Monocular Camera and Single-Beam Sonar-Based Underwater Collision-Free Navigation with Domain Randomization

Pengzhi Yang[1⋆], Haowen Liu[2⋆], Monika Roznere[2], and Alberto Quattrini Li[2]

[1] University of Electronic Science and Technology of China, Sichuan, China, 610056
peyang@alu.uestc.edu.cn
[2] Dartmouth College, Hanover, NH, USA, 03755,
{haowen.liu.gr, monika.roznere.gr, alberto.quattrini.li}@dartmouth.edu

Abstract. Underwater navigation presents several challenges, including unstructured unknown environments, lack of reliable localization systems (e.g., GPS), and poor visibility. Furthermore, good-quality obstacle detection sensors for underwater robots are scant and costly; and many sensors like RGB-D cameras and LiDAR only work in-air. To enable reliable mapless underwater navigation despite these challenges, we propose a low-cost end-to-end navigation system, based on a monocular camera and a fixed single-beam echo-sounder, that efficiently navigates an underwater robot to waypoints while avoiding nearby obstacles. Our proposed method is based on Proximal Policy Optimization (PPO), which takes as input current relative goal information, estimated depth images, echo-sounder readings, and previous executed actions, and outputs 3D robot actions in a normalized scale. End-to-end training was done in simulation, where we adopted domain randomization (varying underwater conditions and visibility) to learn a robust policy against noise and changes in visibility conditions. The experiments in simulation and real-world demonstrated that our proposed method is successful and resilient in navigating a low-cost underwater robot in unknown underwater environments. The implementation is made publicly available at https://github.com/dartmouthrobotics/deeprl-uw-robot-navigation.

Keywords: monocular camera and sonar-based 3D underwater navigation, low-cost AUV, deep reinforcement learning, domain randomization

1 Introduction

This paper presents an integrated deep-learning-based system, contingent on monocular images and fixed single-beam echo-sounder (SBES) measurements, for navigating an underwater robot in unknown 3D environments with obstacles.

⋆ Authors contributed equally.

Presented at the International Symposium of Robotics Research (ISRR) 2022.
DOI: 10.1007/978-3-031-25555-7_7

Obstacle avoidance is fundamental for Autonomous Underwater Vehicles (AUVs) to safely explore the largely unmapped underwater realms (e.g., coral reefs, shipwrecks). However, the underwater environment itself poses unique challenges in regards to safe navigation, which is still an open problem for AUVs [1]. There are limited sensors and positioning systems (e.g., GPS) that accurately measure the surroundings and operate underwater, thus preventing the use of well-established navigation methods [2] that were originally designed for ground vehicles with sensors like LiDAR. In addition, the sensor configurations in low-cost AUVs, equipped with monocular camera, inexpensive IMU, compass, and fixed SBES, bear their own individual drawbacks, such as no scale information and drifting/uncertain measurements. These challenges make the classic methods for obstacle avoidance and navigation in unknown environments – i.e., those which (1) estimate the geometry of the space using sensors with direct [3] or indirect [4,5] state estimation methods and (2) apply specific behaviors or planning in the partial map (e.g., Vector Field Histogram [6], Dynamic Window Approach [7]) – not directly applicable in underwater scenarios.

With recent advances in deep reinforcement learning (DRL) [8, 9], several end-to-end deep neural network based methods have emerged, from raw images to control outputs. These end-to-end methods – typically tasked to endlessly navigate or reach a visual target – demonstrated good performance for ground robots in unknown environments [10]. Comparatively, underwater domains bring problems to learning-based vision navigation due to a more complex image formation model that results in, e.g., backscattering and light attenuation.

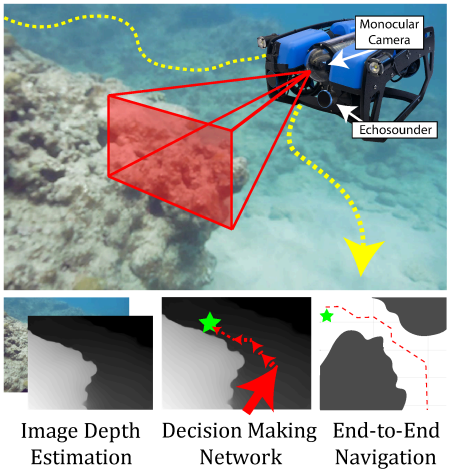

Fig. 1. How to guide an underwater robot to 3D waypoints given only monocular images, fixed echo-sounder range measurements, and a localization system, but *no map*, while also avoiding obstacles?

This paper proposes a goal-oriented end-to-end DRL navigation approach, given that classical planning methods are not straightforward to apply as they require accurate maps, which are difficult to obtain due to the underwater perception challenges described above. In particular, we design the first multi-modal end-to-end underwater navigation system in unstructured 3D environments for which no map is available, based on Proximal Policy Optimization (PPO) [11], which allows for continuous action space. The provided inputs are goal positions, estimated depth images, and range measurements from the fixed SBES. Monocular camera and fixed SBES keep the AUV's cost low, while exploiting and complementing the individual sensor' strengths – i.e., large field of view from the monocular camera that can provide relative scene depth and the absolute range measurement from the SBES. We also propose a method to mitigate the

sim-to-real gap problem by leveraging domain randomization into our system. We generated realistic simulated environments with different underwater visibility and randomized training environments, enhancing the model robustness to the changing visual conditions in real underwater domain. Extensive experimental analysis with tests and ablation studies of the proposed navigation system were conducted both in simulation and real-world. Results demonstrated high safety and efficiency compared to traditional navigation baselines and other sensor/model configurations, as well as reliable transferability to new environments.

2 Related Work

Obstacle avoidance and navigation without a prior map has been studied starting with wheeled mobile robots equipped with bumpers and sonar sensors [12] and later branching off into different environments and sensor configurations. For underwater domains, one of the main challenges is the limit of choices for sensors. While some underwater LiDAR solutions are available [13], they are expensive (US$100,000 or more) and bulky – requiring a laser scanner and a camera. In addition, there is a lack of global positioning systems and the acoustic based positioning systems are affected by noise, making mapping underwater challenging [1]. Our goal is to enable navigation for low-cost AUVs. Therefore, in the following, we discuss applications using sensors (i.e., SBES, cameras) that are typically configured on low-cost underwater robots.

In practice, many underwater navigation systems depend on acoustic, inertial, and magnetic sensors [14–16]. For example, Calado *et al.* [17] proposed a method where the robot used a SBES to detect obstacles and construct a map of them. However, SBES can only provide a fixed single distance measurement and has high uncertainty given the wide beam cone – around 30°. To infer more about the complex scene, the robot must frequently turn in multiple directions, which negatively affects navigation efficiency. Alternatively, multi-beam and mechanical scanning sonars can cover a larger field of view [18]. Hernández *et al.* [19] used a multi-beam sonar to simultaneously build an occupancy map of the environment and generate collision-free paths to the goals. Grefstad *et al.* [20] proposed a navigation and collision avoidance method using a mechanically scanning sonar for obstacle detection. However, a scanning sonar takes a few seconds to scan a 360° view. The acoustic sensors' accuracy depends on the environment structure and the type of reflections that arise. In addition, multi-beam and mechanical scanning sonars are significantly more expensive than monocular cameras and SBES (in the order of >US$10k vs. US$10 - US$100).

While cameras have shown to provide dense real-time information about the surroundings out of the water [21], there are fewer underwater obstacle avoidance methods that use cameras. The underwater domain indeed poses significant challenges, including light attenuation and scattering. Most work considers reactive controls, i.e., no goal is specified. Rodríguez-Teiles *et al.* [22] segmented RGB images to determine the direction for escape. Drews-Jr *et al.* [23] estimated a relative depth using the underwater dark channel prior and used that

estimated information to determine the action. There has been recent efforts in 3D trajectory optimization for underwater robots. Xanthidis *et al.* [24] proposed a navigation framework for AUV planning in cases when a map is known or when a point cloud provided by a visual-inertial SLAM system [5] is available. Our proposed method navigates the robot to 3D waypoints without explicit representation of the environment.

Recently, deep learning (DL) methods have shown to work well with underwater robots. Manderson *et al.* [25] proposed a convolutional neural network that takes input RGB images and outputs unscaled, relative path changes for AUV driving. The network was trained with human-labeled data with each image associated with desired changes in yaw and/or pitch to avoid obstacles and explore interesting regions. Later it was extended with a conditional-learning based method for navigating to sparse waypoints, while covering informative trajectories and avoiding obstacles [26]. Our proposed method does not require human-labeled data.

Amidst the progress in DRL, there is more research on robots operating out of water with monocular cameras. Some of these methods addressed the problem of safe endless 2D navigation without specifying any target location. Xie *et al.* [27] trained a Double Deep Q-network to avoid obstacles in simulated worlds and tested it on a wheeled robot. Kahn *et al.* [28] proposed a generalized computation graph for robot navigation that can be trained with fewer samples by subsuming value-based model-free and model-based learning. Other works provided the goal as a target image instead of a location [29–31]. Some methods, based on an end-to-end network, guided the robot to the goal using LiDAR or RGB-D cameras [10, 32–34] and goal's relative position for path planning. Recently, a DD-PPO based method was used to navigate a robot in an unknown indoor (simulated) environment, using a RGB-D camera, GPS, and compass [11]. Our method will be based on PPO, with the additional challenge of not having depth information directly from the camera.

Nevertheless, due to the difficulties of applying DRL in real-world environments, most works performed training in simulation. However, policies learned in simulated environments may not transfer well to the real-world environment, due to the existence of reality (sim-to-real) gap [35]. To address this, several methods utilized domain randomization, where parameters of the simulated world were varied so that policies learned remained robust in real-world domain. For example, Sadeghi and Levine [36] proposed a DRL approach for indoor flight collision avoidance trained only in CAD simulation that was able to generalize to the real world by highly randomizing the simulator's rendering settings.

Our approach draws from the advances in DRL: we design an end-to-end pipeline for low-cost underwater robot navigation to address the underwater challenges, combining multiple sensors and applying domain randomization.

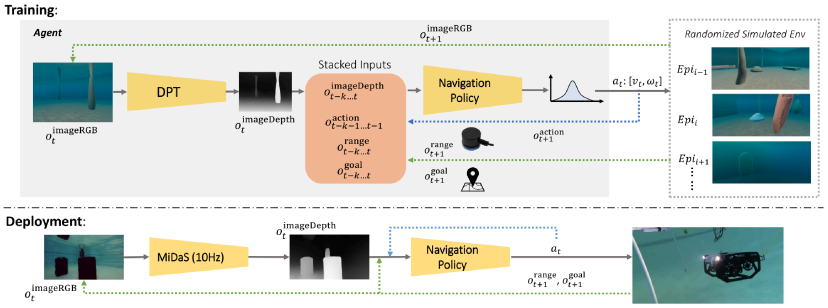

Fig. 2. *Flowchart for the Proposed End-to-End Underwater 3D Navigation System.* The pipeline includes two stages: a depth prediction module (DPT) followed by a decision making module (PPO). During training, at each episode i, the robot is deployed in a randomized simulated environment. Predicted depth map $o_t^{\text{imageDepth}}$ of the raw RGB image o_t^{imageRGB}, relative goal position o_t^{goal}, echo-sounder reading o_t^{range}, and previous executed action a_{t-1} are stacked with past k observations from the previous times steps to feed into the PPO network (solid lines). The robot performs the action sampled from the output policy distribution. New observations (dashed lines) are then obtained for computing the next action at time step $t+1$. During real-world deployment, DPT's computationally less expensive counterpart MiDaS was used as the depth prediction module for real-time inference.

3 Approach

The problem considered in this paper is as follows: an underwater robot deployed in an unknown environment needs to navigate to a goal location $G \in \mathbb{R}^3$, minimizing the travel time, while avoiding collisions with obstacles.

To develop a mapless navigation solution for low-cost robots, we consider an underwater thruster-vectored robot that has an inexpensive sensor suite composed of: (1) a monocular camera, (2) a SBES placed below the camera and looking forward, (3) a compass, (4) pressure sensor for water depth, and (5) a (noisy) localization system. Selecting this sensor configuration allows us to exploit the larger field of view (FOV) covered by the camera while obtaining absolute front distance estimates with the fixed SBES.

For a general solution, robust to noise and changing visual conditions, we approach the real-time 3D navigation problem by devising an end-to-end system (see Fig. 2) based on a neural network for dense depth prediction from monocular images and on a deep reinforcement learning method that takes as input the sensor suite data and outputs vertical and steering commands. We consider a window of prior measurements and executed actions given the absence of prior knowledge of the environment.

In the remainder of this section, we describe in detail the RL approach, the depth prediction network, and how to address the sim-to-real gap.

3.1 Multi-Modal Deep Reinforcement Learning Navigation

Given an unknown environment, the navigation problem can be formulated as a Partially Observable Markov Decision Process (POMDP), defined with a 6-

tuple: state space S that cannot be directly observed by the robot, action space A modifying the current state of the robot, observation space Ω, a state-transition model T, the observation probability distribution O, and a reward function R which returns the reward after a state transition.

Observation space. The observation O_t at time step t consists of: (1) the predicted depth image $o_t^{\text{imageDepth}} \in \mathbb{R}^{128\times160}$; (2) an SBES range measurement $o_t^{\text{range}} \in \mathbb{R}$; (3) the current relative goal position $o_t^{\text{goal}} \in \mathbb{R}^3$ – specifically, $[D_t^h, D_t^v, \theta_t^h]^\top$, where D_t^h, D_t^v are robot's current horizontal, vertical distances to the goal and θ_t^h represents the relative yaw heading difference; and (4) the past executed actions $o_t^{\text{action}} \in \mathbb{R}^2$. We stack observations considering a time window k to capture the robot's progress towards the goal and to avoid obstacles that left the periphery view. In experiments, model using 5 time steps (decision period lasts 0.5 second for each step) showed good performance without adding too much computational expense.

Action space. The action space is $a_t = [v_t, \omega_t] \in \mathbb{R}^2$, where v_t is the vertical linear velocity and ω_t is the yaw angular velocity. To generalize the applicability of the learned behavior to different robots, we consider the actions to be in a range of $[-1.0, 1.0]$ which will be linearly mapped to the range of velocities of a specific robot. Note that while we could include the horizontal forward linear velocity, we decided to keep it constant to facilitate surveying missions that require the same velocity to collect consistent high-quality measurements.

The action is then given by the policy:

$$a_t = \pi(O_t) \tag{1}$$

The goal is to find the optimal policy π^* which maximizes the navigation policy's expected return over a sequence τ of observations, actions, and rewards:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{r\sim p(\tau|\pi)}\Big[\sum \gamma^t r_t\Big] \tag{2}$$

where $\gamma \in [0, 1.0]$ is the discount factor. The optimal policy would translate in a path that is safe and minimizes the time it takes to travel to the goal.

Reward function. Our reward function r_t at time t encodes the objectives to stay not too close to any obstacle (r_t^{obs}) and to reach the goal area as soon as possible (r_t^{goal}).

When the robot is close to an obstacle, it will compute a negative reward:

$$r_t^{\text{obs}} = \begin{cases} -r_{\text{crash}}, & d_t^h < \delta_h \vee d_t^v < \delta_v \vee d_t^{\text{sur}} < \delta_v \\ -s_0(2\delta_h - d_t^h), & \delta_h \le d_t^h < 2\delta_h \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where δ_h, δ_v represent the thresholds for the distances of the robot to the closest obstacle d_t^h, d_t^v – horizontally or vertically, respectively. We also check the distance to the water surface d_t^{sur}, as there might be surface obstacles that cannot be detected given the sensor configuration of the robot. The threshold values δ_h, δ_v should consider the robot's size and turning radius. When any of the constraints are met – i.e., the robot is too close to an obstacle or the surface – the current

episode terminates with a large negative constant reward $-r_{\text{crash}}$. In addition, to guarantee safety, a penalty for motions within a range $[\delta_h, 2\delta_h)$ of distance to nearby obstacles is given according to the current distance. Otherwise, if the robot is far from the obstacles, no negative reward is applied.

To guide the robot towards the goal both horizontally and vertically, we split the goal-based reward into two parts. First, the horizontal goal-based reward:

$$r_t^{\text{goalh}} = \begin{cases} -s_1|\theta_t^h|, & \Delta_h < D_t^h \\ r_{\text{success}} - s_2|\theta_t^h|, & \text{otherwise} \end{cases} \tag{4}$$

If the robot's horizontal distance to the goal D_h^t is greater than a threshold Δ_h, then the penalty is based on the robot's orientation to the goal – i.e., a robot already facing the goal gets a smaller penalty, as the constant forward velocity will ensure shorter arrival time. Otherwise, if the robot is within the goal area, then there is a positive reward with a preference to the robot's orientation towards the goal.

Likewise, the vertical goal-based reward:

$$r_t^{\text{goalv}} = \begin{cases} s_3|\dot{D}_t^v|, & \dot{D}_t^v \leq 0 \wedge \ \Delta_h < D_t^h \\ -s_3|\dot{D}_t^v|, & \dot{D}_t^v > 0 \wedge \ \Delta_h < D_t^h \\ -s_4|D_t^v|, & \text{otherwise} \end{cases} \tag{5}$$

When the robot is not near the goal, the vertical goal-based reward is a positive value if the change in vertical distance over time $\dot{D}_t^v$ is negative or 0 – i.e., the robot is getting closer to the target depth. On the contrary, it is a negative value if the change is positive – i.e., the robot is getting farther from the target depth. Otherwise, if the robot is within goal area, the negative reward is relative to the distance to the target depth. This split (horizontal and vertical) of the goal reward showed better stability in experiments than when a single combined goal reward was applied, potentially due to the separate focus of two mostly independent actions.

The above obstacle- and goal-based rewards conflict with each other; they could lead to oscillations at local optima when an obstacle is nearby. Thus, we devised a priority-based strategy (when the robot is not in the goal area) that focuses on moving away from the obstacle by scaling r_t^{goalh}:

$$r_t^{\text{goalh}} *= s_5(d_t^h - \delta_h)/\delta_h, \ \Delta_h < D_t^h \wedge \delta_h \leq d_t^h < 2\delta_h \tag{6}$$

In all the reward equations, $s_0, \ldots, s_5$ are positive scaling factors. Intuitively, they are set so that rewards are in an appropriate scale for a balanced training performance.

Finally, the collective reward at time t can be obtained as:

$$r_t = r_t^{\text{obs}} + r_t^{\text{goalh}} + r_t^{\text{goalv}} \tag{7}$$

Network architecture. The network structure depicted in Fig. 3(left) illustrates how we integrate the information vectors from the sensors. First, the stacked predicted depth images are processed by three convolutional layers, then

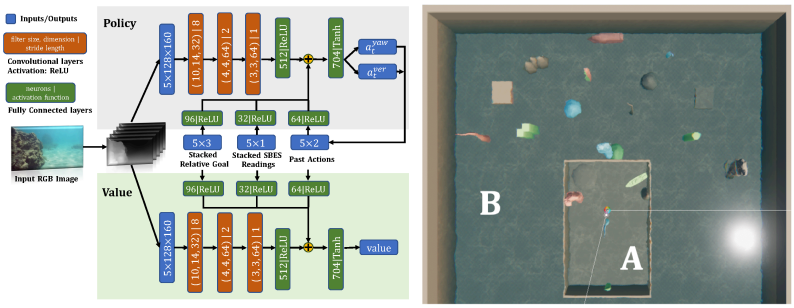

Fig. 3. (left) *Network Architecture.* Predicted depth images are processed by three layers of convolutional layers (orange). Its output is flattened and concatenated with feature vectors (green) representing the stacked relative goal positions, echo-sounder readings, and past actions. The final fully-connected layer outputs a navigation policy and state value. (right) *Top View of the Training Env.* Our model was trained in the above simulated environment in area A (inside area with fewer obstacles and smaller space) and B (outside area with more obstacles and larger space).

the flattened output $\in \mathbb{R}^{512}$ is concatenated with processed feature vectors consisting of the stacked relative goal positions $\in \mathbb{R}^{96}$, SBES readings $\in \mathbb{R}^{32}$, and past actions $\in \mathbb{R}^{64}$. Specifically, the combined echo-sounder readings provide an implicit scale on the relative depth prediction without requiring calibration. The network will produce a navigation policy and state value.

3.2 Image Depth Prediction Network

Accurate image depth predictions is important for our navigation pipeline to work. Previous work used ground truth simulated depth images with Gaussian noise as input for training and applied depth estimation during deployment [27]. However, this broadens the sim-to-real gap as real-world noise in depth predictions is more complex than implemented simulated noise models [37]. Instead, we utilized one of the latest monocular depth prediction networks, Dense Prediction Transformer (DPT) [38], which has an encoder-decoder design and applies a transformer as the encoder's main building block. We selected DPT over other deep neural networks for depth prediction for its state-of-the-art performance in single-view depth estimation and robustness across diverse environments.

3.3 Transferable Model

DRL often has the problem of generalization: models trained in one domain fail to transfer to other domains even if there are small differences between the domains [39]. Unlike in-air, images taken underwater will look drastically different across various environments due to the more complex lighting and backscattering effects [40]. Thus, training the model in a single fixed environment would lead to over-fitting to that environment's visual conditions. One solution is to retrain the depth prediction network with an existing underwater image depth

dataset, which, however, is not available. Another solution is to enhance the input underwater images to its approximate in-air counterpart [40, 41]. Yet, most image enhancement techniques require difficult-to-retrieve information (e.g., water attenuation coefficients, depth maps).

Our approach is to integrate underwater features into the simulation used for training. We modified an existing underwater simulator framework for games to create the training and testing simulations for our proposed approach. The framework contains custom shaders that incorporates a light transmission model to simulate underwater optical effects, thus providing a good amount of realism.

Domain randomization. We integrated domain randomization to generate underwater environments with different visual conditions, thus enabling transferability. In particular, at the start of every training episode, we randomize the underwater visibility – the gradient and conditions in visibility over distance. Visibility was selected as it significantly impacts the relative depth estimation, thus affecting to a large extent how the robot perceives its surroundings.

We decided not to apply domain adaptation [42] – i.e., the process of learning different environment encoding and corresponding adapted policy during training, so that during testing the best environment encoding will be found with the corresponding adapted policy – because searching the best environment encoding is not very practical for underwater deployments. For instance, the search would require robot motions towards obstacles to identify the (potentially changing) visibility feature of the specific environment.

Multi-scenario training. We built the simulated training environment via Unity Engine[3]. We generated two activity areas to represent two classes of environments that an AUV might encounter: A – a small area with fewer obstacles, and B – a big cluttered area with obstacles at various positions and heights (see Fig. 3(right)). In each training episode, the robot's starting pose and goal location are randomly reset in the environment. This exposure to different training scenarios ensures that the learned policy will be more likely to handle more complex environments [35].

4 Experimental Results

We trained and performed experiments in simulation, in real-world with a vector-thruster underwater robot, and with underwater datasets to validate our DRL-based multi-modal sensor navigation system. We performed comparisons and ablation studies with other methods. Our framework is publicly available[4].

4.1 Training Experimental Settings

Our model was first trained and tested on a workstation with two 12GB NVIDIA 2080Ti GPUs. It was implemented with PyTorch and Adam optimizer [43].

[3] http://www.unity.com/

[4] https://github.com/dartmouthrobotics/deeprl-uw-robot-navigation

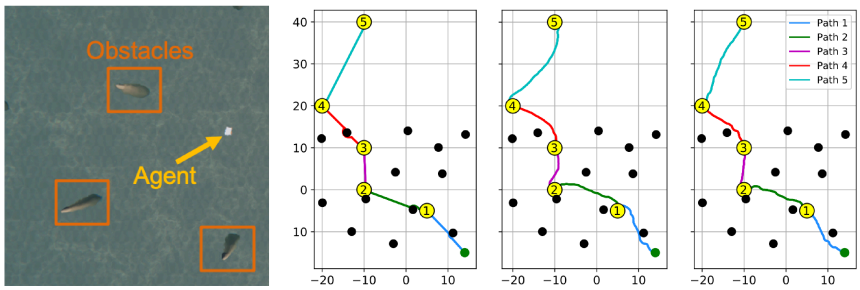

Fig. 4. *Partial top view of runs in Cluttered Env. (left): Bug2 (second), Our Model w/o SBES (third), and Our Model w/ SBES (right).* Legend: robot's start pose (green dot); obstacles (black dots); waypoints to reach in order (circled numbers).

In simulation, the robot's forward velocity, vertical velocity range, and yaw angular velocity range were set to 0.345 m/s, $(-0.23,0.23)$ m/s, $(-\pi/6,\pi/6)$ rad/s, respectively. While the training environment allows for higher velocities, we chose low velocities to avoid any "jerky" motion that could happen with the AUV at high speed. The camera's horizontal and vertical FOVs were set to 80° and 64°. The simulated echo-sounder's max detection range was set to 4 m, which are all consistent with the real-world sensor configuration. The simulation environments' visibility value was randomly chosen within the range of (3,39) m.

We trained for 250 iterations – each with at least 2048 time steps – and observed the reward was stable after around 120 iterations (learning rate of 3e-5). The detailed constant and threshold values for the reward function – i.e., r_{success}, r_{crash}, Δ_h, δ_h, and δ_v – were set to 10, 10, 0.6 m, 0.5 m and 0.3 m, while the scaling factors $s_0, s_1, \ldots, s_5$ were set to 2.0, 0.1, 1.0, 1.0, 8.0, 1.0.

4.2 Performance Comparison with Different Sensor Configurations

We first tested the efficiency of our proposed multi-modal low-cost navigation approach against a traditional metric-based goal-oriented navigation method that does not require any map, given that no map of the underwater environment is available. In particular, we selected Bug2 algorithm given its guarantees on the path length. To have Bug2 work effectively, we employed a multi-beam sonar (MBS), a common but expensive sensor for underwater obstacle avoidance, which emits multiple beams in a plane with a typical horizontal FOV of 120°. We also

Table 1. *Waypoint Tests Results.* 10 runs for each of the three methods: Bug2 with multi-beam sonar, our model trained without fixed single-beam echo-sounder, and our proposed model. The travel time average and standard deviation (in seconds) of successful runs for each waypoint were calculated, as well as the overall success ratio to reach all five waypoints.

Method	Sensors	Traveling Time/s (less is better)					Success Ratio
		wp1	*wp2*	*wp3*	*wp4*	*wp5*	(higher is better)
Bug2	MBS	57.6 ± 0.3	66.95 ± 0.15	41.15 ± 0.45	69.8 ± 0.9	77.65 ± 0.45	**100%**
Ours w/o SBES	Monocular Camera	51.8 ± 5.94	56.5 ± 2.09	35.62 ± 8.07	47.0 ± 2.03	76.0 ± 2.21	40%
Ours w/ SBES	Monocular Camera & SBES	**38.35 ± 0.45**	**49.8 ± 0.78**	**29.3 ± 0.78**	**44.3 ± 0.6**	**67.25 ± 0.6**	**100%**

considered our model trained without the echo-sounder as ablation study to observe the effect of the SBES.

We generated a test environment in simulation with multiple obstacles. The robot's task was to navigate to five randomly set consecutive waypoints. We set all waypoints at the same depth, as typical navigation with an MBS involves the robot first arriving to the target depth and then navigating along the 2D plane.

Fig. 4 shows the trajectories of the three navigation methods and Table 1 reports the quantitative results measured in terms of traveling time and success ratio. Our proposed system with inexpensive monocular camera and SBES achieved the highest navigation efficiency with comparable safety to Bug2 with MBS. While the Bug2 trajectory appeared not to be affected by noise, it spent the longest navigation time especially when moving along the obstacles. Note the echo-sounder played a fundamental role in safe navigation. If the echo-sounder was excluded, the model relied solely on relative monocular image depth estimation to detect surrounding obstacles. As a result, at times the chosen action might be conservative, leading to sub-optimal paths in terms of distance, or too aggressive, increasing the likelihood of collision.

4.3 Ablation Study with Transferability Tests

To show the transferability of our proposed model to different environments and visibilities, we performed an ablation study with the same hyper-parameters and protocols, but considering the following combinations of training settings in a simulated underwater environment: (1) ***Rand***: proposed domain randomization, (2) ***No Rand (Water)***: fixed underwater visibility (approximately 11 m), and (3) ***No Rand (Air)***: no underwater features. To firstly exhibit the models' generalizability, another simulated environment[5] was employed for testing. With different materials, textures, lightings and custom shaders, it had a different visual appearance compared to the training environment. In this environment, the models were tested in three different scenes, constructed to resemble possible underwater obstacles present in the real-world, such as natural structures (Scene1), submerged wrecks (Scene2) and man-made structures (Scene3).

Table 2. *Quantitative Results for Transferability Tests.* 10 runs for the three models in three scenes with different visual conditions. Note: N/A means the method failed to reach the goal during the runs and bold means the best result.

Method		Scene1			Scene2			Scene3		
		Blurry	Medium	Clear	Blurry	Medium	Clear	Blurry	Medium	Clear
No Rand (Air)	reward	5.74 ± 2.17	6.5 ± 5.95	28.14 ± 2.85	0.43 ± 2.26	10.93 ± 11.31	12.05 ± 8.92	24.64 ± 10.19	20.58 ± 13.7	29.18 ± 8.01
	success	0%	10%	**100%**	0%	40%	50%	70%	60%	90%
	trav. time	N/A	70.0	67.2 ± 0.84	N/A	**53.12 ± 0.65**	55.2 ± 2.84	63.29 ± 0.88	66.5 ± 4.53	66.11 ± 1.07
No Rand (Water)	reward	**25.27 ± 8.42**	18.35 ± 11.18	13.46 ± 14.51	2.19 ± 1.78	-1.58 ± 5.94	15.04 ± 10.6	18.03 ± 11.32	30.14 ± 7.5	29.42 ± 3.27
	success	**90%**	90%	40%	0%	10%	70%	60%	**90%**	**100%**
	trav. time	70.5 ± 4.93	88.17 ± 18.36	69.25 ± 1.35	N/A	115.0	59.79 ± 8.25	71.42 ± 6.9	73.39 ± 2.63	65.35 ± 0.78
Rand	reward	24.66 ± 9.3	**28.39 ± 2.26**	**29.56 ± 2.58**	**21.68 ± 9.61**	**23.36 ± 7.49**	**24.86 ± 2.92**	**29.17 ± 11.34**	**30.26 ± 9.25**	**36.26 ± 0.83**
	success	**90%**	**100%**	**100%**	**80%**	**90%**	**100%**	**80%**	**90%**	**100%**
	trav. time	**67.56 ± 0.44**	**68.45 ± 0.72**	**67.05 ± 1.27**	**52.0 ± 0.35**	53.44 ± 1.23	**50.75 ± 0.46**	**60.75 ± 0.56**	**62.56 ± 0.98**	**61.05 ± 0.57**

[5] https://github.com/Scrawk/Ceto

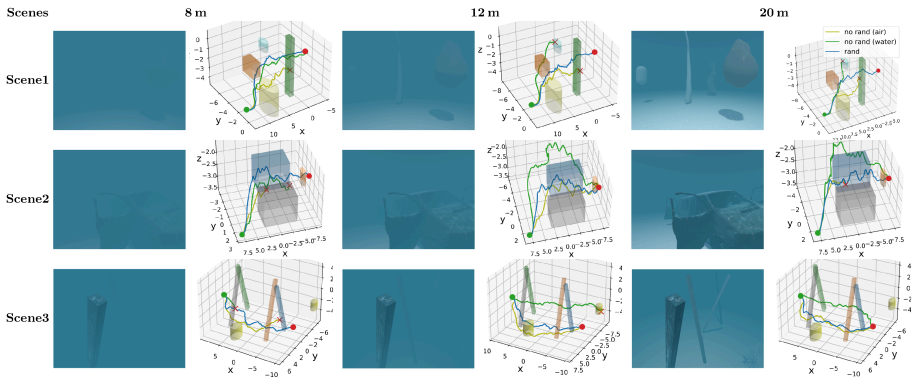

Fig. 5. *Example of Trajectories in Different Scenes with Different Training.* Legend: robot's initial position and goal waypoint (green and red dots); robot collision (red "X"); obstacles (approximated with polygons in the plots for simplicity).

We considered three visibility scenarios: blurry, medium, and relatively clear, with maximum visibility ranges of 8 m, 12 m, and 20 m, respectively. Fig. 5 shows snapshots of each scene and the resulting trajectories in some sample runs.

Comparison metrics. The following metrics were used to compare the three methods' performances (see Table 2):

1) Rewards (higher is better): cumulative reward average and standard deviation over 10 runs,
2) Success Ratio (higher is better): number of times the robot reached the goal with no collision over 10 runs,
3) Travel Time (less is better): average and standard deviation traveling time (s). Failed runs were not considered.

From the results, training with underwater features has the highest gain. Adding domain randomization allows a further increase of the cumulative rewards, success rate, and travel time. Models trained without randomization did not previously encounter abundant visual conditions, thus explored a limited observation space. Accordingly, they would not be easily applicable to different visibility conditions and are more vulnerable to noise especially in low-visibility environments when depth estimations are inaccurate. Scene3 in particular was challenging with blurry visibility, due to the narrow passage between the logs.

4.4 Performance Demonstration in Real-World Environment

We conducted real-world experiments with a BlueROV2 in a swimming pool. The robot was equipped with a Sony IMX322LQJ-C camera[6] with a resolution of 5 MP, a horizontal and vertical FOV of 80° and 64°. The fixed SBES has a 30° beam width and a maximum range set to 4 m. The (noisy) robot's pose was provided by an on-board compass, a water-pressure sensor to recover water

[6] https://www.bluerobotics.com/store/sensors-sonars-cameras/cameras/cam-usb-low-light-r1/

depth, and a short baseline acoustic positioning system (SBL)[7]. A 2.8 GHz Intel i7 laptop with Nvidia Quadro M1200 was used for running the inference network through the Robot Operating System (ROS). For real-time inference, DPT was replaced with its computationally less expensive counterpart MiDaS [44] as our depth prediction network – about 0.08 seconds per inference.

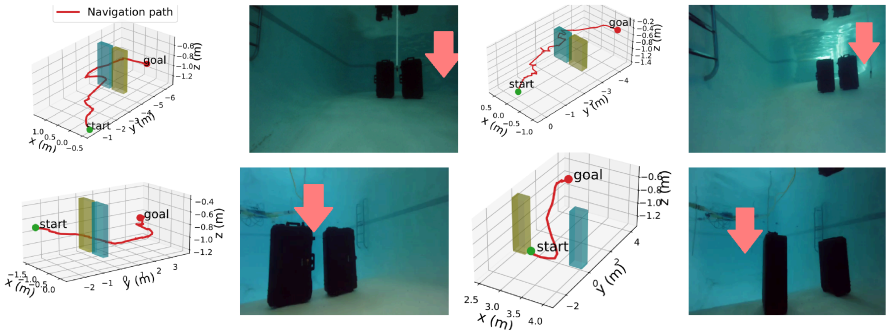

Fig. 6. *Pool Experiment.* Navigation trajectories with localization noise smoothing (legend: Start and goal green and red dots; obstacles, cuboids) and images from the robot's camera. Red arrows point to the approximate goal locations behind the boxes.

The swimming pool was about 20 m by 7 m in size with a shallow (1 m) and deep (3 m) end, and a slope in the middle. Two black boxes (approximate size: 0.8 x 0.5 x 0.3 m were placed in two different configurations: side by side as a large obstacle and with a 1 m separation to create a channel.

Resulting paths and reference images are shown in Fig. 6. Our proposed navigation approach successfully drove the BlueROV2 to different 3D waypoints, avoiding obstacles by going around, above, or through a channel (see Fig. 6). We observed that the SBL provided noisier position information compared to in simulation – at times the robot's location jumped up to a meter. While the noise affected the calculation of the relative position to the goal, our approach does not depend on the absolute robot location to infer obstacle distance, so the robot was able to avoid obstacles.

4.5 Action Prediction from Static Underwater Images

We also tested joint image and SBES reading data from past field trials (in the Caribbean Sea and lake) as input to our model for action prediction. Fig. 7 shows a sample of such images with corresponding depth predictions, locations of the goal, and predicted actions. As expected, with obstacles nearby the predicted action prioritized obstacle avoidance, steering the robot away, otherwise, the action's direction pointed towards the goal. This qualitative test demonstrates our model's generalizability to real-world applications.

[7] https://waterlinked.github.io/explorer-kit/introduction/

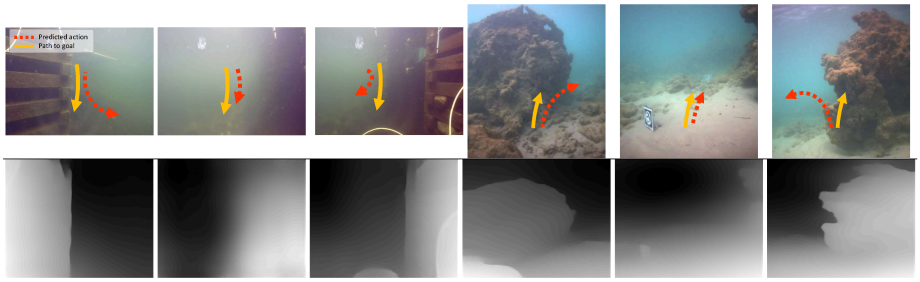

Fig. 7. *Single Image Action and Depth Prediction.* 1st row: images from Lake Sunapee and Caribbean Sea. 2nd row: their respective depth predictions. Direction and magnitude of the action predicted (red arrow); approximate goal location (yellow arrow).

5 Conclusion and Future Work

We presented the first 3D map-less underwater navigation approach, based on Proximal Policy Optimization Network (PPO) and domain randomization, for low-cost underwater robots with a monocular camera and a fixed single-beam echo-sounder. By choosing deep reinforcement learning over classic methods, we were able to address the intrinsic challenges of seamless underwater navigation (e.g., lack of low-cost efficiency sensor and difficulty in generating a map given noisy positioning and perception data). We validated our approach with several comparisons and ablation studies in different simulated environments, as well as real-world validation in a swimming pool and with static underwater images. Results showed that the robot is able to navigate to arbitrary 3D goals while avoiding obstacles inferred from estimated depth images and sonar readings.

In the future, we will investigate explicit sensor fusion of camera and SBES data to achieve better depth prediction with absolute scale, e.g. early fusion [45], as well as controller and SBL data. In addition, we will consider the generation of more complex environments, other real-world experiments, and the design of integrated models for different sensor configurations (e.g., stereo cameras) and dynamic models to adapt our method to heterogeneous underwater robots.

Acknowledgments

We thank Devin Balkcom for access to the pool for experiments, and Bo Zhu, Mary Flanagan, and Sukdith Punjasthitkul for GPU access. This work is supported in part by the Burke Research Initiation Award and NSF CNS-1919647, 2024541, 2144624, OIA-1923004.

References

1. Y. R. Petillot, G. Antonelli, G. Casalino, and F. Ferreira, "Underwater robots: From remotely operated vehicles to intervention-autonomous underwater vehicles," *IEEE Robot. Autom. Mag.*, 2019.
2. A. Pfrunder, P. V. Borges, A. R. Romero, G. Catt, and A. Elfes, "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D LiDAR," in *Proc. IROS*, 2017.

3. J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.
4. C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, 2021.
5. S. Rahman, A. Quattrini Li, and I. Rekleitis, "SVIn2: An underwater SLAM system using sonar, visual, inertial, and depth sensor," in *Proc. IROS*, 2019.
6. D. Panagou, "Motion planning and collision avoidance using navigation vector fields," in *Proc. ICRA*, 2014.
7. D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, 1997.
8. J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, 2013.
9. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.
10. L. Xie, S. Wang, S. Rosa, A. Markham, and N. Trigoni, "Learning with training wheels: speeding up training with a simple controller for deep reinforcement learning," in *Proc. ICRA*, 2018.
11. E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames," in *Proc. ICLR*, 2020.
12. H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, S. Thrun, and R. C. Arkin, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.
13. D. McLeod, J. Jacobson, M. Hardy, and C. Embry, "Autonomous inspection using an underwater 3D LiDAR," in *Proc. OCEANS*, 2013.
14. J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: Recent advances and new challenges," in *IFAC Proc. Conf. of Manuevering and Control of Marine Craft*, 2006.
15. S. B. Williams, P. Newman, J. Rosenblatt, and H. Durrant-Whyte, "Autonomous underwater navigation and control," *Robotica*, 2001.
16. L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE J. Ocean. Eng.*, 2013.
17. P. Calado, R. Gomes, M. B. Nogueira, J. Cardoso, P. Teixeira, P. B. Sujit, and J. B. Sousa, "Obstacle avoidance using echo sounder sonar," in *Proc. OCEANS*, 2011.
18. Y. Petillot, I. T. Ruiz, and D. M. Lane, "Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar," *IEEE J. Ocean. Eng.*, 2001.
19. J. D. Hernández, E. Vidal, G. Vallicrosa, E. Galceran, and M. Carreras, "Online path planning for autonomous underwater vehicles in unknown environments," in *Proc. ICRA*, 2015.
20. Ø. Grefstad and I. Schjølberg, "Navigation and collision avoidance of underwater vehicles using sonar data," in *IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, 2018, pp. 1–6.
21. F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2024–2039, 2015.

22. F. G. Rodríguez-Teiles, R. Pérez-Alcocer, A. Maldonado-Ramírez, L. A. Torres-Méndez, B. B. Dey, and E. A. Martínez-García, "Vision-based reactive autonomous navigation with obstacle avoidance: Towards a non-invasive and cautious exploration of marine habitat," in *Proc. ICRA*, 2014.
23. P. Drews-Jr, E. Hernández, E. R. Nascimento, and M. Campos, "Real-time monocular obstacle avoidance using underwater dark channel prior," in *Proc. IROS*, 2016.
24. M. Xanthidis, N. Karapetyan, H. Damron, S. Rahman, J. Johnson, A. O'Connell, J. M. O'Kane, and I. Rekleitis, "Navigation in the presence of obstacles for an agile autonomous underwater vehicle," in *Proc. ICRA*, 2020.
25. T. Manderson, J. C. G. Higuera, R. Cheng, and G. Dudek, "Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection," in *Proc. IROS*, 2018.
26. T. Manderson, J. C. Gamboa Higuera, S. Wapnick, J.-F. Tremblay, F. Shkurti, D. Meger, and G. Dudek, "Vision-based goal-conditioned policies for underwater navigation in the presence of obstacles," in *Proc. RSS*, 2020.
27. L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," in *RSS workshop on New Frontiers for Deep Learning in Robotics*, 2017.
28. G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *Proc. ICRA*, 2018.
29. Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. ICRA*, 2017.
30. A. Devo, G. Mezzetti, G. Costante, M. L. Fravolini, and P. Valigi, "Towards generalization in target-driven visual navigation by using deep reinforcement learning," *IEEE Trans. Robot.*, 2020.
31. Q. Wu, X. Gong, K. Xu, D. Manocha, J. Dong, and J. Wang, "Towards target-driven visual navigation in indoor scenes via generative imitation learning," *IEEE Robot. Autom. Lett.*, 2020.
32. M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *Proc. ICRA*, 2017.
33. J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *Proc. IROS*, 2017.
34. J. Liang, U. Patel, A. J. Sathyamoorthy, and D. Manocha, "Crowd-steer: Realtime smooth and collision-free robot navigation in densely crowded scenarios trained using high-fidelity simulation," in *Proc. IJCAI*, 2021.
35. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IROS*, 2017.
36. F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," in *Proc. RSS*, 2017.
37. C. Sweeney, G. Izatt, and R. Tedrake, "A supervised approach to predicting noise in depth images," in *Proc. ICRA*, 2019.
38. R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proc. ICCV*, 2021.
39. K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *Proc. ICML*, 2019.

40. D. Akkaynak and T. Treibitz, "A revised underwater image formation model," in *Proc. CVPR*, 2018.
41. M. Roznere and A. Quattrini Li, "Real-time model-based image color correction for underwater robots," in *Proc. IROS*, 2019.
42. X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Proc. RSS*, 2020.
43. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014.
44. R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.
45. M. Roznere and A. Quattrini Li, "Underwater monocular depth estimation using single-beam echo sounder," in *Proc. IROS*, 2020.