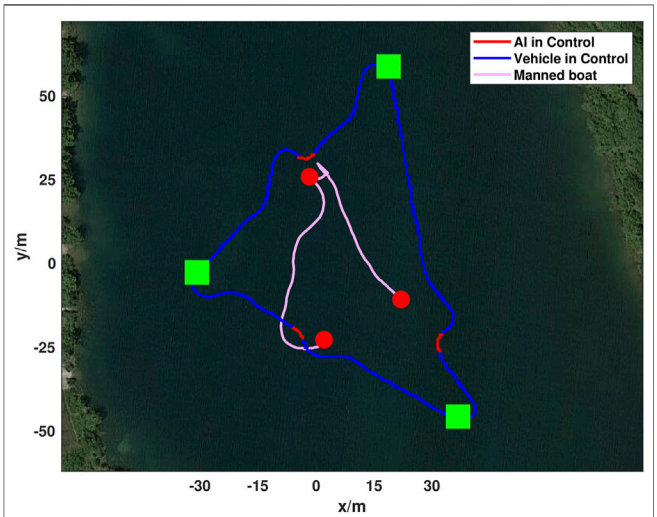

FIGURE 10 | Real-world ASV obstacle avoidance test utilizing a manned boat to inject obstacles into the vehicles path. Mission waypoints are shown in green, ASV DRL agent augmented paths are in red, and the ASV trajectory following the mission path is in blue. The manned boat trajectory is shown in pink and locations where it was first identified as an obstacle by the DRL agent are shown as red circles. Both trajectories start at the bottom left and progress counter-clockwise. The mission shown took 310 s and covered 398 m.

One obstacle was placed on each side of the prescribed course, with the obstacles on the longer sides being set approximately at the midpoint of the side and obstacles on the shorter sides being placed close to the goal waypoint defining one of the course vertexes. The ASV prescribed mission was done so that the ASV traversed in the counter-clockwise direction. The results of the test are shown in

Figure 9 with image A showing a single lap around the course, and image B showing 19 min of continual operation around the course. In total two independent tests were done comprising 52 min and 15 s of continuous avoidance, covering over 3.5 km. Video of ASV obstacle avoidance on a rectangular obstacle avoidance test can be seen in https://youtu.be/IDkI6rXP1r8?t=53.

Throughout the test shown in **Figure 9** and **Table 2** the average cross-track error was 1.39 m, a maximum deviation from the optimal path of 7.75 m, and an average cross-track error across DRL agent avoidance was 4.88 m.

To demonstrate the generalization of the proposed method across dissimilar objects and mission paths a triangular mission was created at Fairfield Lakes, **Figure 10**. The mission course was run a total of 15 times, however only a single lap is shown for visual clarity. In-place of geometrically simple marker buoys a manned boat was used to intersect the ASV's path. The manned boat presented an asymmetric surface with both convex and concave surfaces of non-uniform reflectively and color for the ASV to react to. Manned incursions into the vehicle's path was done to create an unpowered drifting obstacle for avoidance by either crossing or nearly crossing the path. On two occasions during testing contact between vehicles did occur, however this is likely due to the drifting of the manned boat as the DRL agent was not trained to recognize or react to dynamic obstacles. The cross-track error of the mission shown in **Figure 10** and **Table 2**. It was similar to the first ASV mission with a mean cross-track error of 1.81 m, a maximum error of 7.80 m, and an average cross-track error from obstacle path augmentations of 5.66 m. In comparison, the average cross-track error was 0.75 m when the ASV executed a different mission without obstacles. The maximum cross-track error without obstacles was caused by the behaviors and limitations of under-actuated vehicle dynamics when completing a turn-around maneuver. This shows that the effectiveness of the DRL agent is irrespective of obstacle shape or path simplicity.

Throughout testing the largest source of error was the inclination angle and stiffness of the 2-D LiDAR Hokuyo sensor mount on the ASV. With the sensor mounted approximately 30 cm above the waterline the sensor had to be angled properly to allow the 2-D environmental slice to intersect with the water at a proper distance in front of the ASV. Additional problems occurred when the ASV encounter waves and the azimuth of the LiDAR scan relative to the water surface changed. This caused either false positive obstacle readings, or false negative readings depending on the direction of boat pitch. To fix this issue the LiDAR system was mounted on a motorized gimbal to adjust then affix the LiDAR scan azimuth before the start of each test for optimal LiDAR performance. In the future this issue will be solved further by utilizing a 3-D scanning LiDAR.

6 CONCLUSION AND FUTURE WORK

In this paper, a cross-domain capable obstacle avoidance DRL agent is presented. The agent is implemented utilizing a methodology that facilitates cross-domain training and operation. The methodology includes a tertiary multilevel controller and an autonomy package. The tertiary controller enable the DRL agent to be separated from vehicle dynamics and environmental constraints. The autonomy package provides controllers and electrical hardware to rapidly turn off-the-shelf robots into autonomous DRL agents. The cross-domain obstacle avoidance ability of the methodology was validated with a DQN on an AGV and ASV simulation as well as on a real AGV and ASV. The ASV DRL implementation was also validated with dissimilar obstacles. In each case the DRL agent was able to successfully provide path augmentations to steer robotic platforms clear of obstacles encountered during a mission with minimal path deviation. The results detailed within this work show that the prescribed methodology not only aids in generalizing between ground and water domains but also helps bridge the simulation real-world gap. Thus enabling a reduction in time and cost overhead associated with training in challenging real-world domains.

Future work on this project is extensive. The tests will be expanded to validate the proposed methodology on more advanced DRL models such as Actor-Critic and Double DQN and on higher-dimension sensors such as 3-D LiDAR. These more advanced methods will be used to handle more complex navigational problems such as dynamic obstacles and long-term deployments. The methodology detailed in this work also shows promise in allowing training of complex models in the underwater domain through aerial deployment and overall simulation to real-world transfer. However, further investigation is required to verify the feasibility of transitioning between the domains in more complex settings. In each of these more advanced cases it is improbable that an agent will be able to immediately generalize to its operational domain. A study to determine how well certain observation and action spaces transition between domains and the training required in each case for goal domain generalization is needed in the future. While future work is required, this methodology will serve as a starting point for further cross-domain agent development, which will expand the use of reinforcement learning agents on real-world robotic platforms in challenging environments such as ASVs in the water domain.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusion of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

RL developed methodology, led testing, drafted paper, and assisted with implementation. JL led DRL implementation, assisted with testing, and paper drafting. L-FW assisted with testing and drafting of paper. All work was conducted under the supervision of NM.

FUNDING

This material is based upon work supported by NSF 1921060 and ONR N00014-20-1-2085.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). "Tensorflow: A System for Large-Scale Machine Learning," in 12th {USENIX} Symposium on Operating Systems Design and Implementation (Savannah, GA: {OSDI}16), 265–283.

Aurelien, G. (2019). *Chapter 18 - Reinforcement Learning*. O'Reilly Media, Incorporated, 609–664.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade Learning Environment: An Evaluation Platform for General Agents. *jair* 47, 253–279. doi:10.1613/jair.3912

Bingham, B., Aguero, C., McCarrin, M., Klamo, J., Malia, J., Allen, K., et al. (2019). "Toward Maritime Robotic Simulation in Gazebo," in Proceedings of MTS/IEEE OCEANS Conference, Seattle, WA. doi:10.23919/oceans40490.2019.8962724

Bovcon, B., Muhovic, J., Perš, J., and Kristan, M. (2019). "The Mastr1325 Dataset for Training Deep Usv Obstacle Detection Models," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (Macau, China: IROS), 3431–3438. doi:10.1109/IROS40897.2019.8967909

Caccia, M., Bibuli, M., Bono, R., and Bruzzone, G. (2008). Basic Navigation, Guidance and Control of an Unmanned Surface Vehicle. *Auton. Robot* 25, 349–365. doi:10.1007/s10514-008-9100-0

Cheng, Y., and Zhang, W. (2018). Concise Deep Reinforcement Learning Obstacle Avoidance for Underactuated Unmanned marine Vessels. *Neurocomputing* 272, 63–73. doi:10.1016/j.neucom.2017.06.066

Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. (2018). "End-to-end Driving via Conditional Imitation Learning," in 2018 IEEE International Conference on Robotics and Automation (Brisbane, QLD, Australia: ICRA), 4693–4700. doi:10.1109/ICRA.2018.8460487

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., et al. (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas, NV: CVPR). doi:10.1109/cvpr.2016.350

Dubins, L. E. (1957). On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *Am. J. Maths.* 79, 497–516. doi:10.2307/2372560

Dulac-Arnold, G., Mankowitz, D., and Hester, T. (2019). *Challenges of Real-World Reinforcement Learning*. Dataset.

Ferreira, H., Martins, R., Marques, E., Pinto, J., Martins, A., Almeida, J., et al. (2007). "Swordfish: an Autonomous Surface Vehicle for Network Centric Operations," in *OCEANS 2007* (Europe), 1–6. doi:10.1109/OCEANSE.2007.4302467

Fossen, T. I., Pettersen, K. Y., and Galeazzi, R. (2015). Line-of-sight Path Following for Dubins Paths with Adaptive Sideslip Compensation of Drift Forces. *IEEE Trans. Contr. Syst. Technol.* 23, 820–827. doi:10.1109/tcst.2014.2338354

Giusti, A., Guzzi, J., Ciresan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., et al. (2016). A Machine Learning Approach to Visual Perception of forest Trails for mobile Robots. *IEEE Robot. Autom. Lett.* 1, 661–667. doi:10.1109/LRA.2015.2509024

Gupta, S. K. (2013). Developing Autonomy for Unmanned Surface Vehicles. *Tech.Rep*(College Park, MD: University of Maryland).

Han, J., Cho, Y., and Kim, J. (2019). Coastal Slam with marine Radar for Usv Operation in Gps-Restricted Situations. *IEEE J. Oceanic Eng.* 44, 300–309. doi:10.1109/JOE.2018.2883887

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., et al. (2018). "Rainbow: Combining Improvements in Deep Reinforcement Learning," in Proceedings of the AAAI Conference on Artificial Intelligence 32.

Hong, Z., Chen, Y., Su, S., Shann, T., Chang, Y., Yang, H., et al. (2018). Virtual-to-real: Learning to Control in Visual Semantic Segmentation. *CoRR abs/1802*, 00285. doi:10.24963/ijcai.2018.682

Kahn, G., Zhang, T., Levine, S., and Abbeel, P. (2017). "Plato: Policy Learning Using Adaptive Trajectory Optimization," in 2017 IEEE International Conference on Robotics and Automation (Singapore: ICRA), 3342–3349. doi:10.1109/ICRA.2017.7989379

Karapetyan, N., Moulton, J., and Rekleitis, I. (2021). "Meander-based River Coverage by an Autonomous Surface Vehicle," in *Field and Service Robotics*. Editors G. Ishigami and K. Yoshida (Singapore: Springer Singapore), 353–364. doi:10.1007/978-981-15-9460-1_25

Kaspar, M., Osorio, J. D. M., and Bock, J. (2020). *Sim2real Transfer for Reinforcement Learning without Dynamics Randomization*. Las Vegas, NV: CoRR abs/, 11635.

Kellenberger, B., Marcos, D., and Tuia, D. (2018). Detecting Mammals in UAV Images: Best Practices to Address a Substantially Imbalanced Dataset with Deep Learning. *Remote Sensing Environ.* 216, 139–153. doi:10.1016/j.rse.2018.06.028

Lambert, R., Page, B., Chavez, J., and Mahmoudian, N. (2020). "A Low-Cost Autonomous Surface Vehicle for Multi-Vehicle Operations," in *Global Oceans 2020* (Singapore: U.S. Gulf Coast), 1–5. doi:10.1109/IEEECONF38699.2020.9389236

Lei, X., Zhang, Z., and Dong, P. (2018). Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning. *J. Robotics* 2018, 1–10. doi:10.1155/2018/5781591

Loquercio, A., Maqueda, A. I., del-Blanco, C. R., and Scaramuzza, D. (2018). Dronet: Learning to Fly by Driving. *IEEE Robot. Autom. Lett.* 3, 1088–1095. doi:10.1109/LRA.2018.2795643

Mancini, M., Costante, G., Valigi, P., Ciarfuglia, T. A., Delmerico, J., and Scaramuzza, D. (2017). Toward Domain independence for Learning-Based Monocular Depth Estimation. *IEEE Robot. Autom. Lett.* 2, 1778–1785. doi:10.1109/lra.2017.2657002

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level Control through Deep Reinforcement Learning. *Nature* 518, 529–533. doi:10.1038/nature14236

Müller, M., Dosovitskiy, A., Ghanem, B., and Koltun, V. (2018). *Driving Policy Transfer via Modularity and Abstraction*. London, United Kingdom: CoRR abs/1804, 09364.

Nicholson, D. P., Michel, A. P. M., Wankel, S. D., Manganini, K., Sugrue, R. A., Sandwith, Z. O., et al. (2018). Rapid Mapping of Dissolved Methane and Carbon Dioxide in Coastal Ecosystems Using the ChemYak Autonomous Surface Vehicle. *Environ. Sci. Technol.* 52, 13314–13324. doi:10.1021/acs.est.8b04190

Osinski, B., Jakubowski, A., Ziecina, P., Milos, P., Galias, C., Homoceanu, S., and Michalewski, H. (2020). "Simulation-based Reinforcement Learning for Real-World Autonomous Driving," in 2020 IEEE International Conference on Robotics and Automation ((ICRA) (IEEE)). doi:10.1109/icra40945.2020.9196730

Pandey, J., and Hasegawa, K. (2015). "Study on Manoeuverability and Control of an Autonomous Wave Adaptive Modular Vessel (Wam-v) for Ocean Observation," in 2015 International Association of Institutes of Navigation World Congress (Prague, Czech Republic: IAIN), 1–7. doi:10.1109/iain.2015.7352248

Perera, A. G., Wei Law, Y., and Chahl, J. (2018). "Uav-gesture: A Dataset for Uav Control and Gesture Recognition," in Proceedings of the European Conference on Computer Vision (ECCV) Workshops.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play. *Science* 362, 1140–1144. doi:10.1126/science.aar6404

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). "Deterministic Policy Gradient Algorithms," in nternational conference on machine learning (Beijing, China: PMLR)), 387–395.

Singla, A., Padakandla, S., and Bhatnagar, S. (2021). Memory-based Deep Reinforcement Learning for Obstacle Avoidance in UAV with Limited Environment Knowledge. *IEEE Trans. Intell. Transport. Syst.* 22, 107–118. doi:10.1109/tits.2019.2954952

Sun, Y., Ran, X., Zhang, G., Xu, H., and Wang, X. (2020). AUV 3d Path Planning Based on the Improved Hierarchical Deep Q Network. *Jmse* 8, 145. doi:10.3390/jmse8020145

Taipalmaa, J., Passalis, N., Zhang, H., Gabbouj, M., and Raitoharju, J. (2019). "High-resolution Water Segmentation for Autonomous Unmanned Surface Vehicles: a Novel Dataset and Evaluation," in 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (Pittsburgh, PA: MLSP), 1–6. doi:10.1109/MLSP.2019.8918694

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in 2017 IEEE/RSJ international conference on intelligent robots and systems ((IROS) (IEEE)), 23–30. doi:10.1109/iros.2017.8202133

Todorov, E., Erez, T., and Tassa, Y. (2012). "Mujoco: A Physics Engine for Model-Based Control," in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 5026–5033. doi:10.1109/IROS.2012.6386109

van Hasselt, H., Guez, A., and Silver, D. (2016). "Deep Reinforcement Learning with Double Q-Learning," in Proceedings of the AAAI Conference on Artificial Intelligence 30.

Veness, C. (20102002–2014). Movable Type Scripts.Calculate Distance, Bearing and More between Latitude/longitude Points.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). "Dueling Network Architectures for Deep Reinforcement Learning," in Proceedings of The 33rd International Conference on Machine Learning. Editors M. F. Balcan and K. Q. Weinberger (New York, New York, USA: PMLR)Proceedings of Machine Learning Research), 48. 1995–2003.

Woo, J., and Kim, N. (2020). Collision Avoidance for an Unmanned Surface Vehicle Using Deep Reinforcement Learning. *Ocean Eng.* 199, 107001. doi:10.1016/j.oceaneng.2020.107001

Yan, C., Xiang, X., and Wang, C. (2019). Towards Real-Time Path Planning through Deep Reinforcement Learning for a UAV in Dynamic Environments. *J. Intell. Robot Syst.* 98, 297–309. doi:10.1007/s10846-019-01073-3

Yu, C., Liu, Z., Liu, X., Xie, F., Yang, Y., Wei, Q., et al. (2018). "Ds-slam: A Semantic Visual Slam towards Dynamic Environments," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (Madrid, Spain: IROS), 1168–1174. doi:10.1109/IROS.2018.8593691

Zhang, J., Yu, Z., Mao, S., Periaswamy, S. C. G., Patton, J., and Xia, X. (2020). Iadrl: Imitation Augmented Deep Reinforcement Learning Enabled Ugv-Uav Coalition for Tasking in Complex Environments. *IEEE Access* 8, 102335–102347. doi:10.1109/ACCESS.2020.2997304

Zhou, C., Wang, L., Yu, S., and He, H. (2020). "Autonomous Surface Vessel Obstacle Avoidance Based on Hierarchical Reinforcement Learning," in *Offshore Technology* (American Society of Mechanical Engineers (ASME)), 1. doi:10.1115/omae2020-18454

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., et al. (2017). "Target-driven Visual Navigation in Indoor Scenes Using Deep Reinforcement Learning," in 2017 IEEE international conference on robotics and automation (ICRAIEEE), 3357–3364. doi:10.1109/icra.2017.7989381

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.