2021 NDIA GROUND VEHICLE SYSTEMS ENGINEERING and TECHNOLOGY SYMPOSIUM
Autonomy, Artificial Intelligence & Robotics Technical Session
August 10-12, 2021 - Novi, Michigan

Topography Dependent Path Planning using Deep Q-Learning

Eric Martinson[1], PhD, Ben Purman[1], Andy Dallas[1]

[1]Soar Technology, Ann Arbor, MI

ABSTRACT

Future autonomous combat vehicles will need to travel off-road through poorly mapped environments. Three-dimensional topography may be known only to a limited extent (e.g. coarse height), but this will likely be noisy and of limited resolution. For ground vehicles, 3D topography will impact how far ahead the vehicle can "see". Higher vantage points and clear views provide much more useful path planning data than lower vantage points and occluded views from trees and structures. The challenge is incorporating this knowledge into a path planning solution. When should the robot climb higher to get a better view or else continue moving along the shortest path predicted by current information? We investigated the use of Deep Q-Networks (DQN) to reason over this decision space, comparing performance to conventional methods. In the presence of significant sensor noise, the DQN was more successful in finding a path to the target than A for all but one type of terrain.*

Citation: *E. Martinson, B. Purman, A. Dallas, "Topography Dependent Path Planning", In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 10-12, 2021.*

1. INTRODUCTION

Planning is a challenge most commonly broken up into separate problems: local/short-range and global/long-range (Figure 1). Local planning generally includes waypoint navigation and obstacle avoidance. It consumes local sensing data, operates at a fast update rate, and plans/executes a path to the next waypoint over short distances (e.g. <20-m). Long-range planning generates waypoints for waypoint-navigation to follow. It consumes information such as distant target locations, long-range obstacle maps plus uncertainty and/or terrain data to select waypoints that local planning will follow. Whereas local planning must focus on speed and safety, long range planning can be much slower as it processes high sensing uncertainty and partial map visibility to re-plan a path to the goal each time the local-planner reaches a new waypoint. Long-range planning is the subset of planning addressed by this research.

Traditional planning solutions have focused on finding optimal paths to a target. A* [1], in particular, is well known and has spawned a number of variants focused on maintaining optimality while reducing computational cost for robotic navigation such as D* [2], and D*-lite [3]. Within a constrained search space, these traditional methods are actually very good at finding paths

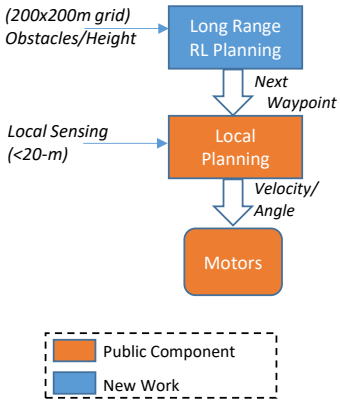

Figure 1. 3-Layer Planning Architecture for Off-road Driving

even when parts of the graph being searched are initially hidden. The real problem to traditional search is complexity. Navigation through dynamic obstacles and 3D terrains increases search complexity exponentially, and unlike greedy navigation approaches (e.g. potential fields [4]), these traditional methods may not be able to find a solution. In order to handle larger search spaces, such as 6+ degree of freedom manipulation spaces or dynamic obstacles require, new sampling based algorithms like RRT [5] were designed that no longer guarantee optimality.

Perceptual uncertainty is yet another degree of complexity not handled well by traditional planning approaches. Unlike alternative forms of spatial complexity, however, perceptual uncertainty is difficult to model as discrete nodes in a search graph. A recent solution to handle both large search graphs and perceptual uncertainty is deep learning. So called end-to-end autonomous driving solutions are a prime example, where Nvidia has adapted deep convolutional neural networks to take image inputs and generate steering/acceleration commands [6]. Deep reinforcement learning [DRL] [7] extends this idea further, applying deep neural networks to policy learning in a reinforcement learning paradigm. However, because hundreds of thousands of hours can be required to train a DRL model, end-to-end efforts have largely remained in simulation with more practical efforts striving to reduce data complexity by selecting pre-programmed maneuvers [8] and incorporating input from a human safety driver [9]. More generally, these deep learning solutions enable local planning with dynamic obstacles (e.g. cars) to follow global GPS plans.

In comparison, this work focuses on path planning for off-road autonomous driving where dynamic obstacles are much less important than navigation through uncertain environments. For this reason, we have flipped the reinforcement learning paradigm – letting alternative fast local planners such as model predictive control [10] solve waypoint navigation and using a deep Q-Network [DQN] to solve global planning. In this paper, we will demonstrate that such a DQN can manage significant perceptual uncertainty by taking advantage of a diverse 3D topography to improve perception and more successfully navigate to the goal. In the future, we will focus on reducing policy training time to practical limits using imitation learning methods [11] [12], where a (manual or semi-manual) reference policy is used to train a learned policy.

In the remainder of this work, we first present the scenario investigated to demonstrate the effectiveness of deep reinforcement learning, followed by algorithms and results. This paper concludes with a summary of future work to bring this effort to real autonomous vehicles.

2. CHALLENGE

Our planning problem focuses on solving grids. For this work, we focus on 10x10 grids with known topography, but unknown obstacles. Grid cells are assumed to represent 3-5 seconds of travel by the local planner, so at a reasonable 10 m/s (22mph) for an autonomous ground vehicle, each cell represents an area of 50x50m^2. The combined grid in this scenario covers a square of 0.5km in width.

2.1. Topography

In this space, we assume that the general topography is known, but specific obstacles that restrict movement between cells are unknown. Topographic features include: hills, valleys, and forests. Each of these features impacts the visibility of obstacles to the robot. By default, the robot can only identify the presence of obstacles in neighboring grid-cells. From that location, it has a

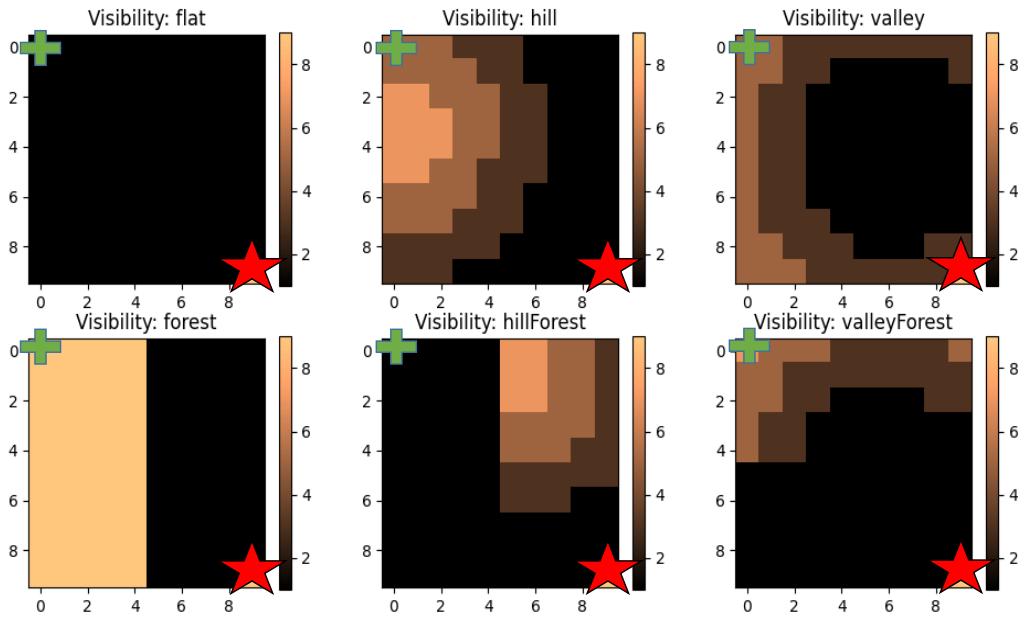

Figure 2. Impact of topographic features on the visibility of obstacles. The green crosses and red stars indicate robot start/stop positions respectively.

visibility of only 1. By climbing a hill, the robot can increase its visibility to potentially view the entire area at once. Conversely, when starting with higher initial visibility, descending into a valley, the robot's visibility is reduced.

Altogether, we explored 6 different types of topographies, each with a different impact on obstacle visibility. Each topography could be randomly varied to increase difficulty.

- **Flat** – no topographic features to increase visibility. All cells had a maximum visibility range of only 1.
- **Hill** – from the top of the hill, the entire map is visible, with a linear decrease to the bottom. The peak of the hill was randomly selected.
- **Valley** – around the valley, the robot had a maximum visibility of 5 squares, while the bottom of the valley had a visibility of only 1. The valley bottom was randomly moved.
- **Forest** – half of the map was indicated as forest with a limited visibility of only 1, while the other half was given full visibility. All four grid halves could be randomly selected.
- **Hill-Forest** – starting with a hill topography, add a forest over a random half.
- Valley-Forest – starting with a valley topography, add a forest over a random half.

2.2. Obstacles

The starting position on a map was always in [0,0], and the goal was at [9,9]. The shortest possible path to the goal without obstacles was 20. In this domain, however, obstacles represented an entire grid-cell of unpassable terrain. A robot trying to move into a cell containing an obstacle would be prevented from moving into that cell by local planning (i.e. obstacle avoidance). The location of obstacles in these grids were unknown to the robot in advance. Obstacles were generated by randomly selecting between 1-5 pairs of points on the map and drawing lines between them. Figure 3 demonstrates examples of different randomly generated sets of obstacles.

A local map tracked the presence of obstacles. Initially set to unknown for all grid cells, obstacles were detected by the robot upon moving into a cell, or after failing to move into another cell. The visibility range of that cell was used to update all cells within a Manhattan distance less than the cell visibility range.

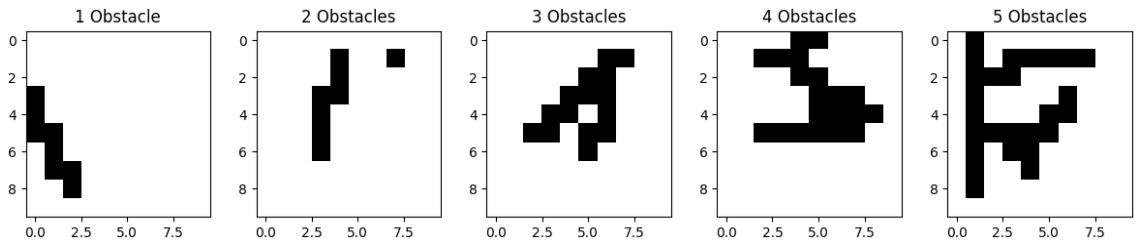

Figure 3. Examples of randomly generated obstacle maps.

2.3. Sensor Noise

For each update, including cells without obstacles, sensor noise was approximated by probabilistically flipping each return with a 0-30% probability depending upon the experiment being conducted. The local grid counted the number and type of detections to vote for the most likely candidate (obstacle, clear, unknown). Because sensor noise was always <50%, the more frequently a cell was viewed by the robot, the more likely that the cell was modeled correctly in the local map.

3. ALGORITHMS

To plan paths through highly uncertain 2.5D environments, we have developed a Deep Q-learning Network [DQN] that consumes a 3D gridded input consisting of:

(1) Expected visibility from all grid cells (determined by a topographic map)
(2) The detected class of each cell (uncertain obstacle, obstacle, clear, visited, and current location).
(3) The previous and current location of the robot.

Figure 4 [left] demonstrates a gridded input to the DQN showing the path of the robot (purple), clear space (blue), obstacles (black), and visibility (green).

This 3D input is passed to a network consisting of 3 convolutional layers and two fully connected layers (Figure 4) to select one of 4 actions (up, down, left, right) to indicate the cell in the grid containing the next waypoint location. In the local/long-range planning architecture (Figure 1), this cell center serves as the waypoint for local planning. The DQN can be executed on-demand (e.g. after reaching the target) or whenever new sensor data are ingested to ultimately guide the robot to the long-range target.

3.1. Network Training

The DQN is trained with 30000 simulated path planning problems to learn the target topography. For each new game, the topography and obstacles are randomly generated to create a new map. Training was initially conducted with 1-3 lines of obstacles, gradually increasing to 5 randomly

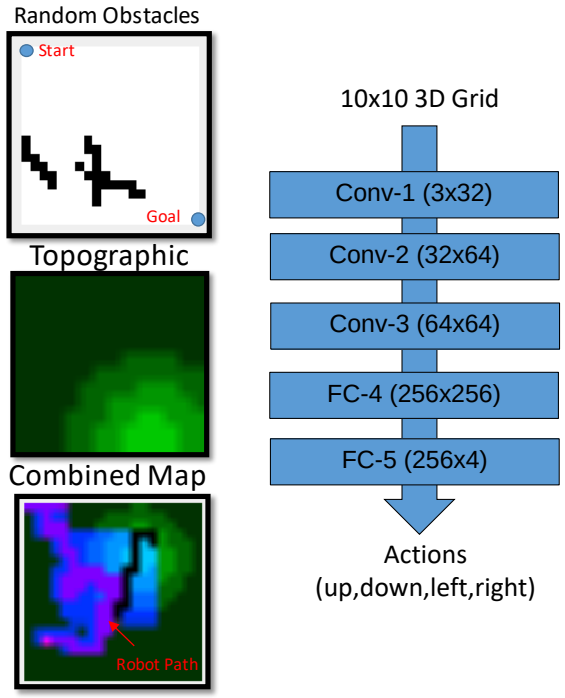

Figure 4.(Left) Example obstacle, topographic, and combined maps used for navigation (Right) DQN architecture consisting of 2D convolutional (Conv) and fully connected (FC) layers.

generated lines as network performance improved. Actions within the game are rewarded as follows:

1) Large positive reward for reaching the target location (+1)
2) Large negative reward for selecting a grid cell that contains an obstacle (-0.75)
3) Large negative reward for failing to reach the target (remaining distance – 1).
4) Mild negative reward for re-entering its own path (-0.05)
5) Mild negative reward for each movement (-0.01) to minimize the distance traveled.

As per the classic reinforcement learning work [citation], actions are selected with an epsilon-greedy [citation] policy – taking the action likely to generate the highest future reward except for a percentage (epsilon) of the time where a random action is selected. Epsilon decreases gradually over time from 100% to 10%. During testing, we continue to use an epsilon of 10% to avoid local minima.

In addition to the epsilon greedy action selection policy taken from [citation], we also use memory re-play to speed up network training – re-exposing the network to previous runs rather than simply updating it with the latest data.

3.2. A Bootstrapping*

As part of a separate effort to speed up training convergence times, A* is used to provide initial paths to the goal as training data. These initial paths, representing ~300 games, provide immediate positive reward to the DQN. With memory replay enabled, these games are preserved and stored alongside the epsilon greedy action selection result to be used in updating the network until it starts identifying its own paths to the goal. Without the A* bootstrapping, it can take thousands of games before the robot regular finds a successful path.

4. RESULTS

A DQN was trained for each of the 6 different topographies and 4 different amounts of sensor noise [0%,10%,20%,30%]. For each topography and noise level investigated, the resulting DQN model was compared to a sequential A* baseline algorithm on a set of 500 randomly generated games. With 10x10 maps, robots were given a maximum of 100 turns to reach the goal. Three metrics were evaluated:

1) Success Rate – in what percentage of games did the robot reach the goal?
2) Path Length – what was the average path length across all 500 games, including those games where the robot failed to reach the goal?
3) # of Observed Cells – how many cells did the robot classify (occupied/clear) at least once during its trip?

In terms of success rate, the highest average performance was demonstrated by A* with zero sensing noise (Figure 6). With any level of sensing noise greater than zero, the DQN outperformed A*, improving success rates by 10% with 30% sensing noise on average. This difference in success rates varied significantly between different topographies (Figure 7). With valleys, a difference of 18% was observed between the DQN and A* solutions. All other solutions still saw differences greater than 5%.

In terms of path length, the DQN was not much better than A* (Figure 8). Only with the valley did the DQN significantly outperform – likely because the DQN also had a much higher success rate than A*. These close results, however, are not a great

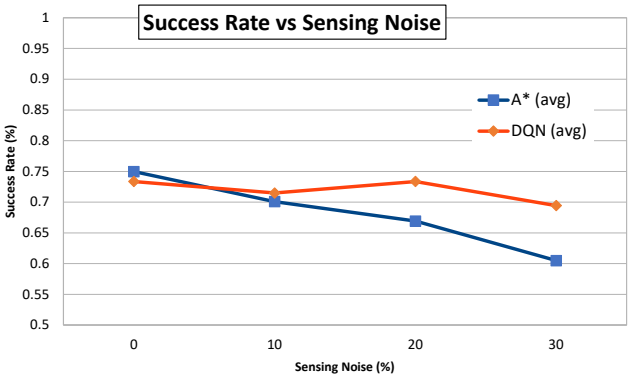

Figure 5. The DQN solution demonstrates significantly improved success rates over A when sensor noise are present.*

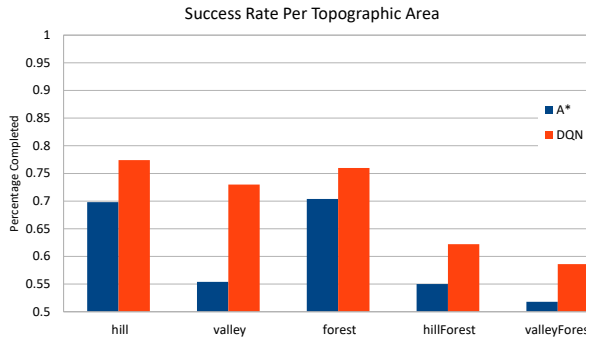

Figure 6. Success rate per topographic area with 30% added sensor noise.

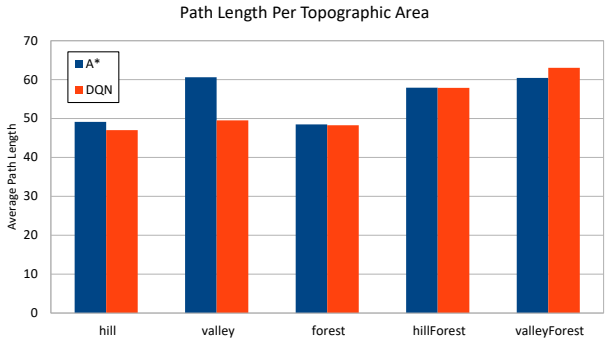

Figure 7. Average path length comparison between the DQN and A solutions with 30% sensor noise.*

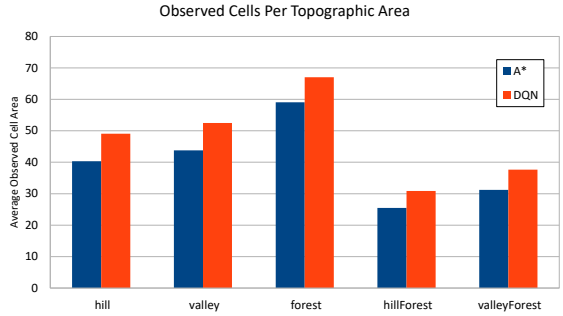

Figure 8. Average count of observed cells per map between the DQN and A solutions with 30% sensor noise.*

surprise as the DQN continues to select a random action 10% of the time as part of its final solution. This means that even with full visibility it cannot achieve optimality.

Even with similar average path lengths, however, the DQN solution is observing more cells on average than A* (Figure 9). In summary, the DQN works well under these conditions because it leverages topography. Without terrain that can be used to improve low visibility, A* and the DQN demonstrate comparable path lengths. Under high visibility conditions (visibility≥3), A* outperforms the DQN even with high added sensor noise. With low general visibility and varied terrain, the DQN takes advantage of the topography to improve performance.

5. FUTURE WORK

The motivation for a DQN-based path planning approach was to investigate strategies for off-road navigation with noisy sensors. At any given time, sensors provide limited information due to sensor noise and occlusion within the scene. RL-based path planning allows a robot to learn strategies for maneuvering in the environment given these sensing limitations. The work presented here provides promising results within a limited simulation environment. A deployment to robotic hardware in conjunction with a strong longer range planner would be an obvious next step. However, there is more work that can also be done prior to such a deployment to improve our understanding of the approach. In particular, we see 3 promising extensions:

1) **Growing the complexity of the simulation scenario** - Introducing a wider range of terrain types, larger scenes, and more realistic challenges in a high-quality simulation environment to validate planning in the presence of sensor noise and occlusion.
2) **Reducing the number of training epochs** – We expect that significant volumes of training will be required on robotic hardware to adapt any simulation results to the real world. However, the current number of simulated iterations are beyond practical limits of real-world hardware-based training missions. For

this purpose, we proposed earlier deploying imitation learning methods as a form of supervisory feedback. Our initial A* bootstrapping algorithm is a form of such feedback, but we expect further training reductions can be made with higher quality training environments. To account for limited supervisory effectiveness, learning to search methods [13] can also be deployed to improve beyond a reference policy.

3) **Generalizing to arbitrary terrains** – These results demonstrated how the DQN could learn to adapt to each individual terrain, improving over A*, but we did not demonstrate a single model that outperformed A* in all environments. To generalize to arbitrary terrains, we would need to train the model for much longer and likely change the network model itself. An alternative, less training intensive solution, is to build up individual, terrain-dependent solutions, and then intelligently switch between them with either logical reasoning [14] or self-aware [15] methods.

Finally, we're interested in exploring other potential domains where this may be applicable, and how well the underlying path planner transfers between domains. Unmanned Aerial Systems (UAS) and Unmanned Underwater Vehicles (UUV) face similar types of problems when developing path planning solutions. Long-term weather forecasts and limited fidelity map data are often used to develop path plans. This type of information is often sufficient for use cases in wide open terrain. However, unexpected weather conditions, congestion due to other vehicles, and maneuvers in proximity to unmapped terrain can limit path planning effectiveness. As we improve our understanding within the ground-vehicle domain, we will also explore the DQN's ability to learn strategies for mitigating these other dynamic aspects of path planning with real-world environments.

6. REFERENCES

[1] N. Nilsson, Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, 1971.

[2] A. Stentz, "The focussed D* algorithm for real-time re-planning," in *Proceedings of the Joint Conference on Artificial Intelligence*, 1995.

[3] S. a. L. M. Koenig, "D* Lite," in *Proceedings of the American Association of Artificial Intelligence*, 2002.

[4] R. Arkin, Behavior-Based Robotics, MIT Press, 1998.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846-894, 2011.

[6] M. Bojarski, B. Firner, B. Flepp, L. Jackel, U. Muller, K. Zieba and D. D. Testa, "End-to-End Deep Learning for Self-Driving Cars," 17 Aug 2016. [Online]. Available: https://developer.nvidia.com/blog/deep-learning-self-driving-cars/. [Accessed 24 May 2021].

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King and D. Ku, "Human-level Control through Deep Reinforcement Learning," *Nature*, vol. 518, p. 529–533, 2015.

[8] P. Wang, C.-Y. Chan and H. Li, "Automated Driving Maneuvers Under Interactive Environment Based on Deep Reinforcement Learning," in *Transportation Research Board*, 2019.

[9] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley and A. Shah, "Learning to Drive in

a Day," wayve.ai, 2018. [Online]. Available: https://wayve.ai/blog/learning-to-drive-in-a-day-with-reinforcement-learning. [Accessed 23 Jan 2019].

[10] J. Liu, P. Jayakumar, J. L. Stein and T. Ersal, "A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments," *Vehicle System Dynamics,* vol. 56, no. 6, pp. 853-882, 2018.

[11] S. Ross and D. Bagnell, "Efficient reductions for imitation learning.," in *International conference on artificial intelligence and statistics* , 2010.

[12] S. Ross and J. A. Bagnell, "Reinforcement and imitation learning via interactive no-regret learning," in *arXiv preprint arXiv:1406.5979.*, 2014.

[13] K. W. Chang, A. Krishnamurthy, A. Agarwal, H. Daume and J. & Langford, "Learning to search better than your teacher.," in *International Conference on Machine Learning*, 2015.

[14] J. Laird, The Soar Cognitive Architecture, MIT Press, 2012.

[15] E. Martinson, N. Paul and L. Moshkina, "Improving Success and Efficiency of Underwater Autonomous Tasks through Dynamic Re-planning and Episodic Reasoning," in *Proceedings of SPIE*, 2021.