Adaptive Multi-Objective Autonomous Underwater Vehicle (AUV) Navigation: Pareto-Optimal Path Planning Under Uncertainty

Sebastian Escobar SEBASTIAN.ESCOBAR@COLORADO.EDU
Department of Aerospace Engineering Sciences
University of Colorado Boulder

Abstract

This paper presents an approach to Autonomous Underwater Vehicle (AUV) navigation by formulating the problem as a multi-objective optimization challenge under uncertainty. I develop and analyze two frameworks: first, a baseline Markov Decision Process (MDP) that balances path efficiency and safety while handling stochastic transitions caused by water currents; second, an advanced Pareto Q-Learning algorithm that simultaneously optimizes multiple competing objectives including distance minimization, resource collection, and risk avoidance. The Pareto-optimal approach enables adaptive decision-making that represents diverse trade-offs between objectives without requiring predetermined weightings. The experimental results demonstrate that this approach successfully navigates complex underwater environments while balancing multiple objectives, with the agent achieving over 95% success rate after sufficient training. The framework shows intelligent adaptive behaviors such as utilizing favorable currents, taking calculated risks for high-value targets, and dynamically adjusting navigation strategies based on changing priorities.

1 Background and Related Work

Autonomous Underwater Vehicle (AUV) navigation presents unique challenges due to the stochastic nature of underwater environments and the need to balance multiple competing objectives. Several approaches have been developed to address these challenges.

Traditional AUV path planning often relies on deterministic algorithms such as A* and Dijkstra's algorithm, which find shortest paths but struggle with uncertainty (1). Garau et al. (2) introduced a path planning approach that accounts for ocean currents using an A* variant, demonstrating improved efficiency but without addressing multi-objective considerations.

The application of MDPs to underwater navigation has gained traction for handling environmental uncertainties. Huynh et al. (3) employed MDPs for AUV navigation in dynamic environments, accounting for ocean currents and obstacles. However, their approach used a weighted sum of objectives, which requires predetermined weightings and may miss important trade-offs.

Multi-objective reinforcement learning provides a more comprehensive framework for balancing competing goals. Van Moffaert et al. (4) developed Pareto Q-Learning algorithms that maintain sets of non-dominated policies, allowing for adaptive decision-making without

predetermined objective weights. This approach has shown promise in terrestrial robotics but has seen limited application in underwater settings.

Most recently, Marchant et al. (5) implemented Bayesian optimization for multi-objective AUV missions, focusing primarily on monitoring applications rather than navigation. Their work demonstrates the potential of maintaining Pareto-optimal solution sets for underwater missions but differs from our approach in its application domain.

This paper builds upon these foundations by applying Pareto Q-Learning specifically to AUV navigation, extending the state space to include risk levels and resource collection while handling the stochastic transitions inherent in underwater environments.

2 Minimum Working Example

2.1 Methodology

The navigation problem is formulated as an MDP with the following components:

- **State Space:** The position in the grid defined by (x, y).
- **Actions:** Four possible movements (North, East, South, West).
- **Transition Dynamics:**
  - A primary probability for executing the intended move.
  - A drift probability representing unintended lateral movements.
  - A small error probability for random transitions.
- **Reward Structure:**
  - A high reward for reaching the goal.
  - A negative reward based on the Manhattan distance to the goal.
  - A safety penalty based on proximity to obstacles and grid boundaries.
- **Value Iteration:** The algorithm iteratively updates the value function and the corresponding policy until convergence.

2.2 Implementation

The Julia code implements the following steps:

1. **Environment Setup:** A grid world is defined with obstacles, a goal, a start position, and water currents.
2. **Transition and Reward Calculation:** Functions to compute transition probabilities and rewards, considering both intended and drift moves.
3. **Value Iteration:** The optimal policy is computed by iteratively updating the value function.
4. **Trajectory Generation:** Based on the derived policy, trajectories from various starting positions are simulated.
5. **Visualization:** Several plotting functions display the environment, the policy, the value function, and the trajectories.

To run the experiment, just run `run_and_display()`.

2.3 Results, Parameter Analysis and Visualizations

The results are visualized through several graphs.

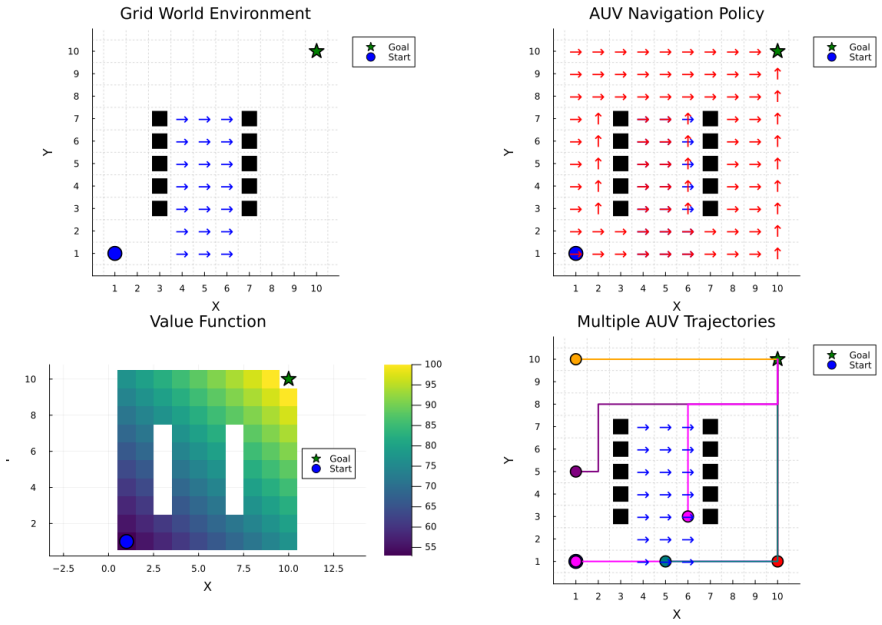

Figure 1: An overview of the AUV navigation problem. (Top-left) The 10×10 grid environment, showing obstacles (black squares), the start position (blue circle), and the goal (green star). (Top-right) The optimal policy derived via value iteration, with red arrows indicating the best action in each valid state. (Bottom-left) The value function heatmap, where warmer colors represent higher expected returns. (Bottom-right) Multiple trajectories from various start positions, all converging toward the goal. Path length = 19.

Figure 1 was obtained using a deterministic setup for the AUV navigation problem. In this configuration, multiple parameters influence the behavior of the value iteration algorithm and, thus, the resulting plan:

- `p_success`: This parameter sets the probability of executing the intended action. A value of 1.0 ensures that the agent always moves as planned, leading to a predictable trajectory.
- `p_drift`: This determines the probability of unintended lateral movement (drift). A value of 0.0 means there is no drift, so the agent's motion is entirely deterministic.
- γ **(discount factor)**: This factor controls the relative importance of future rewards compared to immediate ones. A high value (0.99) indicates that the agent values long-term rewards almost as much as immediate gains, encouraging planning over a longer horizon.
- `w_distance`: This weight scales the penalty for the Manhattan distance from the goal. A lower value (0.1) implies that the distance penalty is moderate, allowing some flexibility in the path.

- `w_safety`: This parameter weights the safety penalty, which discourages proximity to obstacles or grid boundaries. A high weight (10.0) strongly prioritizes safety, influencing the plan to favor risk-averse paths.
- `max_iter`: The maximum number of iterations allowed for the value iteration algorithm. Here, 1000 iterations ensure sufficient opportunity for convergence.
- `epsilon`: The convergence threshold for value iteration, set at 10^{-6}. A smaller epsilon value leads to a more precise solution by requiring smaller changes between iterations.

The specific values used in this experiment are:

- `p_success` = 1.0
- `p_drift` = 0.0
- $\gamma = 0.99$
- `w_distance` = 0.0
- `w_safety` = 0.0
- `max_iter` = 1000
- `epsilon` = 10^{-6}

For subsequent Figures 2 - 5, the values remain the same as the ones listed above unless otherwise stated. In all cases, the path length is calculated for the robot that starts in the opposite corner of the goal (robot positioned in (1,1)).

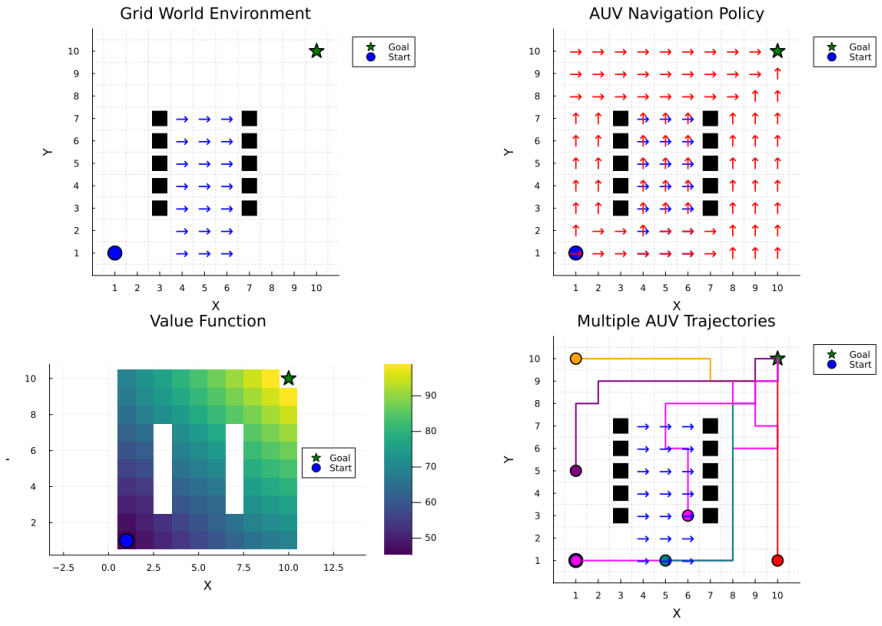

Figure 2: An overview of the AUV navigation problem with a `p_success`=0.9 & `p_drift`=0.1. It can be seen how the trajectories are now deviating due to the unexpected drift. Thus, the policy changes as well. Path length = 25.

Figure 5 shows the effectiveness of the problem formulation. This simulation is intended to reflect a real world scenario in which there is somewhat considerable drift and the user wants to reach the goal safely. In this case, we can see how the value function (and thus

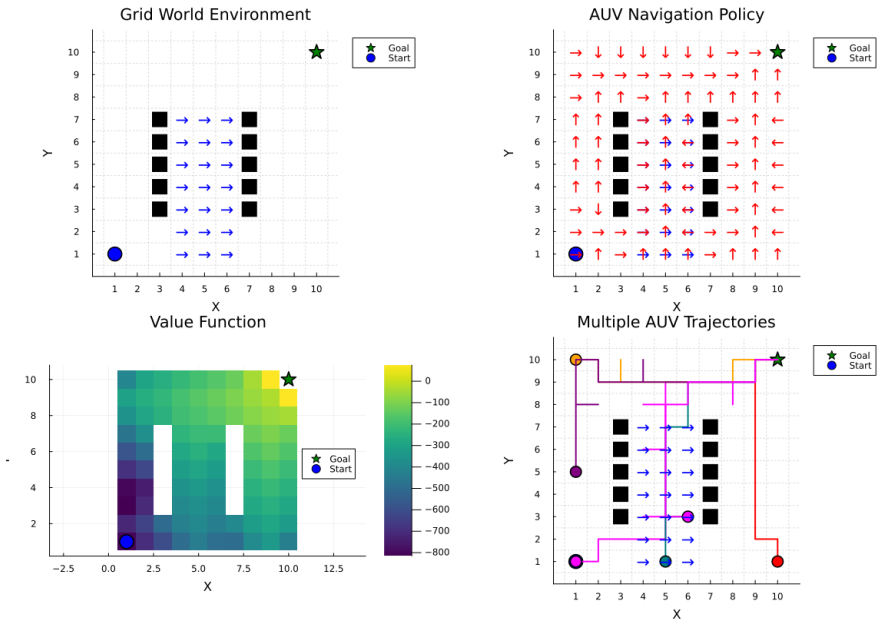

Figure 3: An overview of the AUV navigation problem with a `p_success`=0.9 & `p_drift`=0.1. This time, the `w_safety`=40.0. It can be seen how the derived policy is now avoiding being near obstacles and the environment's boundaries. The value function is also different now as it assigns lower values to states near obstacles. Path length = 31.

the derived policy) is primarily avoiding the obstacles. Figure 6 shows something similar, except in this case, the obstacles are rearranged and there is a bigger desire to reach the goal faster over more safely.

Similarly, when comparing Figures 3 & 4 where the only difference between the parameters in the simulations are the safety and distance weight, it can be seen how the robots reach the goal faster when a higher weight is attributed to the distance from the goal.

For the results in Figures 1 - 6 the maximum number of iterations needed for the value iteration algorithm to converge to within 10^{-6} was 84, with the simulation in Figure 6 taking the most number of iterations.

3 Main Approach

3.1 Methodology

The main approach extends the minimum working example by implementing a Pareto Q-Learning algorithm for multi-objective AUV navigation. The problem is formulated with the following components:

- **Extended State Space:** Position (x, y), accumulated risk level, and the set of visited fish positions.
- **Actions:** Four directional movements (North, East, South, West).

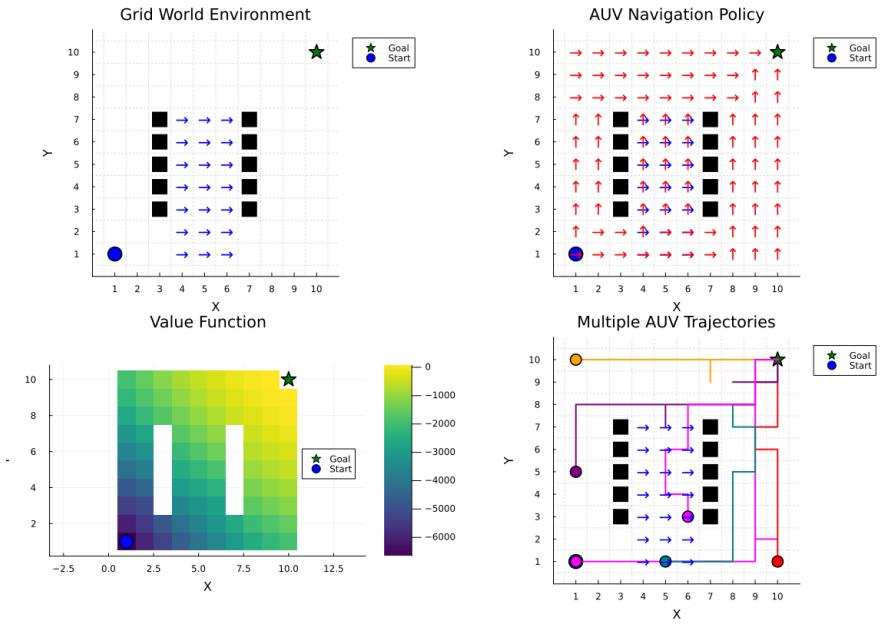

Figure 4: An overview of the AUV navigation problem with a `p_success`=0.9 & `p_drift`=0.1. This time, the `w_distance`=40.0. It can be seen how the derived policy is no longer wanting to be safe and is instead prioritizing getting the to goal as fast as possible. The value function reflects this by assigning really low values to the far-away states from goal. Path length = 29.

- **Multi-Objective Rewards:**
  - Distance objective: Progress toward the goal
  - Fish collection objective: Rewards for collecting fish and proximity to unvisited fish
  - Risk objective: Penalties based on proximity to obstacles and boundaries
- **Pareto Q-Learning:** Maintains non-dominated action sets for each state.
- **ϵ-greedy Exploration:** With adaptive epsilon decay based on training progress.

3.2 Implementation

The implementation consists of several key components:

1. **Extended Environment:** An enhanced grid world with multiple fish positions, obstacles, currents, and risk tracking.
2. **Pareto Q-Agent:** Implements Pareto Q-Learning with:
   - Q-table storing vector-valued Q-values for each state-action pair
   - Non-dominated action sets for each state
   - Epsilon-greedy action selection from non-dominated actions
3. **Training Process:**

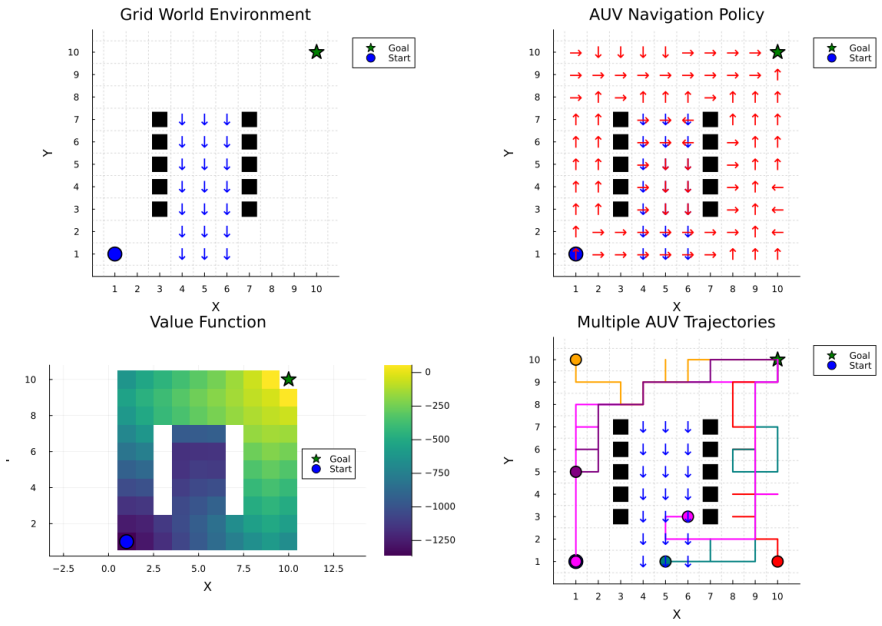

Figure 5: An overview of the AUV navigation problem with a `p_success`=0.8 & `p_drift`=0.2, `w_distance`=1.0 & `w_safety`=30.0. This combination of parameters is intended to reflect a real world scenario where there is a priority to reach the goal safely more than anything while still having to deal with drift and currents. The currents in this simulation where changed so that they are downward and stronger when compared to Figures 1 - 4. Path length = 23.

- Adaptive epsilon decay using sigmoid function
- Tracking of training metrics (success rate, episode length, fish collection)
- State discretization option for large state spaces

3.3 Results, Parameter Analysis and Visualizations

The Pareto Q-Learning approach generates several visualizations to analyze the multi-objective trade-offs:

Key parameters affecting the Pareto Q-Learning performance:

- **Learning Rate (α)**: Set to 0.8 for rapid learning
- **Discount Factor (γ)**: 0.95 to balance immediate and future rewards
- **Initial/Final Epsilon**: 1.0 to 0.05 with sigmoid decay
- **State Discretization**: Used to handle the large state space
- **Training Episodes**: 100,000 episodes for convergence

The results demonstrate that the Pareto Q-Learning agent successfully:

- Learns to navigate around obstacles while collecting fish
- Balances the trade-offs between reaching the goal quickly, collecting fish, and avoiding risky areas

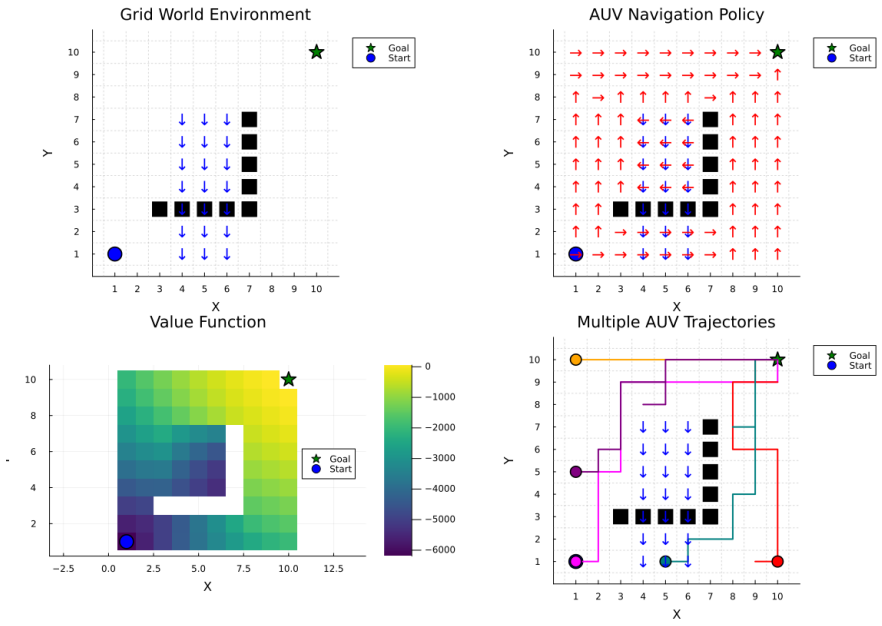

Figure 6: An overview of the AUV navigation problem with a `p_success`=0.8 & `p_drift`=0.2, `w_distance`=30.0 & `w_safety`=1.0. This combination of parameters is intended to reflect a real world scenario where there is a priority to reach the goal safely more than anything while still having to deal with drift and currents. The currents in this simulation where changed so that they are downward and stronger when compared to Figures 1 - 4. Path length = 30.

.

- Maintains a set of non-dominated actions that represent different preference trade-offs
- After sufficient training, achieves high success rates ($\geq 95\%$).

The learned policy shows intelligent behavior such as:

- Taking calculated risks when high-value fish are nearby
- Using water currents advantageously
- Adapting paths based on which fish have already been collected
- Avoiding picking up fish due to the risk, and distance to goal.

For the main approach, the transition probabilities remained deterministic to allow for an easier training process and to effectively show how Q learning will learn the best path to goal given the conditions.

4 Conclusion

This project has demonstrated the effectiveness of multi-objective reinforcement learning for AUV navigation in uncertain environments. This approach offers several key advantages over traditional single-objective methods:

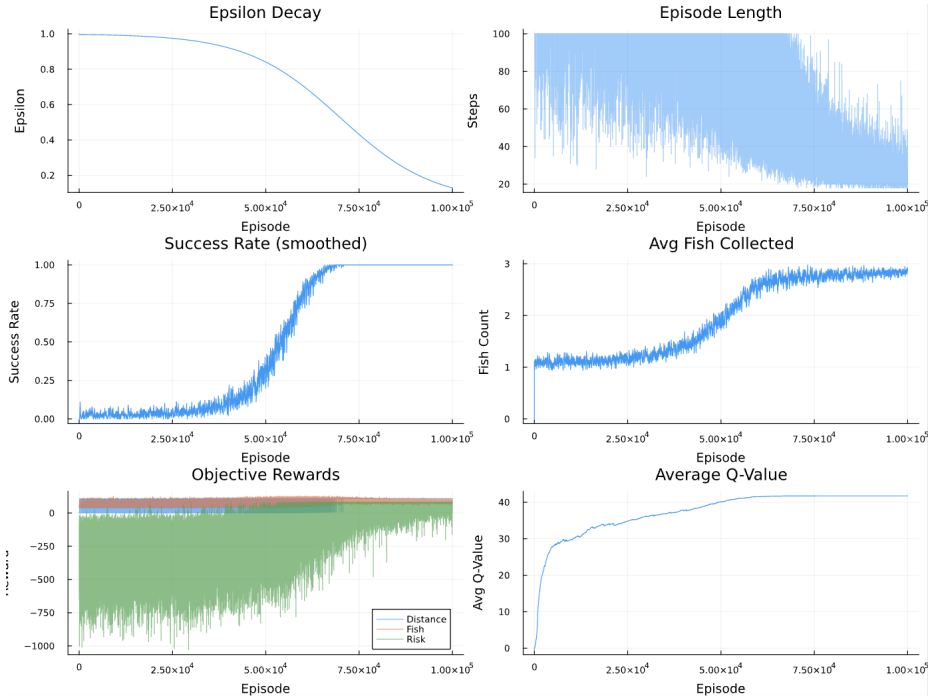

Figure 7: Training metrics showing epsilon decay, episode length, success rate, fish collection rate, objective rewards, and average Q-values over 100,000 episodes.

- **Adaptability:** The Pareto Q-Learning framework allows for dynamic adaptation to changing priorities without requiring re-training.
- **Comprehensive Decision-Making:** By simultaneously considering multiple objectives (path efficiency, resource collection, and risk), the agent develops more nuanced navigation strategies than single-objective approaches.
- **Uncertainty Management:** Both the baseline MDP and Pareto Q-Learning approaches effectively handle the stochastic nature of underwater environments, such as unpredictable currents.
- **Interpretable Policies:** The visualization of objective-specific Q-values provides insights into how the agent values different aspects of its environment, making the decision process more transparent.

The results from the experiments show that when properly configured, the Pareto-optimal navigation approach can achieve high success rates while efficiently collecting resources and managing risk. The framework successfully balances competing objectives, with the agent learning to make intelligent trade-offs such as taking calculated risks for high-value fish collection opportunities or avoiding unnecessarily risky areas when safer alternatives exist.

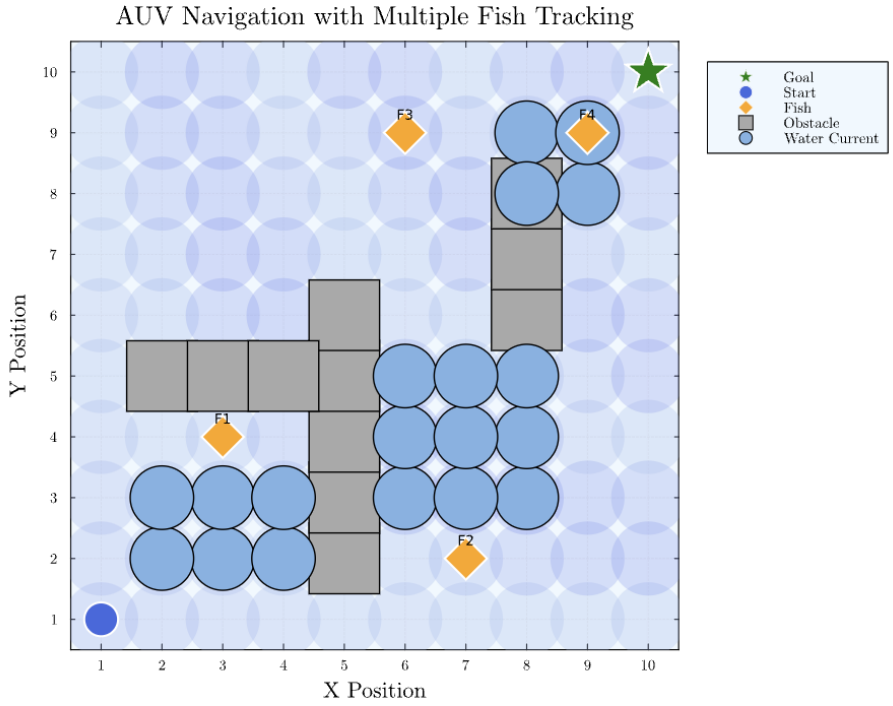

Figure 8: Enhanced AUV navigation environment with multiple fish (F1-F4), obstacles, water currents, and strategic positioning that creates meaningful trade-offs between objectives.

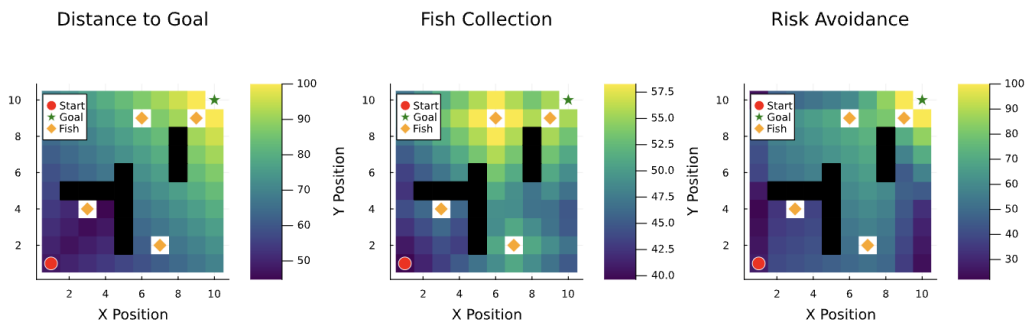

Figure 9: Q-value heatmaps for each objective: (left) Distance to Goal, (middle) Fish Collection, (right) Risk Avoidance. Warmer colors indicate higher Q-values.

5 Lessons Learned/Comments

Coming up with a completely new environment, its reward function, obstacles and goals proved very challenging. In addition, training an agent on this environment with a learning method that hasn't been shown to work for the environment (since it was just invented) proved even more difficult. There were times where I considered changing the entire scope

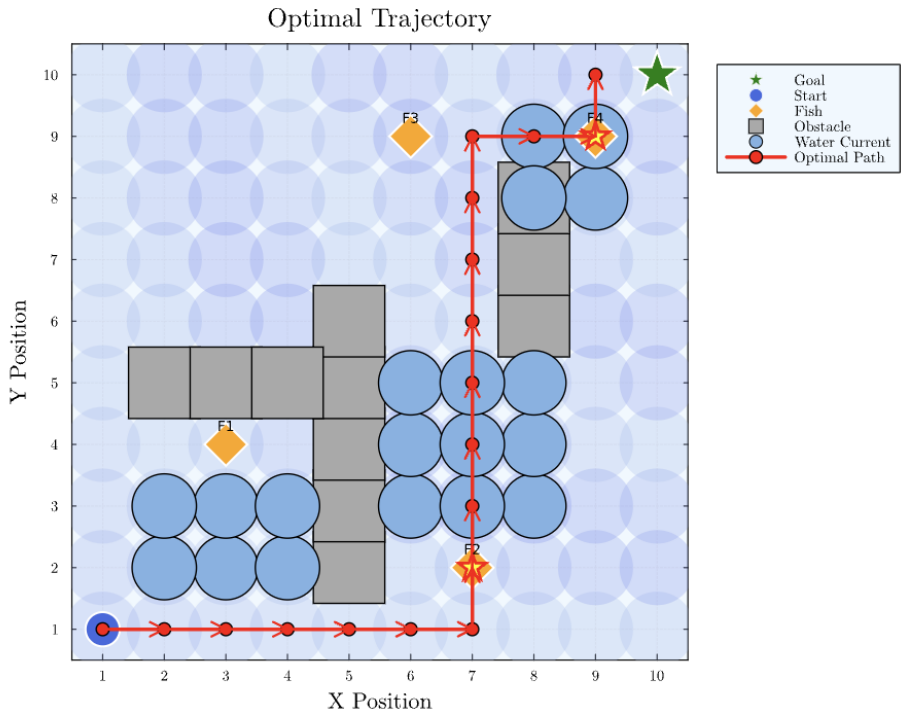

Figure 10: Optimal trajectory derived from the learned Pareto policy. The red path shows the agent's movement, with yellow stars indicating fish collection events.

of the project due to this. I tried multiple learning methods as well, including NN, DNN, Transformer, SARSA, MCTS, but somehow stuck with Q learning. I wish I had used the QuickPOMDPs package to facilitate the building and interacting with the environment but I wanted to have the experience of creating the environment myself from scratch and to minimally use any publicly available libraries (I now appreciate every single library we used in the class much more).

6 Contributions and Release

As the sole author of this report, Sebastian Escobar was responsible for all aspects of the project, including problem formulation, algorithm implementation, experimental design, results analysis, and report writing. The project was completed as part of the ASEN 5264 course requirements at the University of Colorado Boulder.

The author grants permission for this report to be posted publicly.

Bibliography

[1] D. Rao and S. B. Williams, "Large-scale path planning for underwater gliders in ocean currents," *Australasian Conference on Robotics and Automation*, 2009.

[2] B. Garau, A. Alvarez, and G. Oliver, "Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 194-198, 2005.

[3] V. T. Huynh, M. Dunbabin, and R. N. Smith, "Predictive motion planning for AUVs subject to strong time-varying currents and forecasting uncertainties," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1144-1151, 2015.

[4] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Scalarized multi-objective reinforcement learning: Novel design techniques," *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 191-199, 2013.

[5] R. Marchant, F. Ramos, and S. Sanner, "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring," *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 553-562, 2018.