Decentralized Multi-Robot Navigation for Autonomous Surface Vehicles with Distributional Reinforcement Learning

Team: Bhanu Pratap(2411006), Pavan Rajak(2321004)
Date: December 2024

Abstract:
This report outlines the implementation and results of a decentralized multi-robot navigation system for Autonomous Surface Vehicles (ASVs) using Distributional Reinforcement Learning (DRL). The goal of this research is to improve navigation safety and efficiency in environments filled with dynamic and static obstacles, adapting to unpredictable environmental conditions.

---

1. Problem Statement

The increasing demand for Autonomous Surface Vehicles (ASVs) in complex and dynamic environments, such as marine settings, presents significant challenges in terms of collision avoidance, safety, and operational efficiency. A key challenge is the need for multi-robot systems to navigate through both static and dynamic obstacles, often in unknown or unpredictable conditions. Traditional navigation strategies like Artificial Potential Fields (APF) and Reciprocal Velocity Obstacles (RVO) struggle to maintain efficiency and safety when faced with dynamic changes in the environment.

---

2. Objective

The objective of this Project is to first test and benchmark the decentralized navigation solution proposed in the paper, which leverages Distributional Reinforcement Learning (DRL) to handle both static and dynamic obstacles. The primary focus will be on evaluating the performance of the original solution in standard environments. Following this, we will experiment by introducing dynamic obstacles into the environment to assess the robustness and adaptability of the system. Based on these experiments, modifications will be made to the system, and the performance will be reported to analyze the impact of these changes on navigation efficiency, safety, and collision avoidance in more complex and dynamic scenarios.

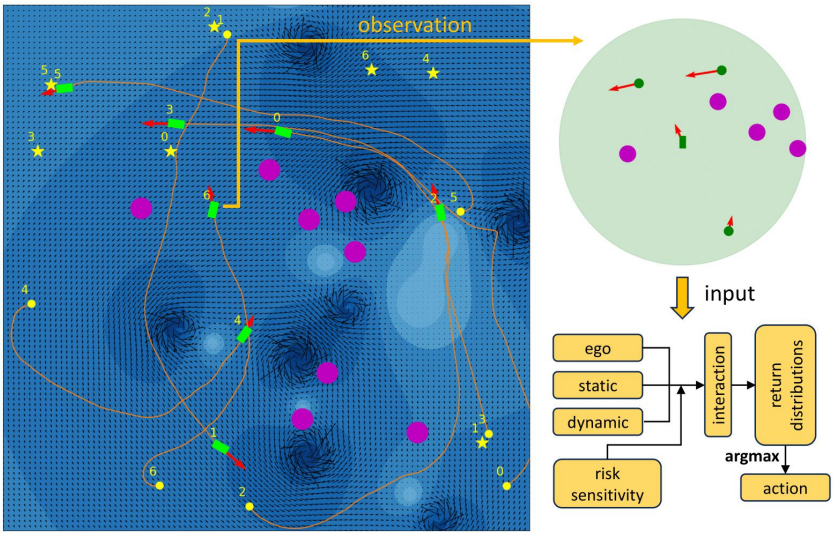
---

3. Paper Overview

The paper titled *"Decentralized Multi-Robot Navigation for Autonomous Surface Vehicles with Distributional Reinforcement Learning"* introduces a novel navigation policy for ASVs using Distributional Reinforcement Learning. The key contributions of the paper are:

Development of a decentralized policy network that ensures coordinated collision avoidance among multiple agents in the environment.
Incorporation of adaptive risk sensitivity to improve decision-making in uncertain environments.
Performance evaluation of the proposed framework in comparison to traditional methods (RVO and APF), demonstrating superior performance in environments with dynamic obstacles.

The approach is implemented using Implicit Quantile Networks (IQN), which enables the system to learn return distributions and adapt to varying levels of risk.



In the above figure all pink circular dots are static obstacles and other agents are considered as moving obstacles for the observation input for a specific agent in IQN

3.1 Theoretical Concepts

The problem is modeled as a Markov Decision Process (MDP) defined by the tuple (S, A, P, R, γ):

State Space (S): Includes the ego robot's position, velocity, detected static obstacles, and other robots. The state is a fixed-size input derived from the closest obstacles and robots.
Action Space (A): Includes possible actions such as steering or speed adjustments.
State Transition Function (P(s′|s,a)): Describes the unknown dynamics influenced by factors like currents or other agents' behaviors.
Reward Function (R(s,a)): Encourages movement toward the goal, penalizes collisions, and rewards goal achievement.
Discount Factor (γ): Balances immediate rewards and future outcomes.

Artificial Potential Fields (APF):
APF is a classical navigation method that generates attractive and repulsive forces based on the robot's position relative to goals and obstacles.

The goal creates an attractive potential field to guide the robot toward its destination.

Obstacles create repulsive potential fields to steer the robot away from potential collisions.
APF is computationally efficient and works well in static environments but often struggles with local minima problems and dynamic scenarios, leading to inefficiencies in complex multi-robot systems.

Reciprocal Velocity Obstacles (RVO):
RVO is a decentralized collision avoidance technique that accounts for the relative velocity and positions of neighboring agents.

RVO predicts future collisions and adjusts the robot's velocity vector to avoid conflicts while maintaining smooth navigation.
It is particularly useful in multi-agent systems where agents must coordinate their movements in dense and dynamic environments.
However, RVO does not account for long-term planning, making it less effective in highly unpredictable or constrained environments.

Deep Q-Networks (DQN):
DQN is a reinforcement learning algorithm that combines Q-learning with deep neural networks to approximate the Q-value function, which represents the expected cumulative reward for taking a specific action in a given state.

Q-Learning Basis:
The algorithm builds on Q-learning, where the agent iteratively updates Q-values using the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

Deep Neural Networks:
Instead of using a table to store Q-values for all state-action pairs, DQN employs a neural network to approximate Q(s,a) making it feasible to handle large or continuous state spaces.

IQN: Implicit Quantile Networks
IQN differs from traditional DQN by learning the full distribution of returns instead of expected returns, capturing variability and risk. It utilizes quantile functions and Conditional Value at Risk (CVaR) to allow risk-sensitive decision-making. IQN's training loss optimizes a distributional loss, enabling agents to learn a richer representation of future rewards.

5. Code Results: Benchmarks

The framework was benchmarked against traditional methods like DQN, RVO, and APF. The key metrics for comparison include:

Reward Learning: IQN outperforms DQN in terms of cumulative rewards, especially in environments with dynamic obstacles.
Energy, Time, and Success Rate: IQN demonstrates better energy efficiency and higher success rates compared to DQN, RVO, and APF.
Scenario Visualization: Visual plots of agents with IQN policy show improved coordination and collision avoidance with obstacles.

Table : 1 Training and Evaluation Configuration

Parameter	Value
Seed	9
Total Timesteps	6,000,000
Evaluation Frequency	60,000
Save Directory	/home/rfal/Multi_Robot_Distributional_RL_Navigation/training_data
Use IQN	True

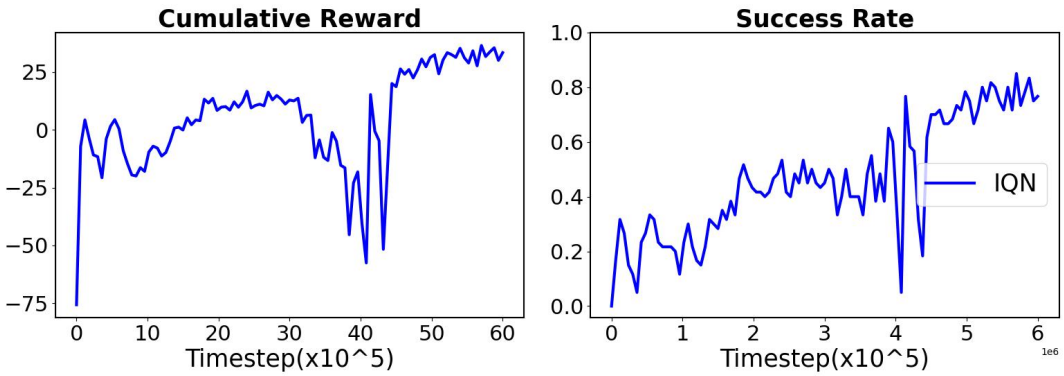
Table : 2 Training Schedule

Timestep	Cooperative Agents	Non-Cooperative Agents	Cores	Obstacles	Min Start-to-Goal Distance
0	3	0	3	0	30.0
1,000,000	5	0	4	1	35.0
2,000,000	7	0	5	2	40.0
3,000,000	7	0	8	3	40.0
4,000,000	7	0	8	4	40.0
5,000,000	7	0	8	5	40.0

Table : 3 Testing Schedule

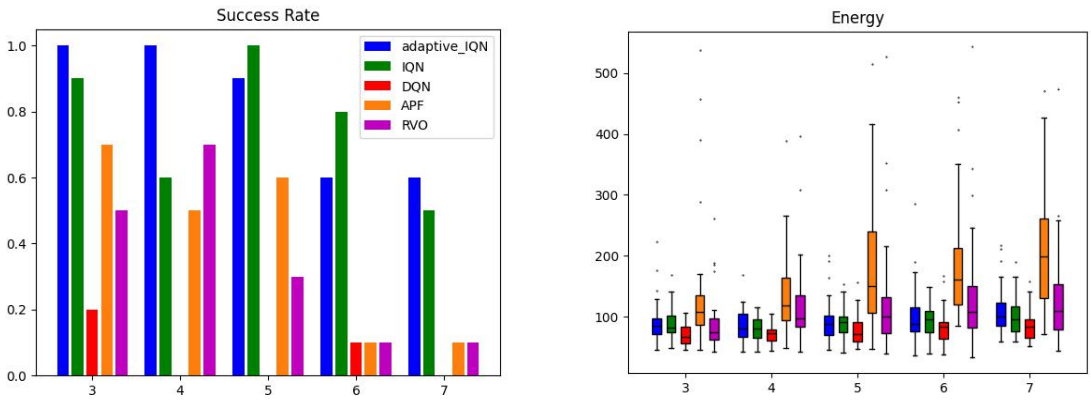
Episodes	Cooperative Agents	Non-Cooperative Agents	Cores	Obstacles	Min Start-to-Goal Distance
10	3	0	3	0	40.0
10	5	0	4	1	40.0
10	7	0	5	2	40.0
10	3	0	4	3	40.0
10	5	0	6	4	40.0
10	7	0	8	5	40.0

Outputs :



The above figure shows the cumulative reward after the 60,00,000 iterations for the IQN model while training and its success rate during the testing

Evaluation and comparison Results :



The above two figures depicting how the IQN models outperforms the other models like RVO,APF and DQN in terms of succes rate and energy where the y - axis is showing number of co-operative agents per exiperiment while the evaluations

---

6. Experimentations

6.1 Environment Changes:

The environment was modified to include dynamic obstacles, representing moving entities that interact with the robots. This change introduced additional complexity to the navigation task, requiring updates to the reward structure to incentivize the avoidance of these dynamic obstacles.

Added methods and code in Marin_env file :

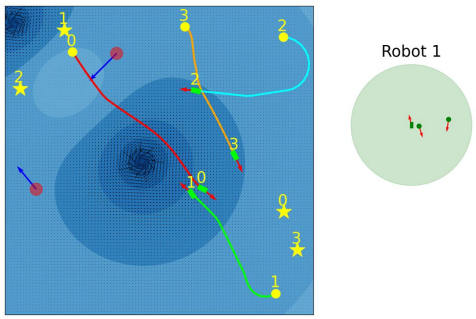
dynamic_env: bool ; *added a boolean to check whether dynamic env is passed in config file*

self.dynamic_obstacles = [] ; *added new list to handel Dynamic obstacles*

More updates *:*

1) added generate_dynamic_obstacles method
2) added update_dynamic_obstacles method
3) added the init for above two methods
4) added update code line in reset method
5) added velocity of dynamic obstacles saving logic in episode saving method
6) *all other simple changes addressed in Code Readme file*

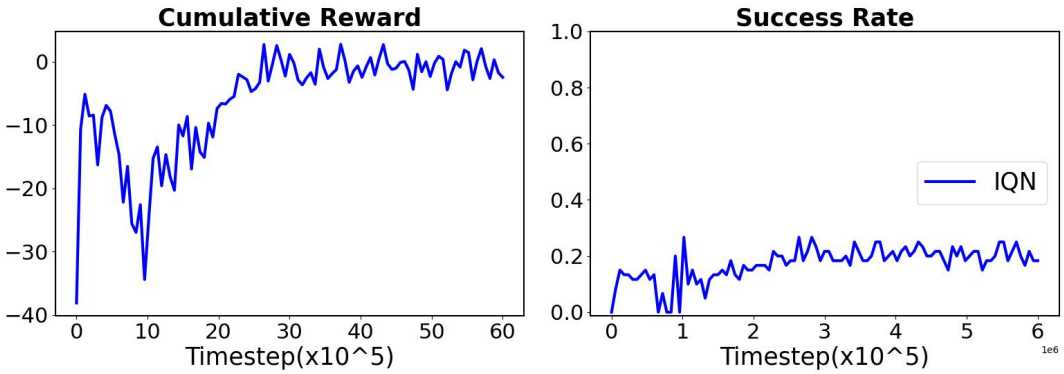
Changed Env :



The above figure shows the seen of updated dynamic obstacle based Marin env for training which are red in colour

6.2 Results:
The results show that the IQN model successfully adapted to the inclusion of dynamic obstacles, although performance struggled when the number of obstacles or agents increased. A comparison of the updated reward structure with the previous one indicated that the system's performance improved when agents avoided moving obstacles.

Parameter	Value
Seed	9
Total Timesteps	6,000,000
Evaluation Frequency	60,000
Save Directory	/home/rfal/Multi_Robot_Distributional_RL_Navigation/training_data
Use IQN	True
dynamic_env	True

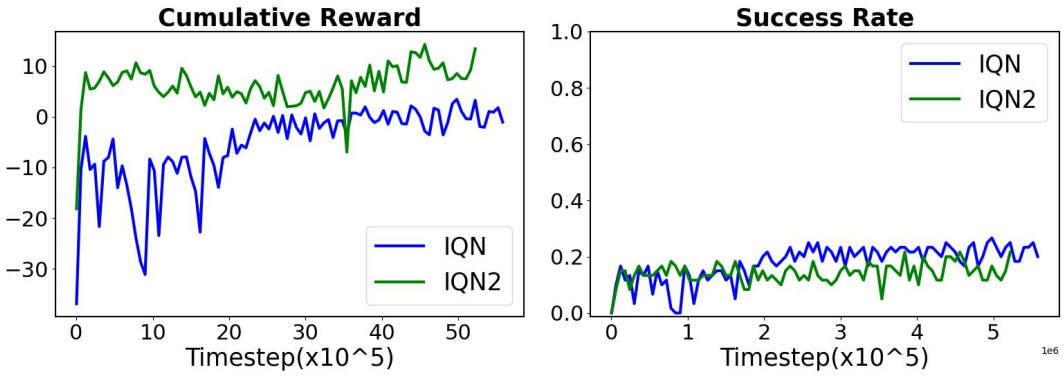


The above figure shows the cumulative reward after the 60,00,000 iterations for the IQN model while training and its success rate during the testing which dipped due to the addition of dynamic obstacles

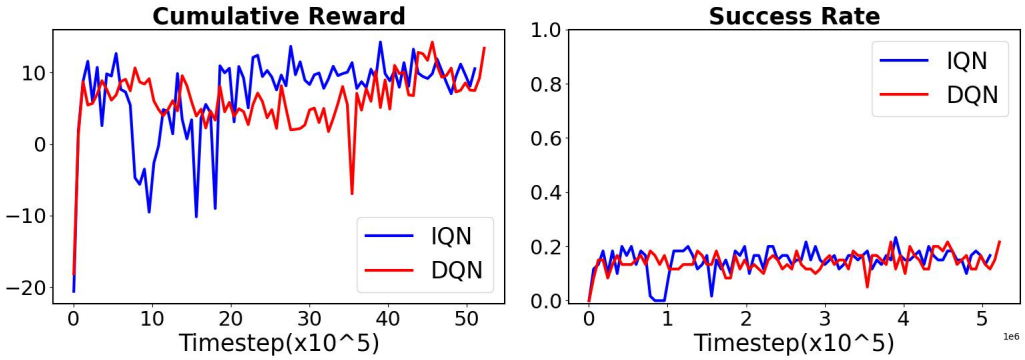
6.3 Updates and Experiments:

Several updates were implemented, including:

A new reward structure that rewards agents for avoiding dynamic obstacles.
Adaptive IQN with CVaR calculations to handle the risks associated with dynamic environments.
We added extra reward +2 for successfuly avoiding the dynamic obsatcles for 25 steps during training.
Performance evaluation under different configurations of agents, obstacles, and environmental conditions.

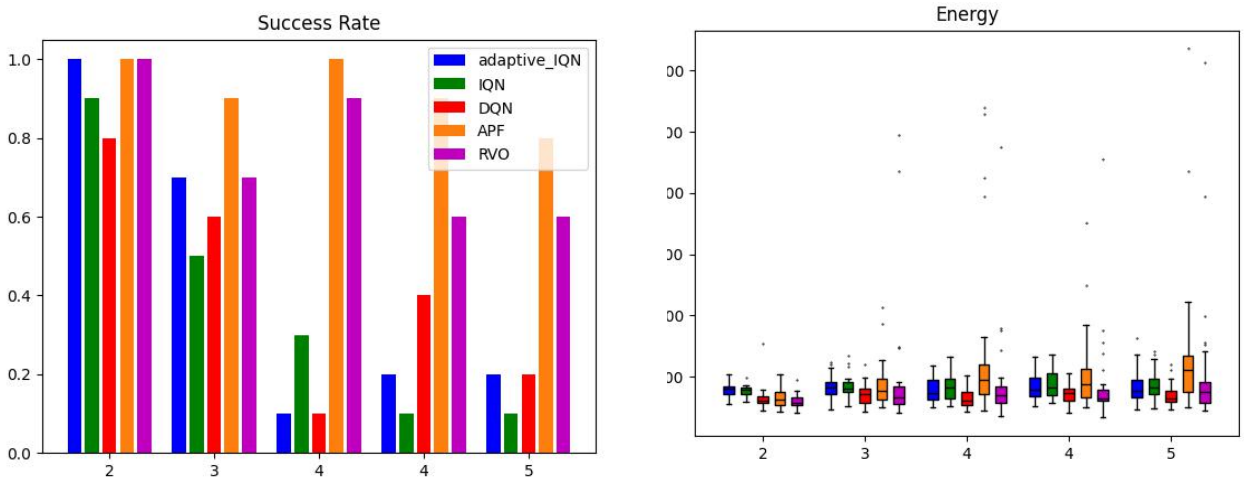


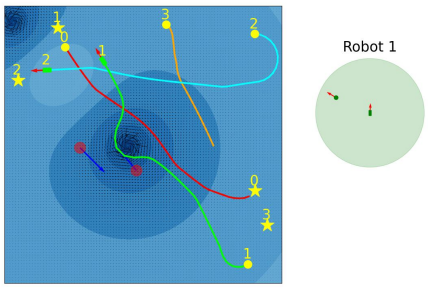
We can see that from above Figures the updated Reward structure helped the model to increase the cummulative reward while training but the succes rate does not improved much on testing.



Plot comparison of IQN and DQN models trained on same updated reward structure with dynamic obstacles.(there is not much differnce in Both)

Evaluation Results :





The above two figures depicting how the IQN models is not performing well as compared with other models like RVO,APF and DQN in terms of succes rate and energy where the y - axis is showing number of co-operative agents per exiperiment while the evaluations

7. Final Conclusion

The experiments confirmed that while IQN provided a more risk-sensitive and robust policy for multi-robot navigation, challenges remained when dealing with a high number of dynamic obstacles and agents. Traditional methods like RVO and APF performed better under these conditions. However, the IQN approach showed promise in environments with static obstacles and smaller-scale dynamic obstacle scenarios.

---

8. Future Scope

To address the limitations observed in dynamic environments, future work will focus on:

- Integrating attention-based mechanisms to better handle the complexity of dynamic obstacles.
- Exploring decentralized policies based on Proximal Policy Optimization (PPO) to further improve scalability and adaptability.
- Enhancing the framework to handle larger and more complex environments with multiple agents and obstacles.