ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/372075623

# Data Driven Control for marine vehicle maneuvering

**Thesis** · July 2023

DOI: 10.13140/RG.2.2.16194.15045

CITATION
1

READS
244

1 author:

Md Shadab Alam
Eindhoven University of Technology

**9** PUBLICATIONS **31** CITATIONS

SEE PROFILE

All content following this page was uploaded by Md Shadab Alam on 04 July 2023.

The user has requested enhancement of the downloaded file.

DEPARTMENT OF OCEAN ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI – 600036

# Data Driven Control for Marine Vehicle Maneuvering

*A Thesis*

*Submitted by*

**MD SHADAB ALAM**

*For the award of the degree*

*Of*

**MASTER OF SCIENCE**

May 2023

DEPARTMENT OF OCEAN ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI – 600036

# Data Driven Control for Marine Vehicle Maneuvering

*A Thesis*

*Submitted by*

**MD SHADAB ALAM**

*For the award of the degree*

*Of*

**MASTER OF SCIENCE**

May 2023

© 2023 Indian Institute of Technology Madras

*An investment in knowledge pays the best interest.*

**– Benjamin Franklin**

*To my beloved family, who has always been my source of inspiration and support. Thank you for believing in me and encouraging me to never give up on my dreams. This thesis is dedicated to you with love and gratitude.*

*And to my mentor, whose guidance and wisdom has been invaluable in shaping my academic journey. Thank you for your unwavering support and for pushing me to aim higher. This achievement would not have been possible without you.*

*Finally, to all those who have directly or indirectly contributed to my growth and learning, thank you for being a part of this journey. This dedication is a testament to your impact and influence on my life.*

# THESIS CERTIFICATE

This is to undertake that the Thesis titled **DATA DRIVEN CONTROL FOR MARINE VEHICLE MANEUVERING**, submitted by me to the Indian Institute of Technology Madras, for the award of **Master of Science**, is a bonafide record of the research work done by me under the supervision of **Dr. Abhilash Sharma Somayajula**. The contents of this Thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

**Chennai 600036** **Md Shadab Alam**

**Date:**

**Dr. Abhilash Sharma Somayajula**
Research advisor
Assistant Professor
Department of Ocean Engineering
IIT Madras

© 2023 Indian Institute of Technology Madras

# LIST OF PUBLICATIONS

## I. REFEREED JOURNALS BASED ON THESIS

Deraj, Rohit, RS Sanjeev Kumar, **Md Shadab Alam**, and Abhilash Somayajula. "Deep reinforcement learning based controller for ship navigation." Ocean Engineering 273 (2023): 113937.

## II. PUBLICATIONS IN CONFERENCE PROCEEDINGS

**Alam, M.S.**, Ramkumar Sudha, S.K., Somayajula, A. AI on the water: Applying DRL to autonomous vessel navigation. In Proceedings of the Sixth International Conference in Ocean Engineering (ICOE2023). Springer.

Jose, J., **Alam, M.S.**, Somayajula, A.S. Navigating the Ocean with DRL: Path following for marine vessels. In Proceedings of the Sixth International Conference in Ocean Engineering (ICOE2023). Springer.

Ramkumar Sudha, S.K., **Alam, M.S.**, Reddy, B., Somayajula, A.S. Comparison of path following in ships using modern and traditional controllers. In Proceedings of the Sixth International Conference in Ocean Engineering (ICOE2023). Springer.

# ACKNOWLEDGEMENTS

Words cannot express my gratitude to my supervisor, **Dr. Abhilash Sharma Somayajula**, without whom this would have been impossible. I am deeply indebted to him for his constant guidance and motivation which helped me grow professionally and personally. His mentorship is not limited to research, and I am also thankful to him for his encouragement and advise regarding graduate program applications.

I would also like to acknowledge **Dr. Nilanjan Saha**, **Dr. Suresh Rajendran** and **Dr. Nirav Patel** members of my MS degree project review committee, for their input and suggestions that helped my work.

I am grateful to my friend **R.S Sanjeev Kumar** and **Rohit Deraj** with whom I had countless discussions in the initial phase of this project.

Last but not the least, I would like to express gratitude to my family for their unconditional support throughout my entire study at IIT Madras.

# ABSTRACT

**KEYWORDS** Reinforcement Learning; Deep Learning; Autonomous Vehicle; Path-following; Obstacles Avoidance; Machine Learning Controller

The majority of global marine accidents are caused by human decision-making errors, which has resulted in increased interest in automation within the marine industry. However, obstacle avoidance for autonomous surface vehicles in unknown environments is particularly difficult. This study investigates the possibility of utilizing a deep reinforcement learning (DRL) approach to control an underactuated autonomous surface vehicle following a predetermined path while avoiding collisions with static and dynamic obstacles. The ship's movement is modelled using a three-degree-of-freedom (3-DOF) dynamic model, with the KRISO container ship (KCS) being selected for the study due to its extensive use in previous research and readily available hydrodynamic coefficients for numerical modelling. The study evaluates the performance of various DRL algorithms, such as Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), and Proximal Policy Optimization (PPO) algorithms, for path following and their effectiveness in the presence of wind, as well as comparing them to the traditional PD controller. The study also explores DQN and DDPG algorithms for both static and dynamic obstacle avoidance and proposes a hybrid network that uses two networks for improved path following and obstacle avoidance capabilities.

# CONTENTS

Page

**ACKNOWLEDGEMENTS** **i**

**ABSTRACT** **iii**

**LIST OF TABLES** **ix**

**LIST OF FIGURES** **xi**

**GLOSSARY** **xv**

**ABBREVIATIONS** **xix**

**NOTATION** **xxi**

**CHAPTER 1 INTRODUCTION** **1**

1.1 Background and motivation . . . . . . . . . . . . . . . . . . . . 1
1.2 Literature review . . . . . . . . . . . . . . . . . . . . 2
1.3 Objective . . . . . . . . . . . . . . . . . . . . 7
1.4 Scope . . . . . . . . . . . . . . . . . . . . 8

**CHAPTER 2 SHIP DYNAMICS** **11**

2.1 Forces due to hull . . . . . . . . . . . . . . . . . . . . 15
2.2 Forces due to propeller . . . . . . . . . . . . . . . . . . . . 15
2.3 Forces due to rudder . . . . . . . . . . . . . . . . . . . . 17
2.4 Rudder angle variation . . . . . . . . . . . . . . . . . . . . 18

**CHAPTER 3 REINFORCEMENT LEARNING ALGORITHMS** **21**

3.1 Introduction . . . . . . . . . . . . . . . . . . . . 21
3.2 Deep Q-Learning . . . . . . . . . . . . . . . . . . . . 22
3.3 Deep Deterministic Policy Gradient . . . . . . . . . . . . . . . . . . . . 24
3.4 Proximal Policy Optimization . . . . . . . . . . . . . . . . . . . . 27

**CHAPTER 4 RL-FRAMEWORK** **29**

4.1 Waypoint Tracking . . . . . . . . . . . . . . . . . . . . 29
4.1.1 Observation States . . . . . . . . . . . . . . . . . . . . 30
4.1.2 Action Space . . . . . . . . . . . . . . . . . . . . 32
4.1.3 Reward Structure . . . . . . . . . . . . . . . . . . . . 32
4.1.4 Training Process . . . . . . . . . . . . . . . . . . . . 35
4.1.5 Episode termination . . . . . . . . . . . . . . . . . . . . 35

4.2 Static Obstacle Avoidance . . . . . . . . . . . . . . . . . . . . . . 37
4.2.1 Observation States . . . . . . . . . . . . . . . . . . . . . . . . 37
4.2.2 Action Space . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
4.2.3 Reward Structure . . . . . . . . . . . . . . . . . . . . . . . . . 38
4.2.4 Training Process . . . . . . . . . . . . . . . . . . . . . . . . . . 38
4.2.5 Episode Termination . . . . . . . . . . . . . . . . . . . . . . . 40
4.3 Dynamic Obstacle Avoidance . . . . . . . . . . . . . . . . . . . . . 40
4.3.1 Measure of Collision Risk . . . . . . . . . . . . . . . . . . . . . 41
4.3.2 Observation Space . . . . . . . . . . . . . . . . . . . . . . . . . 43
4.3.3 Action Space . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.3.4 Reward Structure . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.3.5 Training Process . . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.3.6 Episode Termination . . . . . . . . . . . . . . . . . . . . . . . 45

**CHAPTER 5 RESULTS AND DISCUSSIONS 47**

5.1 Comparison of modern and traditional controllers . . . . . . . . . . . . 47
5.1.1 Hyperparameters of the network . . . . . . . . . . . . . . . . . . 47
5.1.2 Calm water results . . . . . . . . . . . . . . . . . . . . . . . . . 49
5.1.3 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
5.2 Performance with wind forces . . . . . . . . . . . . . . . . . . . . . 56
5.2.1 Modelling wind forces . . . . . . . . . . . . . . . . . . . . . . . 57
5.2.2 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
5.2.3 Comparison with a PD-based controller . . . . . . . . . . . . . . . 60
5.3 Comparison of DRL-based controllers . . . . . . . . . . . . . . . . . 63
5.3.1 Waypoint Tacking . . . . . . . . . . . . . . . . . . . . . . . . . 64
5.3.2 Hyperparameters of the network . . . . . . . . . . . . . . . . . . 64
5.3.3 Path following through waypoint tracking . . . . . . . . . . . . . 64
5.3.4 Performance with wind forces . . . . . . . . . . . . . . . . . . . 71
5.3.5 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
5.4 Static Obstacle Avoidance . . . . . . . . . . . . . . . . . . . . . . 83
5.4.1 Hyperparameters of the network . . . . . . . . . . . . . . . . . . 83
5.4.2 Single static obstacle avoidance . . . . . . . . . . . . . . . . . . 88
5.4.3 Multiple Static Obstacle Avoidance . . . . . . . . . . . . . . . . . 89
5.5 Dynamic Obstacle Avoidance . . . . . . . . . . . . . . . . . . . . . 97
5.5.1 Hyperparameters of the network . . . . . . . . . . . . . . . . . . 97
5.5.2 Obstacle Avoidance results . . . . . . . . . . . . . . . . . . . . . 97

**CHAPTER 6 DISCUSSION 103**

6.1 Waypoint Tracking . . . . . . . . . . . . . . . . . . . . . . . . . . 103
6.1.1 Yaw rate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104
6.1.2 Reward Function . . . . . . . . . . . . . . . . . . . . . . . . . . 104
6.1.3 Effect of reward $r_3$ . . . . . . . . . . . . . . . . . . . . . . . 107
6.2 Static Obstacle Avoidance . . . . . . . . . . . . . . . . . . . . . . 108
6.2.1 Hybrid Path Following and Obstacle Avoidance Architecture . . . . . . 109

**CHAPTER 7 CONCLUSION AND FUTURE STUDIES 113**

APPENDIX A MAKING THE EPSILON GOES TO A CONSTANT VALUE WHICH IS GREATER THAN 0 117

APPENDIX B TUNING OF THE PD CONTROLLER 121

APPENDIX C HYBRID ARCHITECTURE OF DRL AND PD BASED CONTROLLER 123

BIBLIOGRAPHY 125

CURRICULUM VITAE 129

GENERAL TEST COMMITTEE 131

# LIST OF TABLES

**Table** **Caption** **Page**

2.1 KCS ship parameters . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.2 Prime-II system of normalization . . . . . . . . . . . . . . . . . . . . . 14
2.3 Ship parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
2.4 Ship hydrodynamic coefficients . . . . . . . . . . . . . . . . . . . . . 16
2.5 Propeller parameters . . . . . . . . . . . . . . . . . . . . . . . . . . 17
2.6 Rudder force parameters . . . . . . . . . . . . . . . . . . . . . . . . 19

5.1 Hyperparameters for DQN model . . . . . . . . . . . . . . . . . . . . 49
5.2 Sensors Onboard . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
5.3 Waypoints . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
5.4 Wind force parameters . . . . . . . . . . . . . . . . . . . . . . . . . 58
5.5 Wind force cases . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
5.6 Comparison of the DQN and Traditional controller . . . . . . . . . . . 63
5.7 Hyperparameters for DQN model . . . . . . . . . . . . . . . . . . . . 65
5.8 Hyperparameters for PPO model . . . . . . . . . . . . . . . . . . . . 66
5.9 Hyperparameters for DDPG model . . . . . . . . . . . . . . . . . . . . 67
5.10 Comparison of the three DRL-based controllers in terms of RMSE and CE 85
5.11 Hyperparameters for DQN model . . . . . . . . . . . . . . . . . . . . 85
5.12 Hyperparameters for DDPG model . . . . . . . . . . . . . . . . . . . . 88
5.13 Comparison of agent when obstacle of size 0.25L is in between line joining the waypoints . . . . . . . . . . . . . . . . . . . . . . . . . . 90
5.14 Comparison of agent when obstacle of size 0.5L is in between line joining the waypoints . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
5.15 Comparison of agent when obstacle of size 0.25L is in path of the ship . 91
5.16 Comparison of agent when obstacle of size 0.5L is in is in path of the ship 93
5.17 Hyperparameters for DQN model . . . . . . . . . . . . . . . . . . . . 99
5.18 Hyperparameters for DDPG model . . . . . . . . . . . . . . . . . . . . 99
5.19 Comparison of agent in case of dynamic obstacle . . . . . . . . . . . . 102

A.1 Comparison of the DQN controller . . . . . . . . . . . . . . . . . . . 117

# LIST OF FIGURES

| Figure | Caption | Page |
|---|---|---|
| 2.1 | Representation of ship kinematic variables | 13 |
| 3.1 | DQN algorithm flowchart | 23 |
| 4.1 | Depiction of the waypoint tracking problem | 31 |
| 4.2 | Cross track reward | 33 |
| 4.3 | Course angle reward | 33 |
| 4.4 | Scatter plot of randomly generated destination waypoints for training | 36 |
| 4.5 | Visualizing the termination condition | 37 |
| 4.6 | Depiction of the static obstacle avoidance problem | 39 |
| 4.7 | Depiction of training for static obstacle avoidance | 39 |
| 4.8 | Calculation of DCPA and TCPA | 42 |
| 4.9 | Depiction of the dynamic obstacle avoidance problem | 43 |
| 4.10 | Depiction of training for dynamic obstacle avoidance | 44 |
| 5.1 | Training returns and loss for the DQN model | 48 |
| 5.2 | Single waypoint tracking for DQN | 50 |
| 5.3 | Circle maneuver waypoint tracking | 51 |
| 5.4 | Ellipse maneuver waypoint tracking ($d_{RMSE} = 0.2969L$) | 52 |
| 5.5 | Eight (‘8’) maneuver waypoint tracking ($d_{RSME} = 0.4557L$) | 52 |
| 5.6 | 1:75.5 scaled model of KCS | 53 |
| 5.7 | Sensors used: simpleRTK2B SBC(left) and SBG Ellipse-A (right) | 54 |
| 5.8 | Waypoints and the Trajectory: white represents the square generated by the waypoints, red represents the trajectory traversed by the vessel. | 54 |
| 5.9 | Comparison of the path traversed by the model in experiment and simulation (Simulation $d_{RMSE} = 0.6286L$ Experimental $d_{RMSE} = 0.8257L$) | 55 |
| 5.10 | Comparison of cross-track error between experiment and simulation. | 55 |
| 5.11 | Comparison of course-angle error between experiment and simulation. | 56 |
| 5.12 | Different cases for path following in presence of constant and uniform wind for DQN(3-actions) controller | 59 |
| 5.13 | ILOS Guidance | 61 |
| 5.14 | Comparison of the path traversed by the model in PD controller and RL controller in simulation | 62 |
| 5.15 | Single waypoint tracking for DQN | 65 |
| 5.16 | DDPG Single waypoint tracking | 66 |
| 5.17 | PPO Single waypoint tracking | 67 |
| 5.18 | Training returns and loss for the DQN model | 68 |
| 5.19 | Training returns and loss for the PPO model | 69 |
| 5.20 | Training returns and loss for the DDPG model | 70 |
| 5.21 | Circle (radius = 6L) maneuver waypoint tracking | 72 |
| 5.22 | Circle (radius = 12L) maneuver waypoint tracking | 73 |

5.23 Ellipse maneuver waypoint tracking . . . . . . . . . . . . . . . . . . . 74
5.24 Eight maneuver waypoint tracking . . . . . . . . . . . . . . . . . . . . 75
5.25 Star maneuver waypoint tracking . . . . . . . . . . . . . . . . . . . . . 76
5.26 Straight Line Path with wind $V_w = 6L$ and $\beta_w = 90°$ . . . . . . . . . . . 77
5.27 Straight Line Path with wind $V_w = 6L$ and $\beta_w = 270°$ . . . . . . . . . . 78
5.28 Straight Line Path with wind $V_w = 9L$ and $\beta_w = 90°$ . . . . . . . . . . . 79
5.29 Straight Line Path with wind $V_w = 9L$ and $\beta_w = 270°$ . . . . . . . . . . 80
5.30 Ellipse Path with wind $V_w = 6L$ and $\beta_w = 0°$ . . . . . . . . . . . . . . . 81
5.31 Eight Path with wind $V_w = 6L$ and $\beta_w = 45°$ . . . . . . . . . . . . . . . 82
5.32 Comparison of Loss and Returns of different DRL-based controllers . . 84
5.33 Training returns and loss for the DQN model . . . . . . . . . . . . . . 86
5.34 Training returns and loss for the DDPG model . . . . . . . . . . . . . . 87
5.35 DQN agent successfully avoids obstacle of size 0.25L when present on line joining the waypoints . . . . . . . . . . . . . . . . . . . . . . . . 89
5.36 DQN agent successfully avoids obstacle of size 0.5L when present on line joining the waypoints . . . . . . . . . . . . . . . . . . . . . . . . 90
5.37 DQN agent successfully avoids obstacle of size 0.25L when present in path of ship . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
5.38 DQN agent successfully avoids obstacle of size 0.5L when present in path of ship . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
5.39 DQN agent successfully avoids obstacle on a straight line path . . . . . 92
5.40 DDPG agent successfully avoids obstacle of size 0.25L when present on line joining the waypoints . . . . . . . . . . . . . . . . . . . . . . . . 93
5.41 DDPG agent successfully avoids obstacle of size 0.5L when present on line joining the waypoints . . . . . . . . . . . . . . . . . . . . . . . . 94
5.42 DDPG agent successfully avoids obstacle of size 0.25L when present in path of ship . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
5.43 DDPG agent successfully avoids obstacle of size 0.5L when present in path of ship . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
5.44 DDPG agent successfully avoids obstacle on a straight line path . . . . . 96
5.45 Multiple Static Obstacle Avoidance . . . . . . . . . . . . . . . . . . . 98
5.46 Training returns for DQN dynamic obstacle model . . . . . . . . . . . . 100
5.47 Training returns for DDPG dynamic obstacle model . . . . . . . . . . . 101

6.1 Relative cross track error for straight line maneuver with wind speeds $V_w = 6, 9$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 105
6.2 Without yaw rate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 106
6.3 Modified Rewards . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107
6.4 Constant Reward $r_3$ . . . . . . . . . . . . . . . . . . . . . . . . . . . 108
6.5 Agent unable to reach destination . . . . . . . . . . . . . . . . . . . . 110
6.6 Hybrid architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111

A.1 Training loss and returns for DQN model . . . . . . . . . . . . . . . . 118
A.2 Path Following of the DQN model . . . . . . . . . . . . . . . . . . . . 119

B.1 Comparison of the path traversed by the model in PD controller and RL controller in simulation . . . . . . . . . . . . . . . . . . . . . . . . 122

C.1 Hybrid architecture of PD and DRL controller . . . . . . . . . . . . . . . 124

# GLOSSARY

| | |
|---|---|
| **Action** | A decision or command made by the controller to transition the autonomous ship from one state to another.. |
| **Actor-Critic** | A hybrid RL algorithm that combines elements of both policy gradient methods (actor) and value-based methods (critic) to learn both a policy and a value function simultaneously.. |
| **Agent** | The entity or system (in this case, the autonomous ship) that interacts with the environment and makes decisions based on the received observations.. |
| **Autonomous Ship** | A vessel capable of operating and navigating without direct human intervention.. |
| **Controller** | A device or algorithm responsible for controlling the behavior and actions of a system or agent.. |
| **Deep Q-Network** | A deep neural network architecture used to approximate the Q-function in Q-learning, allowing for more complex and high-dimensional state spaces.. |
| **Deep Reinforcement Learning** | A branch of machine learning that combines deep neural networks with reinforcement learning algorithms to train agents to make sequential decisions in an environment.. |
| **Experience Replay** | A technique in DRL where the agent's experiences (state, action, reward, next state) are stored and sampled randomly during training to break the temporal correlation between consecutive samples.. |
| **Exploration-Exploitation Tradeoff** | The balance between exploring new actions and exploiting known actions to maximize |

| | |
|---|---|
| | the cumulative reward in RL.. |
| **Greedy Policy** | A policy that always selects the action with the highest estimated value, based on the learned Q-function or policy network.. |
| **Line-of-Sight (LOS)** | A straight line connecting an observer or reference point to a target object.. |
| **Markov Decision Process** | A mathematical framework used to model decision-making problems in RL, characterized by states, actions, rewards, and transition probabilities.. |
| **Observation** | Information perceived or sensed by the agent about the environment, which is used to determine the current state.. |
| **Policy** | A strategy or rule that guides the decision-making process of an agent in RL, mapping states to actions. |
| **Policy Gradient** | A class of RL algorithms that directly optimize the policy by estimating the gradient of expected rewards with respect to the policy parameters.. |
| **Policy Network** | The neural network component in DRL that maps states to actions and is trained to optimize the decision-making process.. |
| **Proportional-Derivative (PD) Controller:** | A control algorithm that combines proportional and derivative terms to generate control signals based on the error between the desired setpoint and the measured output.. |
| **Q-Learning** | A model-free RL algorithm that learns an action-value function, known as the Q-function, to estimate the value of taking a particular action in a given state.. |
| **Reinforcement Learning** | A type of machine learning where an agent learns to interact with an environment to maximize a cumulative reward signal.. |
| **Reward** | A numerical signal provided by the |

| | |
|---|---|
| | environment to evaluate the desirability of a particular action or state.. |
| **Simulation Environment** | A virtual or simulated representation of the real-world environment, used for training and evaluating the DRL-based controller without risks or costs associated with real-world testing.. |
| **State** | A representation of the current condition or configuration of the autonomous ship and its environment.. |
| **Value Function** | A function that estimates the expected return or value of being in a particular state or taking a specific action in RL.. |

# ABBREVIATIONS

| | |
|---|---|
| **AI** | Artificial Intelligence. |
| **APF** | Artificial Potential Field. |
| **BCS** | Body Coordinate System. |
| **CE** | Controller Effort. |
| **CR** | Collision Risk. |
| **DCPA** | Distance to Closest Point of Approach. |
| **DDPG** | Deep Deterministic Policy Gradient. |
| **DQN** | Deep Q-Network. |
| **DRL** | Deep Reinforcement Learning. |
| **GCS** | Global Coordinate System. |
| **KCS** | KRISO container ship. |
| **MDP** | Markov Decision Process. |
| **MMG** | Maneuvering Modelling Group. |
| **PD** | Proportional–Derivative. |
| **PPO** | Proximal Policy Optimization. |
| **RL** | Reinforcement Learning. |
| **TCPA** | Time to Closest Point of Approach. |

# NOTATION

| | |
|---|---|
| $\alpha_R$ | Effective inflow angle to rudder |
| $\beta$ | Drift angle |
| $\beta_w$ | Direction of wind with respect to X-axis of GCS |
| $\chi_e$ | Course angle error |
| $\delta$ | Actual rudder angle |
| $\delta_c$ | Commanded rudder angle |
| $\dot{e}$ | Rate of change of heading error |
| $\dot{\delta}$ | Rudder rate |
| $\dot{\delta}_{max}$ | Maximum rudder rate |
| $\dot{\psi}_d$ | Rate of change of desired heading angle |
| $\epsilon$ | Probability with which the RL agent takes a random action |
| $\eta$ | Ratio of propeller diameter to the rudder height |
| $\gamma_R$ | Flow straightening factor of hull |
| $\gamma_w$ | Relative wind angle with respect to ship |
| $\kappa$ | An experimental constant for expressing $u_R$ |
| $\pi$ | Policy of Markov Decision Process |
| $\psi$ | Current heading angle |
| $\psi_{desired}$ | Desired heading angle |
| $\rho$ | Density of water |

| | |
|---|---|
| $\rho_a$ | Density of air |
| $\theta$ | Weight and biases of neural network |
| $\varepsilon$ | Ratio of wake fraction at propeller and rudder positions |
| $a$ | Action of Markov Decision Process |
| $a_0$ | Constant curve fitting parameter for determining propeller thrust coefficient |
| $a_1$ | Linear curve fitting parameter for determining propeller thrust coefficient |
| $a_2$ | Quadratic curve fitting parameter for determining propeller thrust coefficient |
| $a_H$ | Rudder force increase factor |
| $A_R$ | Rudder Area |
| $A_x$ | Lateral projected areas of the hull |
| $A_y$ | Longitudinal projected areas of the hull |
| $B$ | Beam of the vessel |
| $C_{w\psi}$ | Yaw wind coefficient |
| $C_{wx}$ | Surge wind coefficient |
| $C_{wy}$ | Sway wind coefficient |
| $d$ | Depth moulded |
| $D_p$ | Propeller diameter |
| $d_c$ | Cross track error |
| $d_{em}$ | Draft |
| $d_{wp}$ | Distance to destination waypoint |

| | |
|---|---|
| $e$ | Heading error |
| $f_\alpha$ | Rudder lift gradient coefficient |
| $J$ | Advance coefficient |
| $K_d$ | Derivative gain |
| $K_p$ | Proportional gain |
| $K_T$ | Propeller thrust coefficient |
| $L$ | Length between perpendiculars |
| $L_{OA}$ | Length overall |
| $l_R$ | Correction of flow straightening factor to yaw rate |
| $N$ | External yaw moment |
| $n$ | Propeller revolution rate |
| $N_H$ | Hull hydrodynamic yaw moment |
| $N_R$ | Yaw moment due to rudder |
| $R$ | Cumulative sum of reward at end of episode |
| $r$ | Yaw rate |
| $r_1, r_2, r_3$ | Rewards associated with cross-track error, course angle error and distance to goal |
| $s$ | Observation state of Markov Decision Process |
| $t$ | Thrust deduction factor |
| $T_R$ | Rudder time constant |
| $t_R$ | Steering resistance deduction factor |

| | |
|---|---|
| $U$ | Design speed |
| $u$ | Surge velocity |
| $u_{rw}, v_{rw}$ | Velocity components of the ship relative to the wind |
| $U_R$ | Rudder inflow velocity |
| $u_R$ | Longitudinal inflow velocity component to rudder |
| $U_{wr}$ | Relative wind velocity |
| $v$ | Sway velocity |
| $v_R$ | Lateral inflow velocity component to rudder |
| $V_w$ | Velocity of wind |
| $w$ | Effective wake fraction |
| $W_\phi$ | Yaw wind moment |
| $W_x$ | Surge wind force |
| $W_y$ | Sway wind force |
| $X$ | External surge force |
| $x, y$ | Coordinate of current position of ship in GCS |
| $x_G$ | Longitudinal center of gravity of ship (LCG) |
| $x_i, y_i$ | Coordinate of initial position of ship in GCS |
| $x_g, y_g$ | Coordinate of goal/next waypoint in GCS |
| $X_H$ | Hull hydrodynamic surge force |
| $x_H$ | Longitudinal coordinate of acting point of the additional lateral force component induced by steering |

| | |
|---|---|
| $X_P$ | Surge force due to propeller |
| $X_R$ | Surge force due to rudder |
| $x_R$ | Location of the rudder with respect to midship |
| $Y$ | External sway force |
| $Y_H$ | Hull hydrodynamic sway force |
| $Y_R$ | Sway force due to rudder |

# CHAPTER 1

# INTRODUCTION

## 1.1 BACKGROUND AND MOTIVATION

pproximately 80-85% of marine-related accidents are attributed to human error, posing risks to human lives, the environment, and significant financial losses for ship owners Baker and McCafferty (2005). A notable incident exemplifying this is the Ever Given accident in the Suez Canal in 2021, where strong winds caused the vessel to deviate from its intended course. In light of these challenges, the emergence of advanced AI technology, particularly reinforcement learning (RL), offers promising opportunities for the maritime industry to mitigate such accidents. RL can provide effective solutions for ship path following, trajectory tracking, and collision avoidance, thus reducing the occurrence of hazardous incidents.

Nevertheless, the integration of RL-based automation solutions in the maritime sector has implications for the workforce and human involvement in maritime operations. While automation can enhance efficiency, minimize human errors, and enhance safety, concerns about job displacement and shifts in job roles arise. To address these challenges and ensure a successful transition, it is crucial to prioritize the collaboration between humans and AI systems. This can be accomplished through proactive workforce planning, implementation of upskilling and reskilling programs, and the creation of new roles that complement and augment the capabilities of AI systems. Effective collaboration among industry stakeholders, policymakers, and labor representatives is paramount to establish guidelines, regulations, and ethical frameworks that address the social and economic implications of AI adoption. This approach ensures that all stakeholders can maximize the benefits of AI while safeguarding the well-being and interests of the workforce.

## 1.2 LITERATURE REVIEW

The reinforcement learning (RL) method operates on a reward and penalty system, where agents are incentivized for actions that achieve goals and punished for those that have negative outcomes. As the agent interacts with the system, it learns from experience to consistently make choices leading to higher rewards and avoid those that lead to lower rewards. This type of data-driven controller is model-free, meaning that no model of the system being controlled is required. The system dynamics and control strategy are learned by the agent through interaction with the environment. Traditional RL uses Q-tables to store the action-choosing policy of the agent, which document the expected reward for all possible combinations of discrete states and actions. This table is updated every time the agent moves to a new state through a chosen action from the previous state. However, recent advancements in deep learning have led to the development of deep reinforcement learning (DRL), which replaces Q-tables with neural networks to store the policy of the agent (Mnih *et al.*, 2013). The utilization of deep reinforcement learning (DRL) techniques improves upon traditional RL methods, particularly in handling continuous or discrete state and action spaces. DRL leverages deep neural networks to approximate value functions or policy functions, enabling the handling of high-dimensional and continuous state spaces (Perera *et al.*, 2015). By incorporating deep learning, DRL algorithms can learn complex representations and capture intricate patterns in the environment. This allows for more efficient exploration and exploitation, leading to improved decision-making in dynamic and uncertain environments. Additionally, DRL can handle both continuous and discrete action spaces by employing various neural network architectures, such as deep Q-networks (DQN) or actor-critic models, which learn value or policy functions directly from raw sensory inputs.

Conventional autopilots typically rely on line of sight (LOS) guidance systems and proportional-integral-derivative (PID) controllers for achieving waypoint tracking in path following scenarios (Lekkas and Fossen, 2012; Moreira *et al.*, 2007). Traditional methods, including LOS guidance systems and PID controllers, are well-suited for situations with clearly defined paths and visible obstacles. These methods assume predictable system dynamics and allow control objectives to be expressed mathematically. In such cases, a separate path-planning algorithm is employed in conjunction with the controller to determine desired waypoints and adapt the path when static or dynamic obstacles are encountered. However, dynamically updating the path in real-time presents a significant challenge, requiring robustness and computational power. The emergence of AI-based control strategies offers a new approach where a single controller can handle both control and path planning functions without extensive real-time computations. This approach has demonstrated promise in specific applications, such as active heave compensation (Zinage and Somayajula, 2020, 2021; Zinage, 2021). However, there is a limited number of studies that effectively compare DRL-based controllers with traditional methods Sivaraj *et al.* (2022).

Reinforcement Learning (RL) methods have gained increasing attention in recent years as a promising approach to tackling path-planning problems in various domains. Many studies have explored the potential of RL methods for path planning, each with its own unique focus and contributions. For example, Wang *et al.* (2018) investigated the application of Q-learning algorithms to path planning and demonstrated that the trained agent could successfully plan paths and avoid static obstacles. However, their study did not take into account the dynamics of the vessel, and it was limited to planning a path that only avoided static obstacles. Another study by Shen *et al.* (2019) focused on collision avoidance of multiple ships using a deep Q-learning algorithm. They demonstrated that their algorithm effectively prevented collisions among three ships in simulations, where the dynamics of the ships were modelled using Nomoto's first-order model. However, their study also revealed that the waypoints could not be effectively tracked with obstacle. Furthermore, the study was conducted in a rectangular basin, and the results were not validated in a real-world setting. In contrast, Sivaraj *et al.* (2022) developed a Deep Q-network (DQN) for path following and heading control of a KVLCC2 tanker in calm water and waves. Their study focused on training different agents to track various headings, and they demonstrated that the trained agents were able to effectively control the vessel's heading and track the desired path. The study also compared the performance of the DQN-based method with a conventional proportional-integral-derivative (PID) controller and showed that the DQN-based method outperformed the PID controller in terms of tracking accuracy and robustness to disturbances.

Chen *et al.* (2019) developed a Q-learning-based model to implement a practical path following algorithm for under-actuated cargo ships using Nomoto's first order equation. Their study showed that the Q-learning-based model outperformed traditional path planning algorithms like RRTs and A*, with shorter path lengths and smoother turns. In a different study, Woo *et al.* (2019) trained a Deep Deterministic Policy Gradient (DDPG) based steering controller and used it in combination with a vector field guidance law to achieve path following. Their study compared the performance of the steering controller at different levels of training using simulations and experiments on an unmanned surface vehicle (WAM-V). However, their study did not compare the performance of the steering controller with traditional control approaches. Martinsen and Lekkas (2018) utilized DDPG models for straight-line path following and transfer learning to follow curved paths for three different vessels. Their study demonstrated the effectiveness of DRL-based guidance through simulations and showed that it could accumulate better rewards than traditional Line-of-Sight (LOS)-based guidance. Similarly, Zhou *et al.* (2019) used a Deep Q-network (DQN) for path planning of a single USV and USV formation, with a focus on the kinematics of the vessels rather than a dynamic model. The simulations showed that the developed method could effectively avoid collision with static obstacles while maintaining formation between three USVs.

There are several studies that compare the effectiveness of DRL-based controllers to a different path-planning algorithm. The dynamic window approach (Fox *et al.*, 1997) is one of the methods for path planning and collision avoidance. Chun *et al.* (2021) applied the Proximal Policy Optimization (PPO) algorithm to implement COLREGs (International

Regulations for Preventing Collisions at Sea). They compared the performance of the PPO algorithm with that of the A* algorithm and observed that the maximum Collision Risk (CR) was significantly reduced with the PPO algorithm. Woo and Kim (2020) utilized a Semi Markov Decision Process (SMDP) model to implement COLREGs and suggested a technique where a USV can perceive its surroundings by utilizing a grid map representation. The creation of this grid map representation is a vital element of their proposed approach. By providing details to the convolutional neural network (CNN), the CNN's ability to extract features can be adjusted to foster collision situation awareness.

The path planning and navigation study by Garrido *et al.* (2006) utilized the Voronoi diagram and fast marching method. Firstly, the safest regions are identified by employing the Voronoi diagram. Secondly, the fast marching method is applied to the areas extracted by the Voronoi diagram to determine the shortest path. Lazarowska (2020) utilized a discrete Artificial Potential Field (APF) to obtain a collision-free trajectory for an own ship in near real-time and optimized the trajectory using a Path Optimization Algorithm (POA). In a study conducted by Lyu and Yin (2019), an approach was suggested for achieving deterministic path planning for Unmanned Surface Vehicles (USVs) in a dynamic environment. The method employed a modified Artificial Potential Field (APF) to address the issue of collision avoidance in path planning. Chen *et al.* (2017) used the barrier function for obstacle avoidance and compared it with a potential field method and the Hamilton-Jacobi method (Takei *et al.*, 2010). Zhou *et al.* (2019) implemented cooperative path planning for a surface vessel using deep reinforcement learning. They showed that their method effectively avoided collision with static obstacles while maintaining a formation between three USVs. Xu *et al.* (2022) developed the path planning and dynamic collision avoidance (PPDC) algorithm to ensure the safe navigation of Unmanned Surface Vehicles (USVs). Their approach showed good performance in simulations with dynamic obstacles. Zhao *et al.* (2019) implemented the Proximal Policy Optimization (PPO) algorithm for path following and following COLREGs, and compared it against the traditional Proportional Integral Derivative (PID) controller.

Their study showed that the PPO algorithm had superior performance in terms of tracking accuracy and smoothness.

Several researchers have attempted to integrate path planning with compliant behavior with the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) for autonomous ships. By incorporating COLREGs into RL algorithms, autonomous ships can navigate in accordance with international maritime regulations, ensuring safe interactions with other vessels and avoiding collisions. For instance, Guo *et al.* (2020) implemented a path planning agent based on DDPG to choose continuous actions for rudder angle and speed to ensure COLREGs compliance. However, details of the vessel dynamics used for training were not provided. The study combined artificial potential field (APF) with the DDPG method and reported that this method converged faster and required fewer training episodes and time. Additionally, Shen and Guo (2016) used an actor-critic algorithm for path-following, with reward functions based on the error compared to a reference course. They later extended their method to include limiting lines and navigational polygons to prevent ship collisions (Shen *et al.*, 2019). In comparison, Layek *et al.* (2017) compared DDPG and NAF-based models, with both outperforming random search-based control during passage through a specified gate with random initial orientations. However, the study only considered the kinematics of the vessel and ignored simulation dynamics. Zhao *et al.* (2019) developed a PPO algorithm that utilized the LOS guidance system to navigate a ship simulated by the 3-DOF dynamic model. The study compared the DRL controller against a traditional PID controller and concluded that the RL controller resulted in a smaller cross-track error. Similarly, Heiberg *et al.* (2022) developed a PPO algorithm for path following and obstacle avoidance using collision risk theory, while Zhao and Roh (2019) proposed multi-ship collision avoidance based on DRL by categorizing target ships according to their regions relative to the autonomous ship, demonstrating the autonomous ship's ability to follow some of the COLREGs rules.

In terms of computational requirements, the RL controller demonstrates significant advantages once it is trained offline, demanding minimal time and computational power for executing actions, typically measured in microseconds. Although the RL controller may exhibit slightly longer computation times for waypoint tracking compared to the PD controller with ILOS guidance system, it remains relatively fast, delivering outputs within fractions of a second. Conversely, when obstacles are present, the traditional controller relies on a path planner that requires more time compared to the DRL-based approach.

The integration of RL-based automation solutions in the maritime industry carries implications for the workforce and human involvement in operations. While AI technologies offer benefits like enhanced efficiency and safety, concerns arise regarding potential job displacement and the need for reskilling. To ensure a smooth transition and maximize benefits for all stakeholders, effective management of AI technology integration becomes crucial. This entails comprehensive workforce planning, including training programs designed to equip employees with the necessary skills to work alongside AI systems.

Building trust and facilitating effective collaboration between human operators and AI systems are essential for successful integration. Additionally, addressing ethical and legal considerations, ensuring data privacy and accountability, and promoting fairness in decision-making are important steps for maximizing benefits and minimizing potential risks. Striking a balance between the potential of RL-based automation and the well-being and involvement of the maritime workforce is vital for the successful adoption of AI technologies in the industry.

## 1.3 OBJECTIVE

- Investigate the effectiveness of DRL-based controllers in a practical scenario involving a container ship.

- Evaluate the performance of DQN, DDPG, and PPO algorithms on the KCS hull with known hydrodynamic coefficients.

- Develop a DRL-based controller for the ship's path-following task considering a nonlinear model and incorporating environmental forces.

- Compare the performance of the DRL-based controller with a traditional PD controller utilizing an ILOS guidance system.

- Explore waypoint tracking accuracy of the DRL-based controller.

- Assess collision avoidance capabilities of the DRL-based controller with static and dynamic obstacles.

- Investigate a hybrid architecture of the controllers using multiple neural networks to enhance obstacle avoidance and waypoint tracking accuracy.

## 1.4 SCOPE

- Comparison of DRL-based methods and traditional methods for path following and collision avoidance in the context of ship navigation.

- Development of a DQN-based controller for path following of a ship governed by a nonlinear model.

- Evaluation of the performance of DQN, DDPG, and PPO algorithms on the KCS hull, a benchmark vessel with known hydrodynamic coefficients.

- Investigation of the application of DRL-based controllers in practical scenarios involving a container ship.

- Assessment of the DRL-based controller's performance in calm waters and in the presence of significant environmental disturbances.

- Verification of simulation results through field experiments.

- Exploration of collision avoidance with both static and dynamic obstacles.

- Examination of waypoint tracking accuracy.

The thesis is structured as follows to ensure a cohesive and organized presentation of the research. In Chapter 2, a detailed overview of the KCS vessel's dynamics is provided, including an exploration of the non-linear equations of motion and simulated manoeuvring tests. Chapter 3 delves into the fundamentals of RL and Q-learning algorithms, offering comprehensive explanations of the DQN, DDPG, and PPO algorithms. Moving on to Chapter 4, the problem of ship navigation is defined within the RL framework as a suitable learning environment. Chapter 5 presents the experimental and simulation results, examining the performance of RL agents across various manoeuvres, incorporating the modeling of wind forces, and analyzing path following in the presence of wind, static obstacles, and dynamic obstacles. Finally, Chapter 6 serves as a summary, discussing the obtained results, reflecting on the study's findings, and drawing meaningful conclusions.

# CHAPTER 2

# SHIP DYNAMICS

For the purpose of evaluating the control algorithms employed in this study, numerical simulations were conducted on the KCS vessel. The mathematical modelling of the ship dynamics is performed using the MMG (Maneuvering Modelling Group) model, as described by Yasukawa and Yoshimura (2015). The 3-DOF non-linear equations of motion, which include surge, sway, and yaw motions, are utilized to calculate the ship's maneuvering motions. An initial value problem is solved progressively at each time step, using a Runge-Kutta implicit solver. The commanded rudder angle $\delta_c$ was provided as an input at each time step. Table 2.1 presents the details of the KCS vessel that was used for simulating the ship dynamics.

Table 2.1: KCS ship parameters

| **Ship Parameter** | **Value** |
|---|---|
| Length between perpendiculars ($L$) | $230\,m$ |
| Length overall ($L_{OA}$) | $232.5\,m$ |
| Depth moulded ($d$) | $19\,m$ |
| Beam ($B$) | $32.2\,m$ |
| Draft ($d_{em}$) | $10.8\,m$ |
| Displacement | $53330.75\,tons$ |
| LCG ($x_G$) | $-3.408\,m$ |
| Radius of gyration | $57.5\,m$ |
| Design speed ($U$) | $12.347\,m/s$ |

The vessel's motion is tracked using two coordinate systems: a global coordinate system

(GCS) and a body coordinate system (BCS). The GCS is fixed to the Earth and has its z-axis pointing downwards, while the BCS is fixed to the vessel and moves with it. The origin of the BCS is located at the intersection of the midship, centerline, and waterline of the vessel, with its x-axis pointing towards the bow, y-axis towards starboard, and z-axis towards the keel. These coordinate frames are depicted in Fig. 2.1. The heading angle $\psi$ is defined as the angle between the x-axes of the GCS and BCS frames. The vessel's position and orientation are denoted by $\boldsymbol{\eta} = [x_o, y_o, \psi]^T$, where $x_o$ and $y_o$ denote the position of the BCS origin in the GCS. The velocity vector of the vessel in the BCS is $\mathbf{V} = [u, v, r]^T$, where $u$, $v$, and $r$ represent the surge, sway, and yaw velocities, respectively, expressed in the BCS. The vessel's speed is given by $U = \sqrt{u^2 + v^2}$, and the drift angle ($\beta$) is defined as the angle between the total velocity vector and the longitudinal direction of the vessel, given by $\beta = \tan^{-1}(-v/u)$. The kinematics of ship motion is described by (2.1).

$$\dot{\boldsymbol{\eta}} = [R(\psi)]\mathbf{V} \tag{2.1}$$

where $[R(\psi)]$ represents the rotation matrix given by

$$[R(\psi)] = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Any vector in BCS when pre-multiplied by the rotation matrix will result in the same vector expressed in GCS.

The MMG model used for simulating ship maneuvering (Yoshimura and Masumoto, 2012) is shown in (2.3).

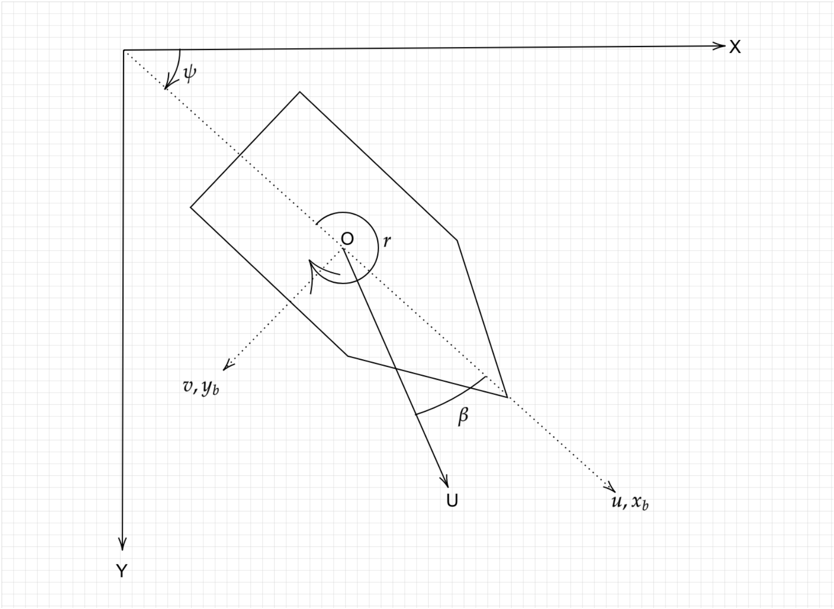


Figure 2.1: Representation of ship kinematic variables

$$\begin{aligned}
&(m + m_x)\dot{u} - mvr - mx_G r^2 = X \\
&(m + m_y)\dot{v} + mx_G \dot{r} + mur = Y \\
&(I_{zz} + J_{zz})\dot{r} + mx_G \dot{v} + mx_G ur = N
\end{aligned} \tag{2.3}$$

In (2.3), variable $m$ represents the mass of the ship, while $I_{zz}$ corresponds to the second moment of inertia in the yaw direction. The added masses associated with surge, sway, and yaw are represented by $m_x$, $m_y$, and $J_{zz}$, respectively. The variables $X$, $Y$, and $N$ represent the external forces acting on the vessel in the surge, sway, and yaw directions, respectively. All these variables are expressed in the BCS coordinate frame.

To non-dimensionalize the surge and sway equations of motion, both sides of the equations are divided by $\frac{\rho}{2}U^2 L d_{em}$, where $\rho$ represents the density of seawater, $L$ is the vessel's length, $U$ is the vessel's design speed, and $d_{em}$ is the vessel's draft. The yaw equation of motion is similarly non-dimensionalized by dividing both sides by $\frac{\rho}{2}U^2 L^2 d_{em}$. This normalization method follows the prime-II system described by Fossen (1999). Table

Table 2.2: Prime-II system of normalization

| Parameter | Prime-II system |
|---|---|
| Length | $L$ |
| Velocity | $U$ |
| Angular velocity | $U/L$ |
| Time | $L/U$ |
| Acceleration | $U^2/L$ |
| Angular acceleration | $U^2/L^2$ |
| Mass | $0.5\rho L^2 d_{em}$ |
| Force | $0.5\rho U^2 L d_{em}$ |
| Moment | $0.5\rho U^2 L^2 d_{em}$ |

Table 2.3: Ship parameters

| Parameter | Non-dimensional value |
|---|---|
| Surge added mass ($m_x$) | 0.006269 |
| Sway added mass ($m_y$) | 0.155164 |
| Yaw added mass moment ($J_{zz}$) | 0.009268 |
| Yaw mass moment of inertia ($I_{zz}$) | 0.011432 |
| Mass of the vessel ($m$) | 0.182280 |

2.2 shows the non-dimensionalization factors for various quantities, and the resulting non-dimensional equations of motion are given by 2.4.

$$\begin{aligned}
(m' + m'_x)\dot{u}' - m'v'r' - m'x'_G r'^2 &= X' \\
(m' + m'_y)\dot{v}' + m'x'_G\dot{r}' + m'u'r' &= Y' \\
(I'_{zz} + J'_{zz})\dot{r}' + m'x'_G\dot{v}' + m'x'_G u'r' &= N'
\end{aligned} \tag{2.4}$$

In Table 2.2, a non-dimensional factor is specified and denoted by $(.)'$. However, in the following sections, to simplify the notation, we will avoid using the prime symbol and assume that all quantities are already non-dimensionalized based on the factors presented in Table 2.2. Additionally, Table 2.3 provides the non-dimensional mass and added mass terms.

The non-dimensional external forces and moments can be decomposed into components due to hull, propeller and rudder as shown in (2.5).

$$\begin{aligned} X &= X_H + X_R + X_P \\ Y &= Y_H + Y_R \\ N &= N_H + N_R \end{aligned} \tag{2.5}$$

where the subscripts $H$, $R$ and $P$ represent the hull, rudder, and propeller effects respectively.

## 2.1 FORCES DUE TO HULL

In (2.6), the forces and moments exerted on the hull by the surrounding fluid can be expressed as a Taylor series with respect to the variables $u$, $\beta$, and $r$. The hydrodynamic coefficients used for the KCS vessel in (2.6), which are non-dimensional, were obtained from Yoshimura and Masumoto (2012) and are presented in Table 2.4. Alternatively, these coefficients can also be estimated by applying system identification methods to data collected from experiments conducted with free-running ship models, as demonstrated by previous studies (Vijay and Somayajula, 2022; Deogaonkar *et al.*, 2023).

$$\begin{aligned} X_H &= X_0 u^2 + X_{\beta\beta}\beta^2 + (X_{\beta r} - m_y)\beta r + X_{rr} r^2 + X_{\beta\beta\beta\beta}\beta^4 \\ Y_H &= Y_\beta \beta + (Y_r - m_x) r + Y_{\beta\beta\beta}\beta^3 + Y_{\beta\beta r}\beta^2 r + Y_{\beta rr}\beta r^2 + Y_{rrr} r^3 \\ N_H &= N_\beta \beta + N_r r + N_{\beta\beta\beta}\beta^3 + N_{\beta\beta r}\beta^2 r + N_{\beta rr}\beta r^2 + N_{rrr} r^3 \end{aligned} \tag{2.6}$$

## 2.2 FORCES DUE TO PROPELLER

The non-dimensional propeller thrust acting in the surge direction can be calculated using (2.7)

$$X_p = 2(1-t) K_T {D_p}^4 n^2 \frac{L}{d_{em}} \tag{2.7}$$

Table 2.4: Ship hydrodynamic coefficients

| Parameter | Non-dimensional value |
|---|---|
| $X_0$ | -0.0167 |
| $X_{\beta\beta}$ | -0.0549 |
| $X_{\beta r} - m_y$ | -0.1084 |
| $X_{rr}$ | -0.0120 |
| $X_{\beta\beta\beta\beta}$ | -0.0417 |
| $Y_\beta$ | 0.2252 |
| $Y_r - m_x$ | 0.0398 |
| $Y_{\beta\beta\beta}$ | 1.7179 |
| $Y_{\beta\beta r}$ | -0.4832 |
| $Y_{\beta rr}$ | 0.8341 |
| $Y_{rrr}$ | -0.0050 |
| $N_\beta$ | 0.1111 |
| $N_r$ | -0.0465 |
| $N_{\beta\beta\beta}$ | 0.1752 |
| $N_{\beta\beta r}$ | -0.6168 |
| $N_{\beta rr}$ | 0.0512 |
| $N_{rrr}$ | -0.0387 |

The (2.8) provides the value of the propeller thrust coefficient $K_T$, where $D_p$ represents the propeller diameter, $n$ is the propeller revolution rate, and $t$ is the thrust deduction factor. The purpose of including the thrust deduction factor is to account for the additional resistance experienced when the propeller operates behind the hull, as opposed to bare hull resistance. Curve fitting of the propeller open water test data is used to determine the value of $K_T$ for the KCS vessel.

$$K_T = a_0 + a_1 J + a_2 J^2 \tag{2.8}$$

Here $J$ represents the advance coefficient and is given by (2.9)

$$J = \frac{u(1-w)}{nD_p} \tag{2.9}$$

where $w$ represents the effective wake fraction. The effective wake fraction $w$ accounts for the reduction in inflow fluid velocity to the propeller when operating in the wake of

Table 2.5: Propeller parameters

| Parameter | Non-dimensional value |
|---|---|
| $t$ | 0.207 |
| $D_p$ | 0.03435 |
| $n$ | 35.86 |
| $w$ | 0.355 |
| $a_0$ | 0.5228 |
| $a_1$ | -0.4390 |
| $a_2$ | -0.0609 |

the hull. The values of propeller parameters are specified in Table 2.5.

## 2.3 FORCES DUE TO RUDDER

The surge and sway forces and the yaw moment due to the rudder are given by

$$
\begin{aligned}
X_R &= -(1 - t_R)F_N sin\delta \\
Y_R &= -(1 + a_H)F_N \cos\delta \\
N_R &= -(x_R + a_H x_H)F_N \cos\delta
\end{aligned}
\tag{2.10}
$$

where $\delta$ represents the instantaneous rudder angle; $t_R$, which is the steering resistance deduction factor; $x_R$, which denotes the non-dimensional location of the rudder relative to midship; and $x_H$, which refers to the non-dimensional position of the point where additional lateral force is applied. The values of these parameters have been obtained from a prior study by Yoshimura and Masumoto (2012) and are presented in Table2.6. The non-dimensional rudder normal force, $F_N$, can be calculated using (2.11).

$$
\begin{aligned}
F_N &= \frac{A_R}{Ld_{em}} f_\alpha U_R^2 \sin\alpha_R \\
\alpha_R &= \delta - \tan^{-1}\frac{-v_R}{u_R}
\end{aligned}
\tag{2.11}
$$

where $A_R$ is the rudder area, $U_R$ is the non-dimensional rudder inflow velocity and $f_\alpha$ is the gradient of rudder lift coefficients. $f_\alpha$ can be approximated as a function of rudder aspect ratio ($\lambda$), as shown in Eq. 2.12.

$$f_\alpha = \frac{6.13\lambda}{(2.25 + \lambda)} \tag{2.12}$$

$v_R$ and $u_R$ in (2.11) are the lateral and in-line velocity components ($U_R = \sqrt{v_R^2 + u_R^2}$) and can be computed as shown in (2.13).

$$\begin{aligned} u_R &= \varepsilon(1 - w) \times \sqrt{\eta \left(1 + \kappa \left(\sqrt{\left(1 + 8\frac{K_T}{\pi J^2}\right)} - 1\right)\right)^2 + (1 - \eta)} \\ v_R &= \gamma_R (v + r l_R) \end{aligned} \tag{2.13}$$

In (2.13), the flow straightening factor to yaw-rate correction is denoted by $l_R$, while $\gamma_R$ represents the flow straightening factor of the hull. The values for these constants are provided in Table 2.6. The ratio of the propeller diameter to the rudder height is denoted by $\eta$ and can be computed as $\eta = \frac{D_p}{h_R}$. The constants $\varepsilon$ and $\kappa$ are obtained from experiments conducted by Yoshimura and Masumoto (2012).

## 2.4 RUDDER ANGLE VARIATION

In real-world situations, it is not possible to change the rudder angle instantaneously due to the high rudder rate $\dot{\delta}$ it would require. This research employs a first-order equation, as presented in (2.14), to regulate the actual rudder angle $\delta$ which is influenced by the commanded rudder angle $\delta_c$. The commanded rudder angle can be changed abruptly, but the actual rudder angle $\delta$ evolves slowly as per the first-order equation. By reducing the value of $T_R$, the response time can be accelerated.

$$T_R \dot{\delta} + \delta = \delta_c \tag{2.14}$$

Table 2.6: Rudder force parameters

| Parameter | Value |
|---|---|
| $\frac{A_R}{Ld_{em}}$ | 0.0182 |
| $t_R$ | 0.258 |
| $a_H$ | 0.361 |
| $x_H$ | -0.436 |
| $x_R$ | -0.5 |
| $l_R$ | -0.755 |
| $\gamma_R$ (starboard) | 0.492 |
| $\gamma_R$ (port) | 0.338 |
| $\kappa$ | 0.633 |
| $\varepsilon$ | 0.956 |
| $\lambda$ | 2.164 |
| $\eta$ | 0.7979 |

In addition, the rudder rate $\dot{\delta}$ is also saturated at $\dot{\delta}_{max}$. So the rudder rate is given by:

$$\dot{\delta} = \begin{cases} \frac{\delta_c - \delta}{T_R} & \text{if } \left|\frac{\delta_c - \delta}{T_R}\right| \leq \dot{\delta}_{max} \\ \dot{\delta}_{max} & \text{if } \frac{\delta_c - \delta}{T_R} > \dot{\delta}_{max} \\ -\dot{\delta}_{max} & \text{if } \frac{\delta_c - \delta}{T_R} < -\dot{\delta}_{max} \end{cases} \tag{2.15}$$

In this study, the non-dimensional rudder time-constant $T_R$ is taken as 0.1 and $\dot{\delta}_{max}$ is chosen as 5° per second (Fossen (2011)) for the full-scale ship.

# CHAPTER 3

# REINFORCEMENT LEARNING ALGORITHMS

## 3.1 INTRODUCTION

RL is a type of machine learning where agents learn to make optimal decisions by interacting with an environment to accumulate rewards. Through trial and error, the agent learns which actions to take in a particular state to maximize rewards. This learning environment is known as a Markov decision process (MDP), which is suitable for representing ship dynamics. The agent's policy, denoted by $\pi(s)$, governs the action taken in a state $s$. An episode is terminated either when a maximum number of time steps is reached or if any termination condition is satisfied. The agent's goal is to maximize the cumulative reward obtained in each episode, known as the episode returns.

$$R = \sum_{t=0}^{T} r_t \tag{3.1}$$

In RL, the value function $V(s)$ is defined as the expected sum of discounted rewards obtained from a given state, as shown in (3.2). Here $\gamma \in [0, 1]$ denotes the discount factor, that adjust the weightage given to rewards obtained in future time steps.

$$V(s) = \mathbb{E}_\pi \left[ r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{T-t} r_T \mid s_t = s \right] \tag{3.2}$$

Q-learning is a well-known reinforcement learning (RL) algorithm that estimates the value of each possible action at a particular state known as the Q-value. The Q-value for a state-action pair is represented as $Q(s, a)$, and corresponds to the expected discounted sum of rewards that the agent can receive by taking the action $a$ from the state $s$. The Q-value for a state-action pair can be computed using the Bellman equation, which sums up the reward obtained by executing action $a$ at state $s$ and the discounted rewards obtained by choosing optimal actions at the subsequent time steps, as shown in (3.3). Expectation refers to the predicted value of rewards, considering available information. It involves a calculation that combines each potential value of the reward with its associated probability density function through summation. By estimating the expected value of rewards, RL agents can make informed decisions and update their policies to maximize cumulative rewards over time. The calculation of expectations plays a fundamental role in modeling and optimizing the RL process. Therefore, The agent improves its Q-value estimation through experience gained from multiple trials.

$$
\begin{aligned}
Q(s,a) &= \mathbb{E}_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots s_t = s, a_t = a] \\
Q(s,a) &= r(s,a) + \gamma \max_{a'} Q(s', a')
\end{aligned}
\tag{3.3}
$$

## 3.2 DEEP Q-LEARNING

Q-Learning has a significant limitation that it can only be used when both the action space and state space are discrete, which is not the case in many real-world scenarios. Although a naive approach might be to discretize the state space and action space, a fine discretization would require a large number of Q-values to be stored and is many times infeasible. To address this issue, function approximators can be used to substitute the Q-table used in Q-learning. Artificial neural networks (ANNs) are a popular choice for function approximators due to their excellent representational capabilities. This has given rise to a new class of RL algorithms known as deep reinforcement learning (DRL).

In deep Q-learning, the process involves inputting the observation spaces into a Q-network, which then estimates the Q-values for each possible action. Unlike traditional Q-learning, deep Q-learning can handle continuous state spaces. The Q-value is denoted as $Q(s,a,\theta)$, where $\theta$ represents the network's parameters. During training, an $\epsilon$-greedy policy is typically employed to encourage exploration rather than always selecting the

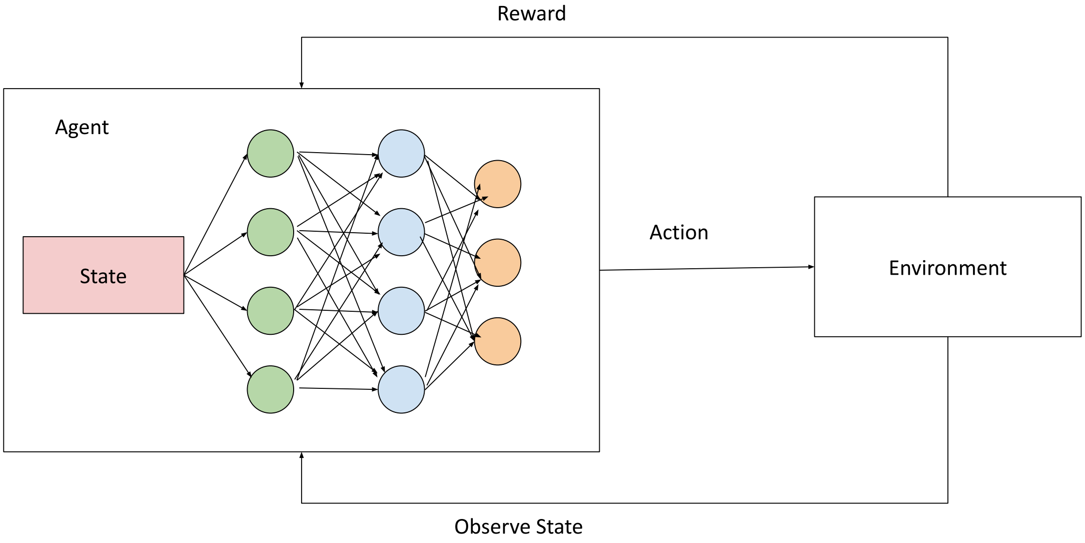


Figure 3.1: DQN algorithm flowchart

action with the highest Q-value. The $\epsilon$-greedy policy is given by Equation (3.4), where the parameter $\epsilon$ determines the probability of the agent choosing a random action. In this research, the value of $\epsilon$ starts at 1 and gradually decreases linearly to 0 after a specified number of episodes (which is a hyperparameter in this work).

$$\pi(s) = \begin{cases} \arg\max_a Q(s,a) \text{with a probability of } 1-\epsilon \\ \text{random action with a probability of } \epsilon \end{cases} \tag{3.4}$$

The network parameters are updated frequently to improve the Q-values output by the network, and this in turn allows the policy to converge faster. Network parameters are updated by backpropagation with the help of an optimizer that tries to minimize the loss function given by (3.5).

$$L(\theta) = \mathbb{E}_{s,a,r,s'}\left[(y - Q(s,a;\theta))^2\right] \tag{3.5}$$

Here the notation $\mathbb{E}_{s,a,r,s'}[.]$ represents the expected value across all feasible state transitions $(s, a, r, s')$. A state transition $(s, a, r, s')$ indicates a movement of the agent from state $s$ to $s'$ by executing an action $a$ and earning a reward $r$. It is worth noting that $y$ is the summation of the obtained reward $r(s, a)$ during the transition from $s$ to $s'$ and the anticipated reward from that point onwards, assuming that the agent selects the action with the maximum Q-value, as represented in (3.6). The second term in (3.6) denotes the anticipated future rewards.

$$y = r(s, a) + \gamma \max_{a'} Q(s', a'; \theta') \tag{3.6}$$

The TD (temporal difference) loss is defined as $[y - Q(s, a; \theta)]$. In (3.6), $\theta'$ refers to the parameters of the target network. The target value, denoted as $y$, is typically determined using a target network in DQN, which is essentially a copy of the Q-network but updated at a slower rate with a soft update factor $\tau$ to promote stable learning. To update the Q-network, a set of transitions are randomly sampled from the replay buffer and the TD loss is computed for these samples according to (3.5), after which the network is updated through backpropagation.

The DQN algorithm used in this study is expressed in a condensed form in Algorithm 1.

### 3.3 DEEP DETERMINISTIC POLICY GRADIENT

Deep Deterministic Policy Gradient (DDPG) (Lillicrap *et al.*, 2015) is a DRL algorithm that uses the actor-critic framework to extend Q-learning to a continuous action space. The policy and Q-value functions are both estimated by neural networks, namely the actor and critic networks respectively. In policy gradient methods, the actor network directly represents the agent's policy. The actor network takes as input the current state and outputs the action based on the policy encoded by the network. The critic network takes as input both the state and action to predict the Q-value. DDPG also makes use of target networks like the DQN algorithm.

**Algorithm 1** DQN algorithm

```
Initialise Q-network and target network with random parameters $\theta_0$ and $\theta_{0'}$
Initialise an empty experience replay buffer $\mathcal{D}$
for Episode = 1,2...N do
    for t=1,2...T do
        Choose action $a_t$ as per (3.4)
        Obtain reward $r_t$ and next state $s_{t+1}$
        Add transition $(s_t, a_t, r_t, s_{t+1})$ to replay buffer $\mathcal{D}$
        End episode if termination conditions are met
        if time step % update frequency ==0 then
            Sample $M$ random transitions from $\mathcal{D}$
            Compute loss as below using $y_i$ from (3.6) :
            $L(\theta) = \sum_{i=0}^{M} \frac{1}{M} [y_i - Q(s_i, a_i; \theta)]^2$
            Update the network using the computed loss
            Update target network as: $\theta' = \tau\theta + (1 - \tau)\theta'$
        end if
    end for
end for
```

In DDPG, noise is added to the action to aid the agent to explore more during training. The behavioural policy can be denoted as:

$$a_t = \mu\left(\mathbf{s}_t\right) + \mathbb{N}_t \tag{3.7}$$

Where $\mathbb{N}_t$ is the noise added to the policy and $\mu(s_t)$ is the current policy. Noise is usually added through a correlated Ornstein-Uhlenbeck process or an uncorrelated Gaussian distribution.

The critic network uses squared TD error as its loss function. The actor-network uses the policy gradient loss which is given below. In (3.8), $\theta^Q$ denotes the parameters of the critic network and $\theta^\mu$ denotes the parameters of the actor-network.

$$\nabla_{\theta^\mu} J = \left[ \nabla_a Q\left(s, a \mid \theta^Q\right)\Big|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu\left(s \mid \theta^\mu\right)\Big|_{s=s_t} \right] \tag{3.8}$$

The DDPG algorithm is specified below in Algorithm 2

**Algorithm 2** DDPG algorithm

```
Initialise actor and critic networks and target networks with random parameters $\theta^{\mu}$ and $\theta^{Q}$
Initialise target networks with parameters $\theta^{Q'} \leftarrow \theta^{Q}, \theta^{\mu'} \leftarrow \theta^{\mu}$
Initialise an empty experience replay buffer $\mathcal{D}$
for Episode = 1,2...N do
    for t=1,2...T do
        Choose action $a_t$ according to (3.7)
        Obtain reward $r_t$ and next state $s_{t+1}$
        Add transition $(s_t, a_t, r_t, s_{t+1})$ to replay buffer $\mathcal{D}$
        End episode if termination conditions are met
        Sample $M$ random transitions from $\mathcal{D}$
        Update the critic network using the loss:
        $L = \frac{1}{N} \sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^{Q}\right)\right)^2$ where
        $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$
        Update the actor-network:
        $\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q\left(s, a \mid \theta^{Q}\right)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu\left(s \mid \theta^{\mu}\right)\Big|_{s_i}$
        if time step % target update frequency ==0 then
            Update target networks:
            $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q}$
            $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
        end if
    end for
end for
```

### 3.4 PROXIMAL POLICY OPTIMIZATION

Proximal Policy Optimization (PPO) (Schulman *et al.*, 2017) algorithm is an actor-critic, on-policy RL algorithm that is based on trust region policy optimization (TRPO, Schulman *et al.* (2015)). It is an on-policy algorithm, meaning the agent's current policy and the behavioural policy which is used to collect data for training are the same. Hence it does not make use of a replay buffer, unlike the DQN and DDPG algorithms. PPO uses a stochastic policy and trains it in an on-policy way. Additionally, an entropy bonus is added to improve exploration. In an iteration, a number of transitions are collected using the same policy before updating the networks.

It reduces computation from TRPO by introducing a constrained surrogate objective function as the actor loss function. There are variants of this surrogate objective, KL penalty and clipped ratio objective functions. The clipped ratio surrogate objective is commonly used and it is given by:

$$\begin{aligned} r_t(\theta) &= \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} \\ L^{CLIP}(\theta) &= \hat{\mathrm{E}}[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t] \end{aligned} \tag{3.9}$$

In (3.9), $\pi_\theta\ (a_t \mid s_t)$ is the probability of action $a_t$ under the current policy, $\pi_{\theta_{old}}\ (a_t \mid s_t)$ is the probability with the old policy and $\epsilon$ is the clip ratio. $\hat{A}_t$ is the estimated advantage at time step $t$. In RL, the advantage is defined as the difference between the Q-value $Q(s,a)$ and the value function $V(s)$. The advantage is usually computed using the Generalized advantage estimation (GAE) method. The value network loss is the mean squared error between the return and the value function, as shown in (3.11).

$$\begin{aligned} \hat{A}(s,a) &= Q(s,a) - V(s) \\ \hat{A}(s,a) &= r + \gamma V(s') - V(s) \end{aligned} \tag{3.10}$$

$$L^{V}(\phi) = -\sum_{i=1}^{N}\sum_{t}\left(\sum_{t'>t}\gamma^{t'-t} r_i^{t'} - V_{\phi}\left(s_i^{t}\right)\right)^{2} \tag{3.11}$$

The PPO algorithm is described in detail below in Algorithm 3

**Algorithm 3** PPO algorithm

```
Initialise actor network $\pi_\theta$ and value network $V_\phi$
for Iteration = 1,2...M do
    for Episode = 1,2...N do
        for t=1,2...T do
            Choose action $a_t$ according to $\pi_\theta$
            End episode if termination conditions are met
        end for
        Estimate advantages $\hat{A}_1, \hat{A}_2 ... \hat{A}_T$ using $V_\phi$
    end for
    for Epochs=1,2...K do
        Update the actor using surrogate objective $L^{CLIP}$ according to (3.9)
        Update value network using critic loss given by (3.11)
    end for
end for
```

# CHAPTER 4

# RL-FRAMEWORK

This chapter discusses the details of the RL framework applied to ship navigation and obstacle avoidance. An observation state of the vessel is provided as input to the RL agent at every time step based on which the agent takes actions to maneuver the ship towards the goal position. The observation states and rewards have to be designed appropriately to achieve the desired outcome.

Tensorflow-agents, the official DRL implementation from the prominent machine learning library Tensorflow (Abadi *et al.*, 2015), is used in this study to efficiently train and test DRL agents with several algorithms.

This chapter is divided into four sections. First section discusses the 3-action state-based DQN controller and its comparison with the traditional PD controller with ILOS guidance system in simulations and experiments for waypoint tracking. The second section emphasizes on comparison between different DRL controllers, namely DQN (5-action state), PPO and DDPG in various manoeuvres and in presence of external disturbances like winds for waypoint tracking. The third section discusses about the DQN and DDPG-based controllers on static obstacle avoidance. Finally, the last section talks about the DQN and DDPG controller with dynamic obstacle avoidance and using a hybrid network to maneuver through calm water.

## 4.1 WAYPOINT TRACKING

In this section, DRL based controller is trained for waypoint tracking and path following. Fig. 4.1 shows the schematic representation of the problem statement for waypoint tracking.

### 4.1.1 Observation States

At each time step, the agent receives an observation state vector that reflects its current state and uses this information to make a decision about which action to take. This observation state vector comprises four variables: *cross-track error ($d_c$), course angle error ($\chi_e$), yaw rate($r$),* and *distance to destination ($d_{wp}$).*

#### 4.1.1.1 Cross-track error

The coordinates of the initial and goal waypoints are denoted by $(x_i, y_i)$ and $(x_g, y_g)$ respectively, while the ship's coordinates at an intermediate time step are denoted by $(x, y)$. The perpendicular distance of the ship's current coordinates from the line joining the initial and destination waypoints is defined as the cross-track error ($d_c$). To compute the cross-track error, two vectors $\hat{v}_1$ and $\vec{v}_2$ are defined as illustrated in Fig. 4.1. The vector $\hat{v}_1$ represents a unit vector in the direction of the line joining the initial and destination waypoints, while $\vec{v}_2$ points from the current location of the ship towards the goal. The cross-track error $d_c$ can be obtained as the cross product of these two vectors, $\hat{v}_1$ and $\vec{v}_2$.

$$\begin{aligned}
\vec{v}_1 &= (x_g - x_i)\hat{\imath} + (y_g - y_i)\hat{\jmath} \\
\hat{v}_1 &= \frac{\vec{v}_1}{|\vec{v}_1|} \\
\vec{v}_2 &= (x_g - x)\hat{\imath} + (y_g - y)\hat{\jmath} \\
d_c &= \vec{v}_2 \times \hat{v}_1
\end{aligned} \tag{4.1}$$

#### 4.1.1.2 Course-angle error

The course angle error refers to the angle between the ship's instantaneous velocity and the desired heading, and it is calculated using the formula given in (4.2). The calculation involves determining the smallest signed angle (ssa) (Fossen, 1999) between the difference in the ship's velocity direction and the direction towards the goal waypoint.

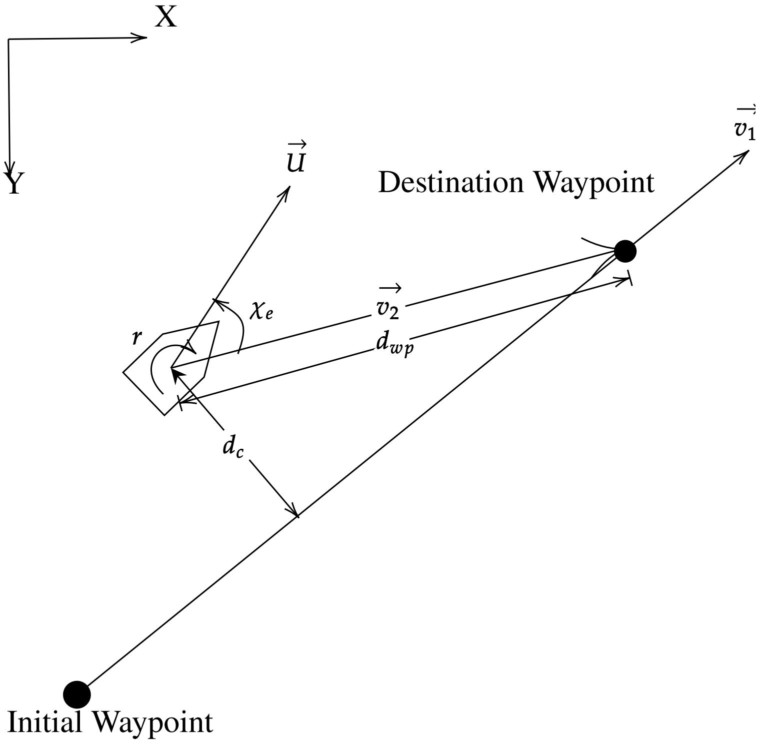


Figure 4.1: Depiction of the waypoint tracking problem

$$\chi_e = ssa\left(arctan2(\vec{U}) - arctan2(\vec{v_2})\right) \tag{4.2}$$

Here $\vec{U}$ is the ship's instantaneous velocity represented in global coordinates and can be calculated as shown in (4.3).

$$\begin{aligned} \vec{U} &= \dot{x}\vec{i} + \dot{y}\vec{j} \\ \dot{x} &= u\cos(\psi) - v\sin(\psi) \\ \dot{y} &= u\sin(\psi) + v\cos(\psi) \end{aligned} \tag{4.3}$$

#### 4.1.1.3 Distance to destination

The distance to the destination waypoint $d_{wp}$ is defined as shown in (4.4).

$$d_{wp} = \sqrt{(x_g - x)^2 + (y_g - y)^2} \tag{4.4}$$

#### 4.1.1.4 Yaw-rate

The yaw-rate $r$ is defined as the vessel's rotational velocity in the BCS's yaw direction.

### 4.1.2 Action Space

The action available to the agent at each time step is the commanded rudder angle $\delta_c$. The relationship between $\delta_c$ and the actual rudder angle $\delta$ can be found in (2.15). It is important to note that the agent is only responsible for controlling the commanded rudder angle, while the actual rudder angle $\delta$ continues to change gradually in accordance with (2.14) and (2.15).

### 4.1.3 Reward Structure

In an RL system, the selection of appropriate reward functions can significantly impact its decision-making process. In this particular scenario, the reward functions were specifically designed to consider the cross-track error, course angle, and distance to the destination waypoint. Specifically, the reward function associated with the cross-track error ($d_c$) is given by:

$$r_1 = 2\exp\left(\frac{-d_c^2}{12.5}\right) - 1 \tag{4.5}$$

The function is chosen such that at each time step, the reward lies between $-1$ and $1$. In addition, the exponential coefficient of this function is determined in such a way that if the cross-track error ($d_c$) exceeds a threshold of 3, the corresponding reward becomes negative. The reward associated with the error in course angle ($\chi_e$) is given by:

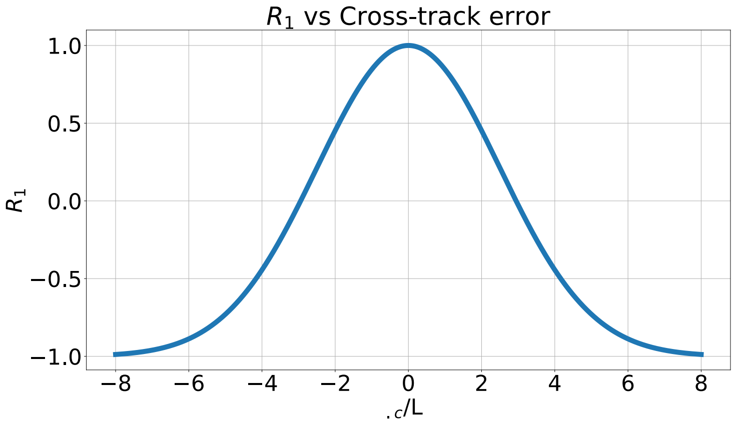


Figure 4.2: Cross track reward

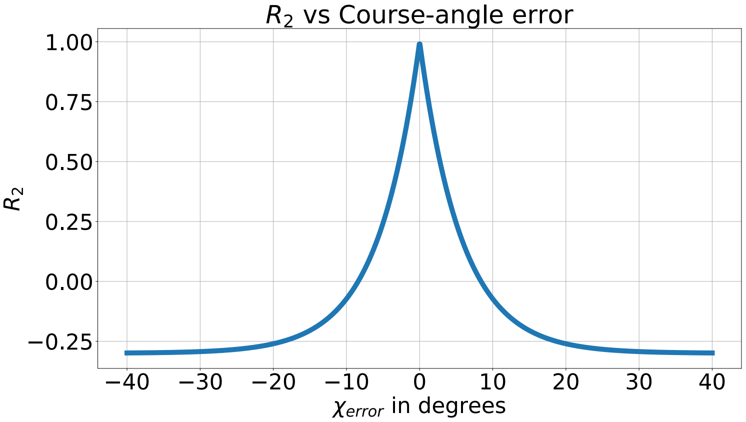


Figure 4.3: Course angle reward

$$r_2 = 1.3 \exp\left(-10\,|\chi_e|\right) - 0.3 \tag{4.6}$$

The selection of the reward function is such that it maintains the reward value within the range of $-0.3$ to 1 at every time step. The coefficient of the exponential term is carefully determined in such a way that for course angle errors greater than $8.5^0$, the reward associated with it becomes negative. It is noteworthy that the exponential function has the cross-track error $d_c^2$ as its argument and $|\chi_e|$ for the course angle error, resulting in the agent giving more importance to reducing the cross-track error than the course angle error. This strategy is employed to ensure that the agent prioritizes reducing cross-track error over course angle error in each time step.

It is important to mention that in order to prevent the agent from loitering in the environment and to encourage it to quickly reach the goal, the total reward obtained during training should be negative. For this purpose, a new reward based on the distance to the destination waypoint ($d_{wp}$) is introduced and is given by

$$r_3 = \frac{-d_{wp}}{4} \tag{4.7}$$

By setting the coefficient to $1/4$ in the reward function, a negative reward is obtained for the entire episode. This approach ensures that when the agent is far from the goal waypoint (i.e., farther than $4L$), it prioritizes reducing the distance to the goal over maintaining the course. The value of 4 is chosen based on the observation that the total episode reward is negative, and the combination of the reward structure for $r_1$ and $r_2$ with the factor of 4 results in a trajectory that gradually decreases course angle error and cross-track error as the distance decreases. Alternative values for the coefficient are possible, but they would lead to a different rate of decrease in course angle error and cross-track error with distance. For instance, a reward of $-d_{wp}/8$ with equal weights for $r_1$ and $r_2$ would give more weight to the cross-track error than the distance to the goal when the vessel is $8L$ away from the goal waypoint. However, by setting the reward to $-d_{wp}/4$, this transition point shifts to $4L$ away from the goal waypoint.

The total reward obtained by the agent throughout an episode, also known as episode return, can be computed by summing up the rewards obtained by the agent at each time step. Here, $r_m$ denotes the reward obtained by the agent at time step $m$, then the episode return can be expressed as the sum of all rewards accumulated by the agent from the beginning to the end of the episode, as shown in (4.8).

$$
\begin{aligned}
r_m &= r_1 + r_2 + r_3 \\
R &= \sum_{m=0}^{n} r_m
\end{aligned}
\tag{4.8}
$$

In (4.8), $r_m$ is the reward at time step $m$ and $R$ is the episode return, which is the cumulative sum of rewards obtained at each time step.

### 4.1.4 Training Process

At the start of each episode, the ship's position is set to the origin, oriented towards the positive X-axis ($\psi = 0$) of the GCS, and with a velocity matching the design speed. There is no initial acceleration in any of the 3-DOF and no initial velocity in the sway and yaw motions.

During training, the initial waypoint, non-dimensional speed, and heading are kept fixed at $(0, 0)$, 1, and 0°, respectively, while the destination is randomly selected for each episode. The radial distance of the destination waypoint from the initial waypoint is uniformly sampled between $8L$ to $28L$, and the direction is uniformly sampled between 0 and $2\pi$. A scatter plot of 1000 randomly sampled goal coordinates is shown in Fig. 4.4.

### 4.1.5 Episode termination

During the training process, the objective of the agent is to maximize the cumulative reward, which drives the ship towards the designated destination. To ensure successful completion of an episode, a tolerance level of 0.5L around the target waypoint is established as a positive termination condition. Once the ship enters this region, the episode is considered successful, and a terminal reward of +100 is granted. However, an untrained agent may not be able to reach the destination point, so specific criteria must be established to determine if the agent is still capable of reaching the goal or if the episode should be terminated to avoid aimless paths. The negative termination condition

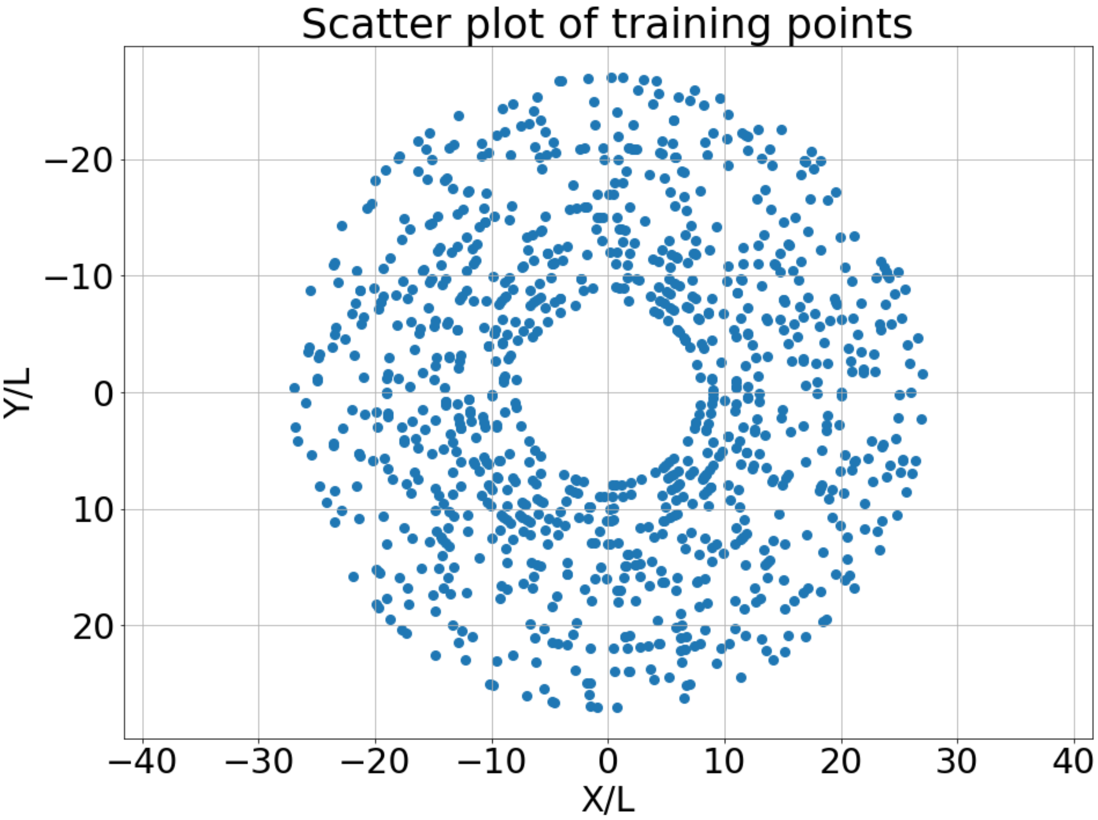


Figure 4.4: Scatter plot of randomly generated destination waypoints for training

is defined as follows:

$$\vec{v_1} \cdot \vec{v_2} < 0 \text{ and } \vec{U} \cdot \vec{v_2} < 0$$

The first condition for termination only relies on the current position of the ship and is independent of its velocity. Fig. 4.5 illustrates that the angle between $\vec{v_1}$ and $\vec{v_2}$ exceeds $\pi/2$ as the dot product becomes negative, and the ship crosses the line that is perpendicular to the line connecting the initial and final waypoints. The second condition considers the direction of the ship's velocity vector. The episode is not concluded even if the ship has passed the destination point, provided that its velocity still points towards the destination point.

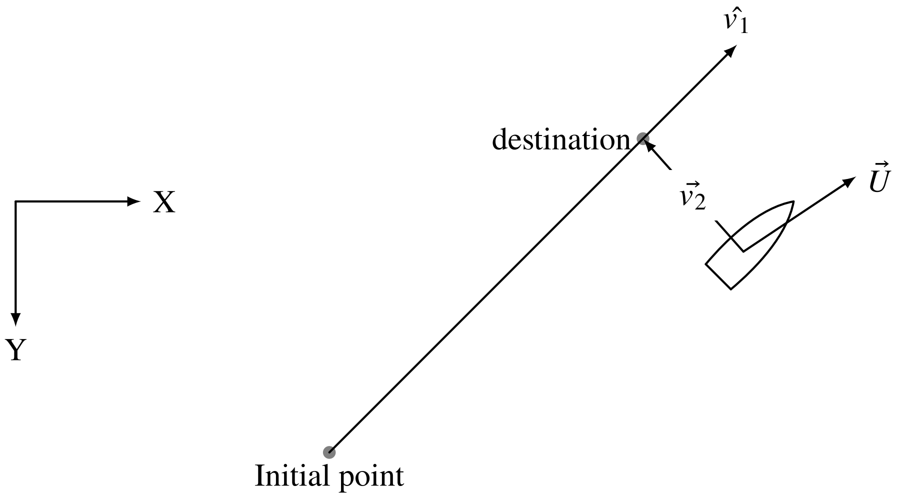


Figure 4.5: Visualizing the termination condition

## 4.2 STATIC OBSTACLE AVOIDANCE

In this section, DRL based controller is trained for static obstacle avoidance. Fig. 4.6 shows the schematic representation of the problem statement for static obstacle avoidance.

### 4.2.1 Observation States

The observation state space formulated for this problem contains 7 variables, including the 4 variables already defined in Sec. 4.1.1. Three new variables are defined for the obstacle avoidance case, *distance to an obstacle ($d_{obs}$)*, *angle to the obstacle ($\chi_{obs}$)* and size of obstacle($S_{obs}$).

#### 4.2.1.1 Distance to obstacle

It is defined as the distance to the obstacle from the vessel's current position.

$$d_{obs} = \sqrt{(x_{obs} - x)^2 + (y_{obs} - y)^2} \tag{4.9}$$

#### 4.2.1.2 Angle to obstacle

It is defined as the smallest signed angle of the angle between the velocity vector and the vector pointing from the ship to the position of the obstacle.

$$\chi_{obs} = ssa\left(arctan2(\vec{U}) - arctan2(\vec{v}_4)\right) \tag{4.10}$$

#### 4.2.1.3 Size of the obstacle

In this study, the radius of the obstacle is assumed to lie between 0 to 1L.

The extended observation space is given by: $[d_c, \chi_e, d_{wp}, r, d_{obs}, \chi_{obs}, S_{obs}]$

### 4.2.2 Action Space

The commanded rudder angle ($\delta_c$) and the actual rudder angle ($\delta$) vary smoothly as governed by (2.14) and (2.15).

### 4.2.3 Reward Structure

The reward structure is the same as described in Sec. 4.1.3

### 4.2.4 Training Process

In each episode, the ship begins at the origin (initial waypoint), oriented about the positive X-axis of the GCS ($\psi = 0$) and with an initial velocity of design speed in the surge direction. The ship has no initial acceleration in three degrees of freedom and no

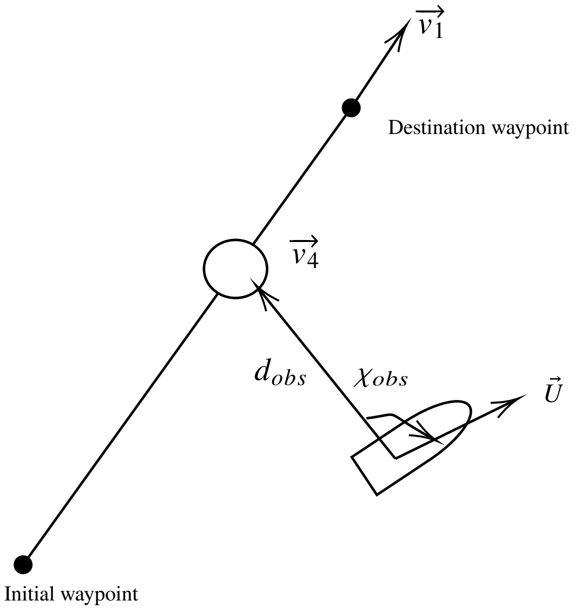
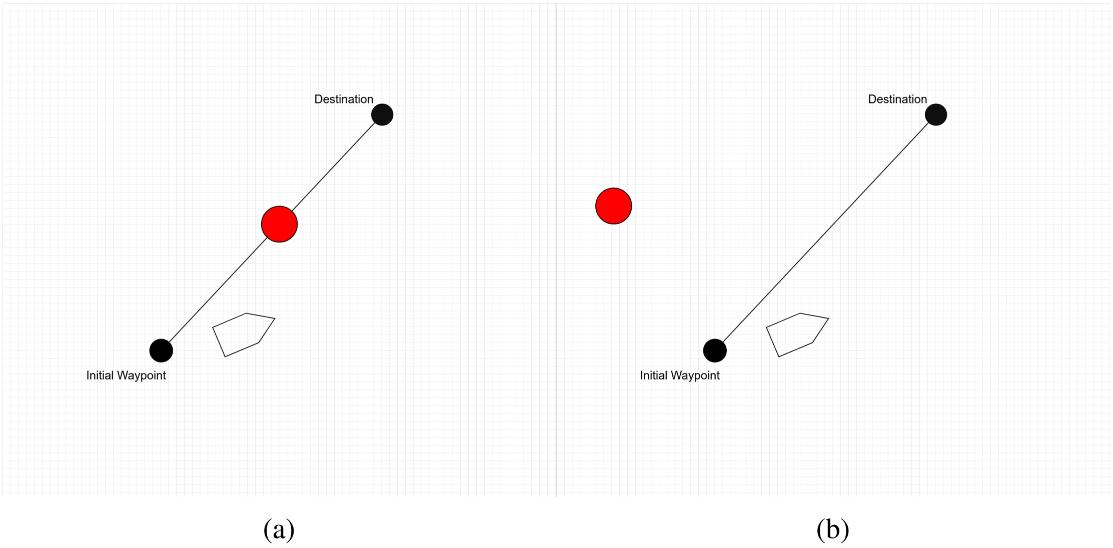


Figure 4.6: Depiction of the static obstacle avoidance problem

initial velocity in the sway and yaw motions.



(a) (b)

Figure 4.7: Depiction of training for static obstacle avoidance

The radial distance between the destination and the initial waypoint is sampled from a uniform distribution ranging from 8L to 18L. 60% of the training time, the obstacle is randomly placed within 0.25 to 0.75 times the destination waypoint from the initial waypoint along the line joining the initial and destination waypoint as shown in Fig. 4.7a.

In contrast, in the remaining 40% of the cases, the obstacle is placed randomly on a circle with a radius within 0.25 to 0.75 times the distance between the destination waypoint and the initial waypoint for the static obstacle (Fig. 4.7b). Note that in this case, the obstacle need not be on the line joining the waypoints and might not even be in the path of the vessel. This biased nature of the obstacle's position is included to let our agent observe the obstacles a sufficient number of times and know how to perform when the obstacle is not in the path.

### 4.2.5 Episode Termination

The termination conditions for the training remain unchanged as described in section 4.1.5. An additional termination condition is implemented, which is when the agent collides with an obstacle. In such cases, a penalty of $-100$ is assigned to the agent. The condition necessary for the agent to learn to avoid obstacles is provided in (4.11).

$$R_1 + \text{destination reward} > R_2 + \text{collision reward} \tag{4.11}$$

where $R_1$ is the total reward accumulated by the agent till it reaches to the destination while $R_2$ is the total reward accumulated by the agent till it collides with the obstacle as given in (4.8).

Although the collision penalty can be set to a high negative value, it has been observed during the training process that the ship takes longer routes and experiences a higher cross-track error in order to avoid obstacles. Furthermore, the sum of rewards ($r_1$, $r_2$ and $r_3$) is negative, indicating that the agent seeks to end the episode quickly in order to maximize its reward. To satisfy the constraint outlined in (4.11), the reward for reaching the destination is reduced to +20. (4.12) illustrates the condition for collision.

$$d_{obs} - S_{obs} \leq 0.5L \tag{4.12}$$

## 4.3 DYNAMIC OBSTACLE AVOIDANCE

In this section, DRL based controller is trained for dynamic obstacle avoidance.

### 4.3.1 Measure of Collision Risk

To ensure safe autonomous navigation in the presence of multiple obstacles, it is important to quantitatively assess Collision Risk (CR) and determine the appropriate avoidance point. The Closest Point Approach (CPA) method is commonly used to evaluate CR by determining the closest point between the ship and obstacle while maintaining their current speed and direction. The distance between the ship and obstacle at the CPA is known as the Distance to the CPA (DCPA), while the time taken by the obstacle to reach the CPA is known as the Time to the CPA (TCPA), as illustrated in Fig. 4.8. In essence, the DCPA indicates the severity of a potential collision, while the TCPA indicates the urgency of the situation.

In previous studies, quantitative assessment of collision risk (CR) has been proposed by utilizing a combination of the closest point of approach (CPA) and time to CPA (TCPA), as presented by Mou *et al.* (2010) and Zhen *et al.* (2017). In this research, CPA was utilized for assessing the ship and obstacle's CR in a quantitative manner. The figures illustrating the concepts of CPA, TCPA, and distance to CPA (DCPA) can be found in Fig. 4.8, and the calculation of TCPA and DCPA can be performed by utilizing 4.13.

$$\begin{aligned} DCPA &= R\sin(\chi_R - \chi_{os} - \theta_T - \pi) \\ TCPA &= \frac{R}{V_R}\cos(\chi_R - \chi_{os} - \theta_T - \pi) \end{aligned} \tag{4.13}$$

where, $R$ represents the distance between a ship and an obstacle, while $V_R$ and $\chi_R$ denote the relative speed and course angle between them. Additionally, $\chi_{OS}$ represents the course of the ship, $\chi_{TS}$ represents the course of the obstacle while $\theta_T$ is the bearing of the obstacle relative to the ship.

Various studies have proposed different methods for the quantitative evaluation of collision risk using TCPA and DCPA. For instance, a relatively simple assessment formula based

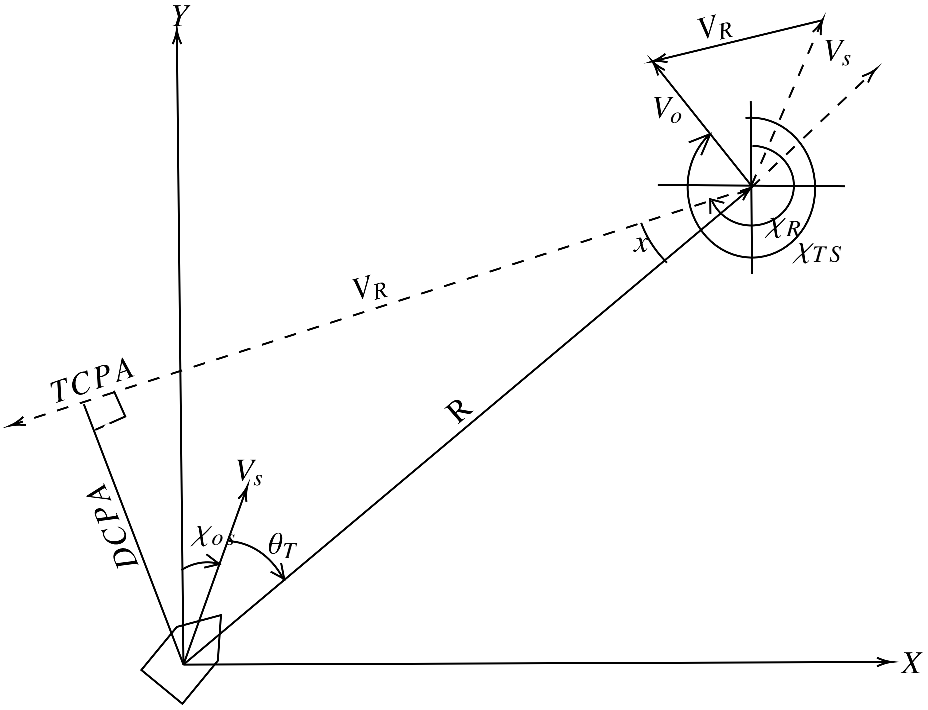


Figure 4.8: Calculation of DCPA and TCPA

on the exponential functions of TCPA and DCPA was proposed by Mou *et al.* (2010). In this study, we have modified the formula proposed by Mou *et al.* (2010) by introducing two weights, $\alpha$ and $\beta$, that are related to DCPA and TCPA.

$$CR = \begin{cases} \exp(-\alpha * DCPA - \beta * TCPA) & \text{if TCPA} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

In this study, it involves the evaluation of CR between the ship and the target obstacle. To quantify the CR, 4.14 is employed. This equation considers the DCPA and TCPA, where a smaller value of these parameters indicates a more hazardous situation. The CR is equal to 0 if either the DCPA or TCPA is infinite, indicating the absence of any target obstacles in the proximity or a negative TCPA. A negative TCPA suggests that either the obstacle and the ship have already crossed each other or they are moving away from each

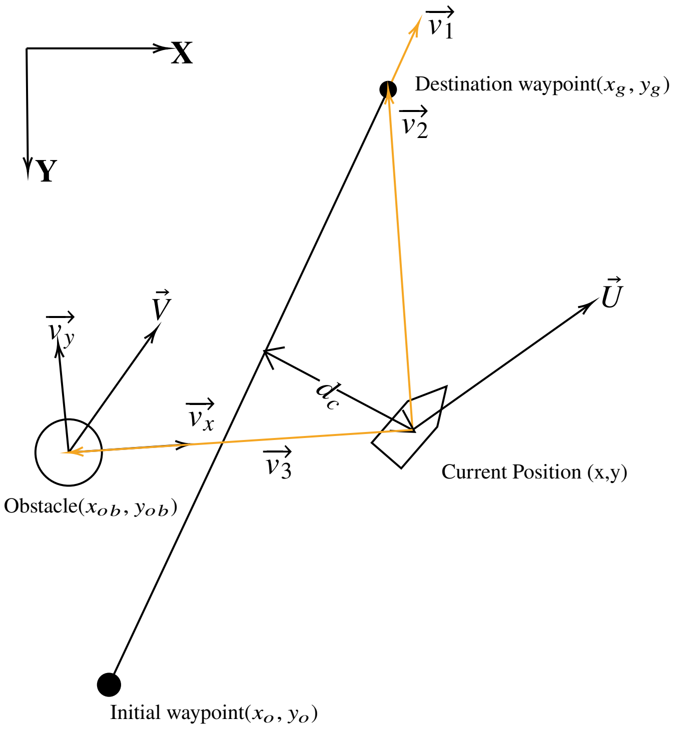


Figure 4.9: Depiction of the dynamic obstacle avoidance problem

other without any chance of collision.

### 4.3.2 Observation Space

The observation state space formulated for this problem contains 9 variables, including the 7 variables already defined in Sec. 4.2.1. Additional observation states are the relative velocity of the obstacle with respect to the ship in the ship direction and the velocity of the obstacle perpendicular to the line joining the obstacle and ship as shown in Fig. 4.9

#### 4.3.2.1 Velocity of obstacle towards the ship($\vec{v_x}$)

This parameter gives an idea of how fast the obstacle is approaching the ship.

#### 4.3.2.2 Velocity of obstacle perpendicular to ship($\vec{v_y}$)

This parameter gives the idea of whether the obstacle is on the starboard or port side of the obstacle. It will be positive when the obstacle is on the port side of the ship while it will be negative when on the starboard side.

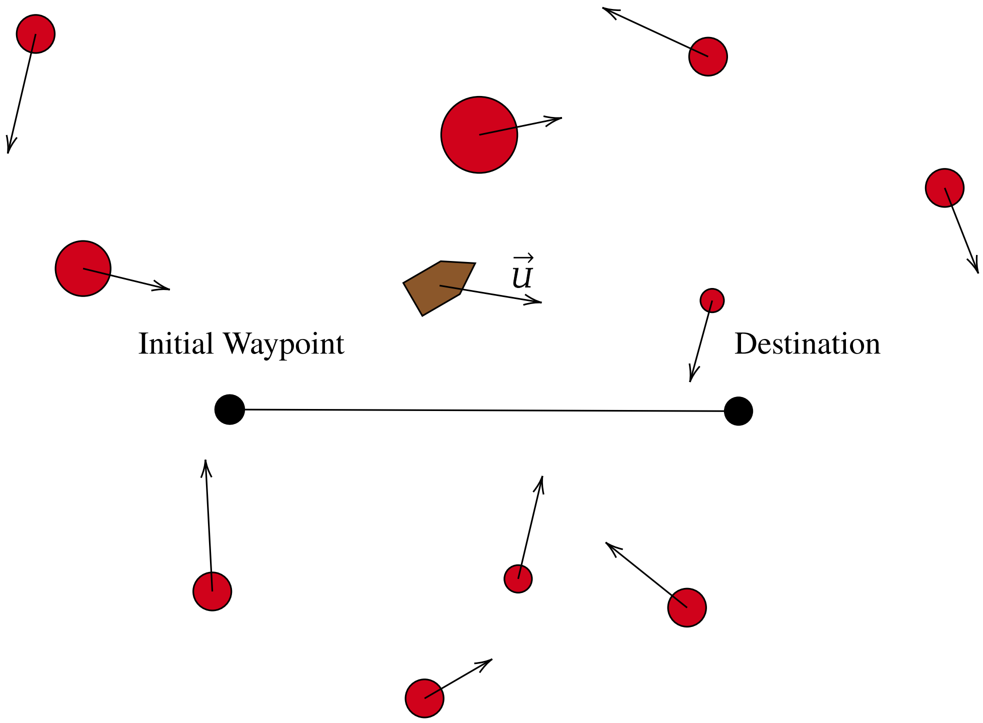


Figure 4.10: Depiction of training for dynamic obstacle avoidance

### 4.3.3 Action Space

This action space is the same as described in Sec. 4.2

### 4.3.4 Reward Structure

The reward structure is the same as described in Sec. 4.1.3

### 4.3.5 Training Process

In each episode, the ship begins at the origin (initial waypoint), oriented about the positive X-axis of the GCS ($\psi = 0$) and with an initial velocity of design speed in the surge direction. The ship has no initial acceleration in three degrees of freedom and no initial velocity in the sway and yaw motions.

In each training episode, the destination is chosen at random, whereas the initial waypoint, velocity, and heading are fixed at $(0, 0)$, 1 and 0°, respectively. The radial distance between the destination and the initial waypoint is sampled from a uniform distribution ranging from $8L$ to $18L$. At starting of the episode, the obstacle is placed at a radial distance of $5L$ to $20L$. The velocity of the obstacle lies between 0 to $0.5L$ per timestep, and the radius of an obstacle is kept between 0 to $1L$.

At the start of each episode, obstacles with random sizes, non-dimensional speed and locations get spawned in the environment. The ship calculates the most critical obstacle judged by calculating Collision Risk (CR) as shown in 4.14. The obstacle with the maximum collision risk is chosen as the critical obstacle. The observation state variable related to the obstacle is calculated for the critical obstacle. At each timestep, the CR is calculated and according to that, the input is fed into the neural network until a termination condition is reached.

### 4.3.6 Episode Termination

The termination criterion remains unchanged as specified in Sec. 4.2.5. Additionally, the penalty for colliding with an obstacle has been reduced to $-200$. This decision was made because, with the introduction of a dynamic obstacle, the ship's trajectory is less smooth due to the presence of multiple obstacles, resulting in a potentially more negative reward compared to the static case where there was only one obstacle, leading to a less negative overall reward.

# CHAPTER 5

# RESULTS AND DISCUSSIONS

This section consists of four components. The first part compares DQN (3-actions) controllers in siulations and experiments alngwith comparison with traditional PD controller with ILOS guidance system through simulations. In the second part, simulations are used to compare the waypoint tracking and path-following abilities of the three DRL-based controllers: DQN(5-actions), PPO, and DDPG. The third part assesses the capability of the DQN and DDPG controllers to handle static obstacles and further uses hybrid architecture for path following and obstacle avoidance. Finally, in the fourth part, the DQN and DDPG controllers are evaluated for their ability to handle dynamic obstacles.

## 5.1 COMPARISON OF MODERN AND TRADITIONAL CONTROLLERS

In this section, the performance of DQN controllers with a 3-action state is compared to that of a traditional PD controller through both simulation and experimental methods. The DQN algorithm, is limited to discrete action values. In this part, we choose three action values DQN ($\delta_c \in -35°, 0°, 35°$) for comparison with PD controller and in experiments.

### 5.1.1 Hyperparameters of the network

After tuning the hyperparameters, the agent is tested on various waypoint tracking maneuvers to ensure satisfactory performance. An exponentially decaying function is selected as the learning rate, and the hyperparameters for the DQN model are provided in Table 5.1. The average training losses and episode returns over 100 episodes are depicted in Fig. 5.1. To achieve a balance between under-fitting and over-fitting, the policy at 8000 episodes is chosen and will be evaluated in the subsequent sections for path following using DQN.

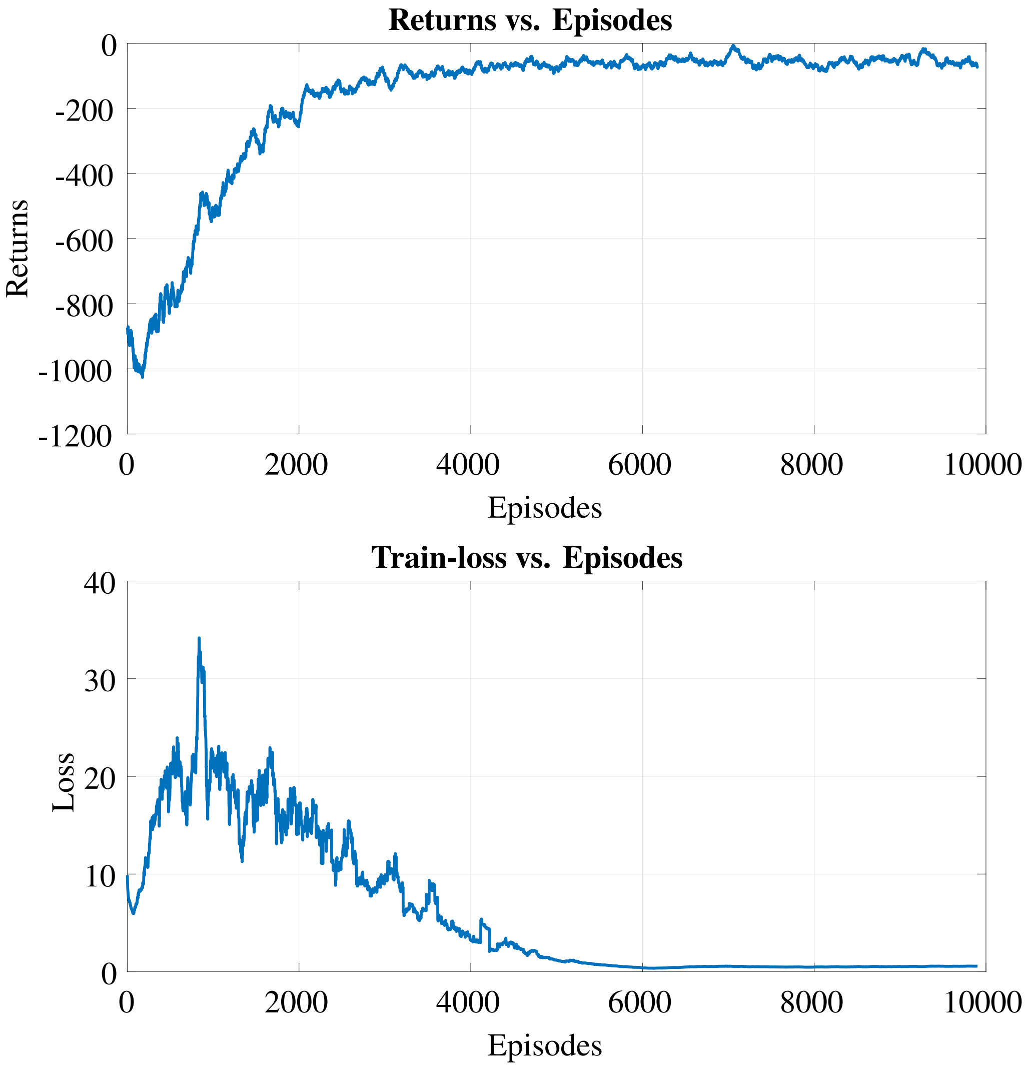


Figure 5.1: Training returns and loss for the DQN model

Table 5.1: Hyperparameters for DQN model

| **Hyperparameter** | **Value** |
|---|---|
| Initial Learning rate | 0.001 |
| Decay Steps | 5000 |
| Decay Rate | 0.5 |
| Hidden layers | (64,64) |
| Discount factor($\gamma$) | 0.95 |
| Sample batch size | 128 |
| Replay buffer size | 100000 |
| Activation function | tanh |
| Maximum time steps | 160 |
| Time step interval Δt | 0.3 |
| Number of episodes | 8000 |
| Update frequency(time steps) | 20 |
| Target network update frequency(time steps) | 1 |
| Target update rate ($\tau$) | 0.01 |

It is also important to note that this study has used TensorFlow version 2.9.1 which allows the GPU operations to be made deterministic. This means that the results produced in the study can be reproduced on any computer by training the agent with the same random seed value (TensorFlow).

### 5.1.2 Calm water results

#### 5.1.2.1 Waypoint tracking

The study evaluates the DQN agent's performance by analyzing its ability to track waypoints and follow a path in various scenarios. The initial state of the vessel is set with a unit non-dimensional velocity in the surge direction and oriented along the global x-axis. To investigate the agent's ability to track points in all quadrants, the study specifies waypoints $(10L, 10L)$, $(10L, -10L)$, $(-10L, -10L)$ and $(-10L, 10L)$ for each case using the same starting state as mentioned above. Fig. 5.2 illustrates that the trained agent successfully reaches the destination points in all four quadrants, and the path it takes is smooth. Furthermore, the agent attempts to reduce both the cross-track error and the course angle error simultaneously. The trajectory's root mean square cross-track error is defined as follows:

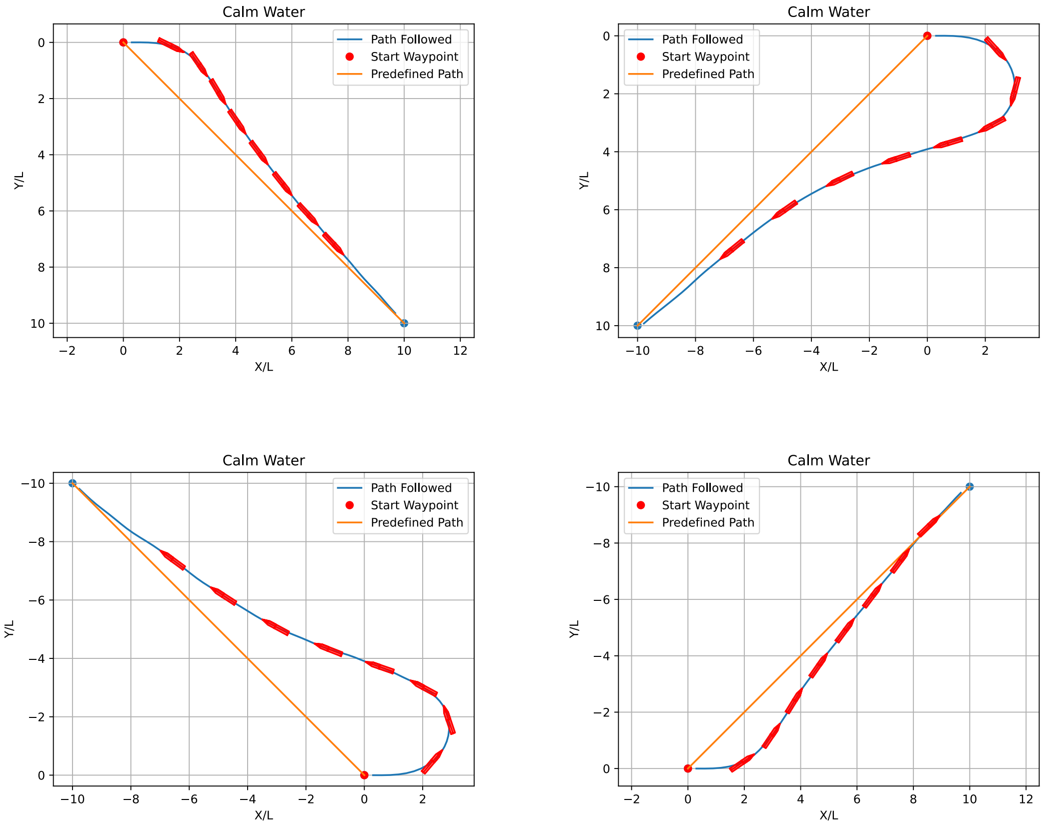
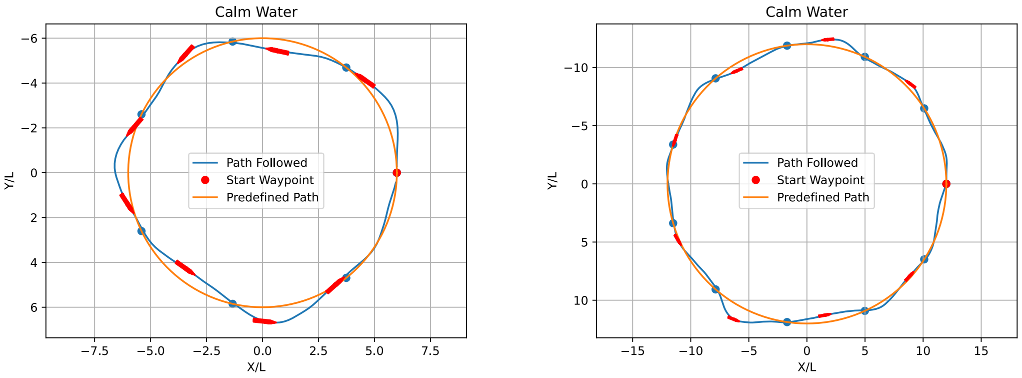


Figure 5.2: Single waypoint tracking for DQN

$$d_{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} d_c^2(n\Delta t)} \tag{5.1}$$

where $N$ is the number of time steps in the trajectory and $d_c(n\Delta t)$ denotes the value of the cross track error at the $n^{th}$ time step. It is seen from Fig. 5.3 that the $d_{RMSE}$ for the larger circle is less than the corresponding value for the smaller circle.

The results of the agent's ability to track an elliptical path are illustrated in Fig. 5.4. The elliptical path has major and minor axes of lengths $28L$ and $24L$ respectively, and is discretized into 15 waypoints. The ship commences at $(x, y) = (14L, 0)$ with an initial orientation along the negative Y-axis ($\psi = -\pi/2$). The distance between the waypoints is approximately $5L$, and the local radius of curvature along the path ranges from about



(a) Eight ($d_{RMSE}$ = 0.6095$L$) (b) Eight ($d_{RMSE}$ = 0.5395$L$)

Figure 5.3: Circle maneuver waypoint tracking

10$L$ to 16$L$. The agent's ability to track the elliptical path is successful with a root mean square error of $d_{RMSE} = 0.2969L$.

In order to examine the influence of propeller rotation on asymmetry in the port and starboard turns, a path in the shape of an "eight" was chosen and discretized into 23 waypoints. The ship initiates the maneuver with a heading of $\psi$=0 from the origin and completes the lower circle of radius 9$L$ before finishing the upper circle of the same radius to successfully complete the eight maneuver, as depicted in Fig. 5.5. The results indicate that the tracking performance is not impacted by the propeller rotation asymmetry as the turns in both directions are tracked with equal accuracy.

### 5.1.3 Experiments

To verify the accuracy of the simulation results through experimental testing, a scaled free-running model of the KCS ship is utilized with the guidance of the RL agent's policy. As illustrated in Fig. 5.6, the scaled ship model is equipped with a brushless DC motor to control the propeller and a stepper motor to control the rudder, while the propeller RPM remains fixed and the RL agent policy outputs the commanded rudder angle. Furthermore, the vehicle is fitted with an Ardusimple simpleRTK2B-SBC GPS system and an SBG Ellipse-A IMU, which are shown in Fig. 5.7, and their measurements

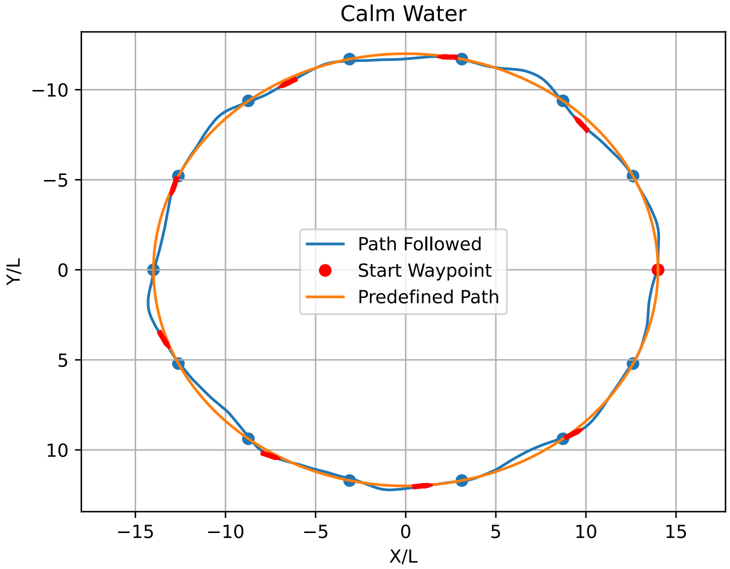


Figure 5.4: Ellipse maneuver waypoint tracking ($d_{RMSE} = 0.2969L$)

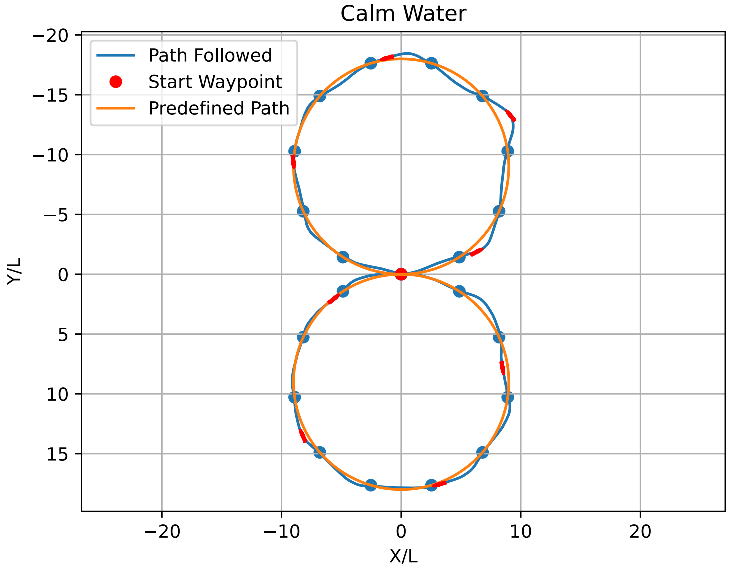


Figure 5.5: Eight ('8') maneuver waypoint tracking ($d_{RSME} = 0.4557L$)

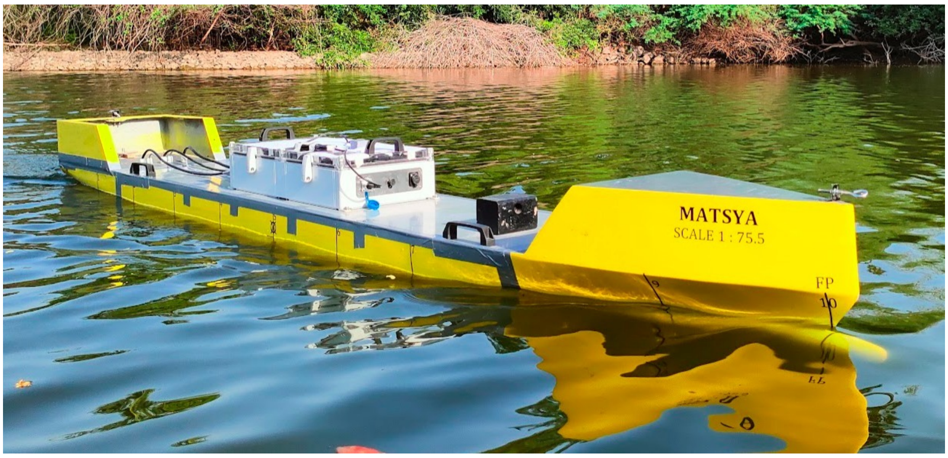


Figure 5.6: 1:75.5 scaled model of KCS

Table 5.2: Sensors Onboard

| Sensor | ROS Standard Message | Frequency (Hz) |
|---|---|---|
| SBG Ellipse-A (IMU) | sensor_msgs/Imu | 100 |
| simpleRTK2B SBC | sensor_msgs/NavSatFix | 10 |

are combined through a Kalman filter. The experiments are conducted on IIT Madras Lake, which measures 300 $m$ in span and 100 $m$ in width.

Because of the limited space available in the lake, it was not possible to execute the same maneuvers as detailed in the previous section. Consequently, simulations and experiments were conducted using a predetermined set of waypoints that form a 40 $m$ square in the middle of the lake. The GPS coordinates of the waypoints are presented in Table 5.3. It is worth noting that the first and last waypoints are identical and serve as the reference point.

In Fig. 5.8, the white lines represent the straight lines connecting the waypoints defined in Table 5.3, while the red line illustrates the actual trajectory of the vehicle throughout the experiment. The comparison between the experimental and simulated results is

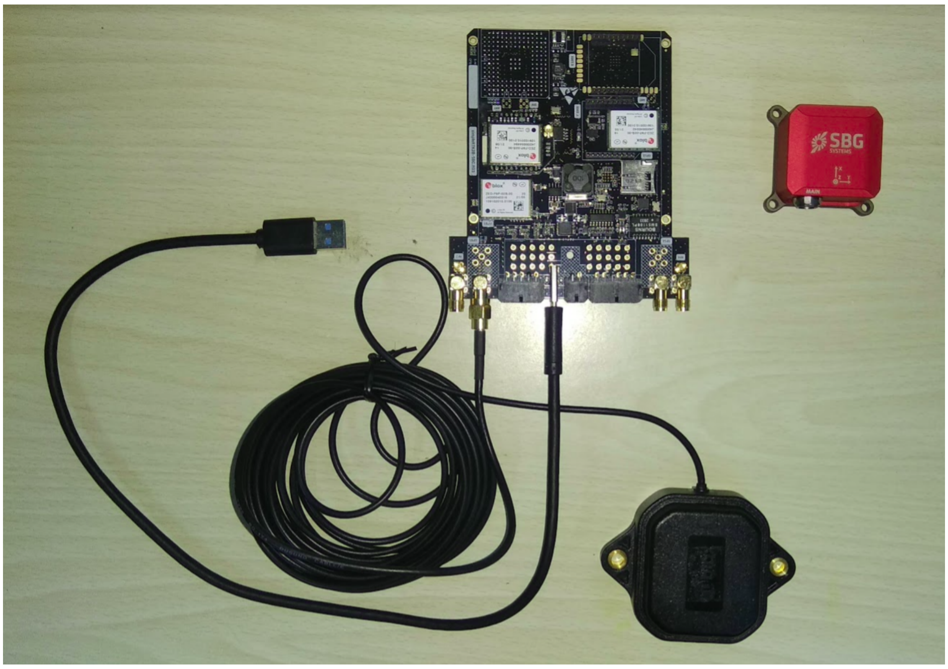


Figure 5.7: Sensors used: simpleRTK2B SBC(left) and SBG Ellipse-A (right)

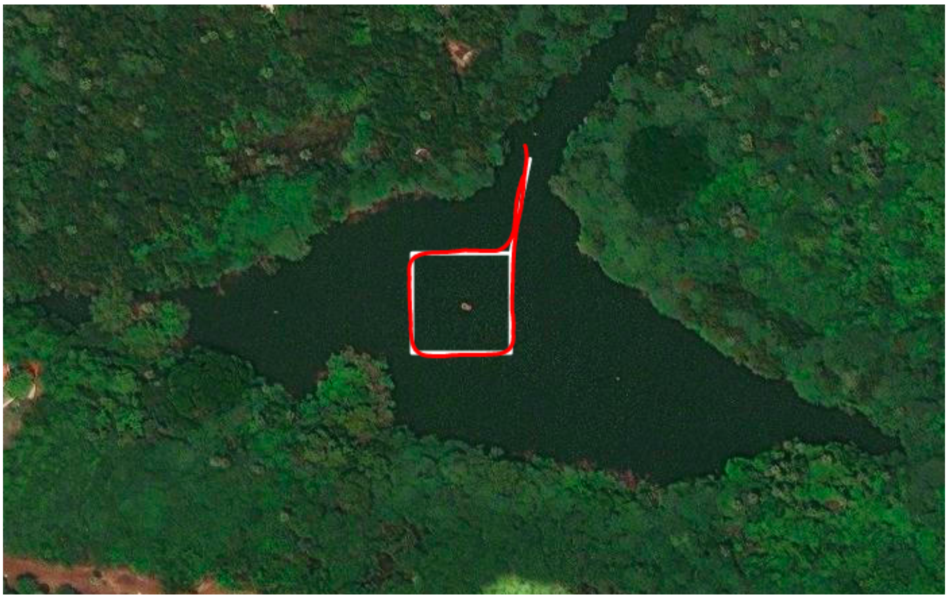


Figure 5.8: Waypoints and the Trajectory: white represents the square generated by the waypoints, red represents the trajectory traversed by the vessel.

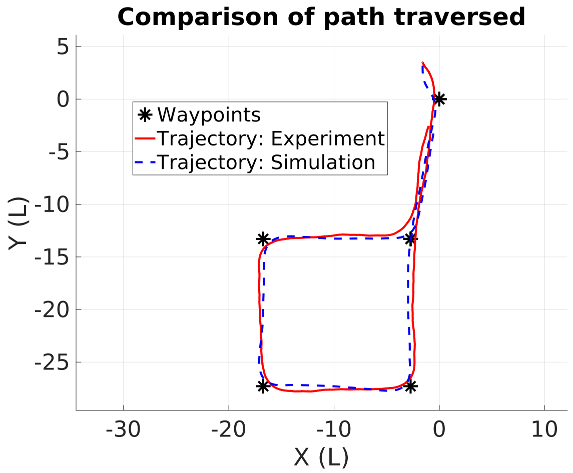


Figure 5.9: Comparison of the path traversed by the model in experiment and simulation (Simulation $d_{RMSE} = 0.6286L$ Experimental $d_{RMSE} = 0.8257L$)

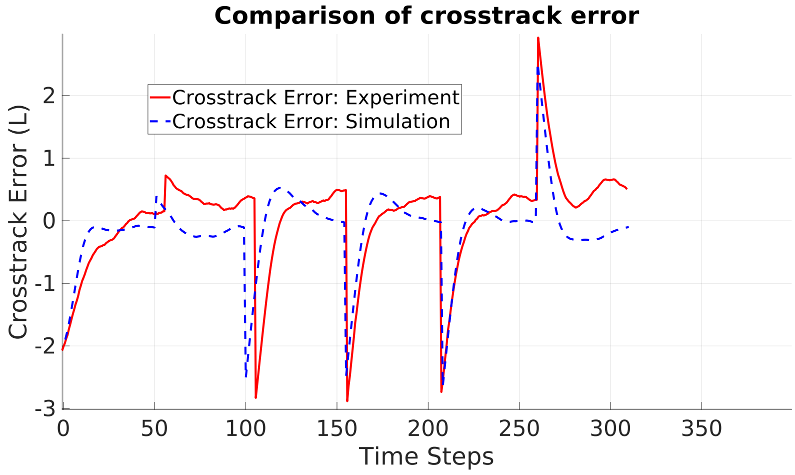


Figure 5.10: Comparison of cross-track error between experiment and simulation.

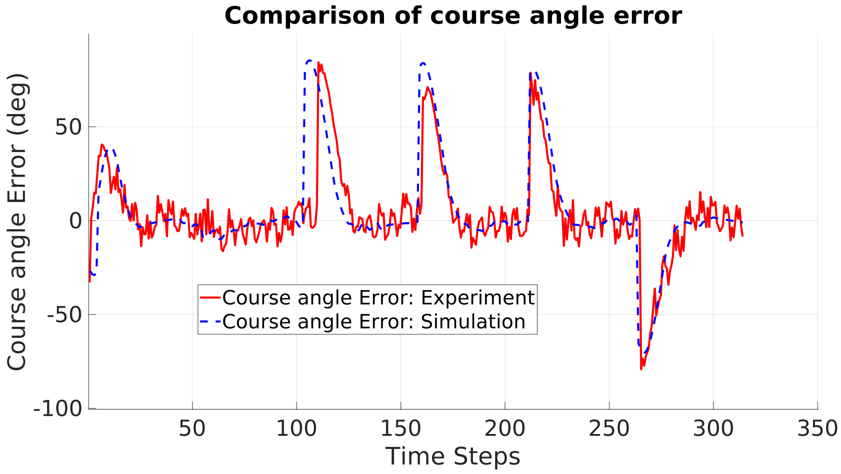


Figure 5.11: Comparison of course-angle error between experiment and simulation.

presented in Fig. 5.9, while the variations in the cross-track error and course angle error are depicted in Fig. 5.10 and Fig. 5.11, respectively. The primary aim of the model was to follow a setpoint of $d_c = 0$ for each edge of the square until the vertex waypoint was reached. The circle of acceptance was set at $3L$ for both experiments and simulations during waypoint tracking. The course angle error values from the simulations correspond well with the corresponding experimental values, and the time of switching waypoints also agrees between the two. The cross-track error curves are slightly different, where the simulations exhibit a higher initial overshoot than the experiments. This can be attributed to minor variations between the simulated and true vessel dynamics and the presence of mild gust winds during the experiments. Overall, the simulated and experimental trajectories match well, validating the model and the DQN-based controller for the craft.

## 5.2 PERFORMANCE WITH WIND FORCES

In a realistic ocean environment, vessels will face disturbances caused by wind, waves, and currents. However, in this study, we will focus on the impact of wind disturbances on the vessel to assess the DQN controller's ability to reject disturbances.

Table 5.3: Waypoints

| Waypoints | Latitude (deg) | Longitude (deg) |
|---|---|---|
| WP1 | 12.994224 | 80.239620 |
| WP2 | 12.993859 | 80.239544 |
| WP3 | 12.993474 | 80.239544 |
| WP4 | 12.993474 | 80.239151 |
| WP5 | 12.993859 | 80.239151 |
| WP6 | 12.993859 | 80.239544 |
| WP7 | 12.994224 | 80.239620 |

### 5.2.1 Modelling wind forces

Until now, the path-following ability of autonomous ships was examined using the DQN agent without considering external environmental forces. This section, however, focuses on the agent's ability to follow a desired path under varying wind conditions, ranging from moderate to severe. As the observation states do not take into account wind parameters, the agent cannot anticipate the effects of wind (feedforward control) and can only react to deviations from the desired path once they become significant (feedback control).

To incorporate the effects of wind forces and moments, the right-hand side of (2.5) is modified by including wind models. The wind velocity, which is non-dimensional, is denoted as $V_w$, and its direction is represented by $\beta_w$, which is defined as the angle between the wind direction and the positive X-axis of the GCS. The non-dimensional wind velocity components with respect to the BCS of the ship are then defined according to (5.2).

$$\begin{aligned} u_w &= V_w \cos(\beta_w - \psi) \\ v_w &= V_w \sin(\beta_w - \psi) \end{aligned} \tag{5.2}$$

The non-dimensional velocity components of the ship relative to the wind can be defined as $u_{rw} = u - u_w$ and $v_{rw} = v - v_w$. The magnitude of non-dimensional relative wind

Table 5.4: Wind force parameters

| **Parameter** | **Value** |
|---|---|
| $A_x$ | 0.1064 |
| $A_y$ | 0.7601 |
| $C_{wx}$ | $cos(\gamma_w)$ |
| $C_{wy}$ | $sin(\gamma_w)$ |
| $C_{w\psi}$ | $0.5sin(\gamma_w)$ |
| $\rho_a$ | $1.225kg/m^3$ |
| $\rho$ | $1025kg/m^3$ |

velocity $U_{wr}$ and relative wind angle with respect to the vessel $\gamma_w$ are given by (5.3).

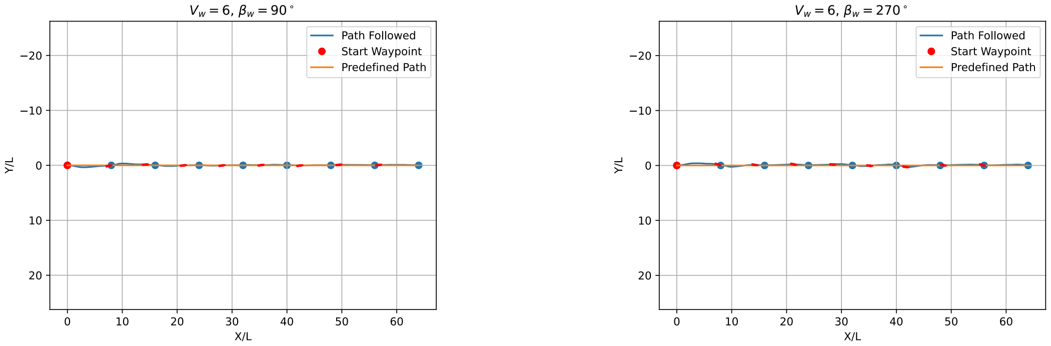
$$\begin{aligned} U_{wr} &= \sqrt{u_{rw}^2 + v_{rw}^2} \\ \gamma_w &= \arctan 2(-v_{rw}, -u_{rw}) \end{aligned} \tag{5.3}$$

The non-dimensional wind force components $W_x$, $W_y$ and the non-dimensional wind yaw moment $W_\psi$ can then be calculated as shown in (5.4)

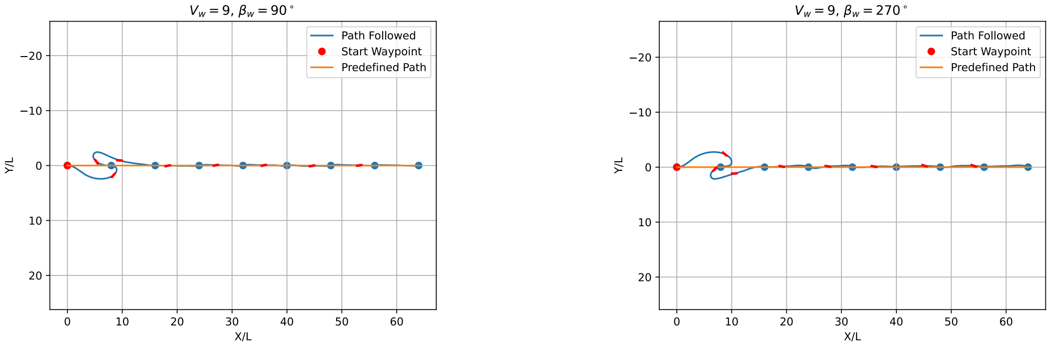
$$\begin{aligned} W_x &= C_{wx}(\gamma_w)\frac{\rho_a}{\rho}A_x U_{wr}^2 \\ W_y &= C_{wy}(\gamma_w)\frac{\rho_a}{\rho}A_y U_{wr}^2 \\ W_\psi &= C_{w\psi}(\gamma_w)\frac{\rho_a}{\rho}A_y L_{OA} U_{wr}^2 \end{aligned} \tag{5.4}$$

The non-dimensional lateral and longitudinal projected areas of the hull above water in the yz and xz planes in the BCS are denoted as $A_x$ and $A_y$, respectively. It should be noted that the non-dimensional factor for the projected areas $A_x$ and $A_y$ is taken as $Ld_{em}$. The wind coefficients $C_{wx}$, $C_{wy}$ and $C_{w\psi}$ are also non-dimensional and assumed to be functions of the relative wind direction $\gamma_w$. The density of air and water are represented as $\rho_a$ and $\rho$, respectively. The non-dimensional overall length of the vessel is denoted by $L_{OA}$, which is normalized by the length between perpendiculars $L$. The wind parameters used in the study are provided in Table 5.4.

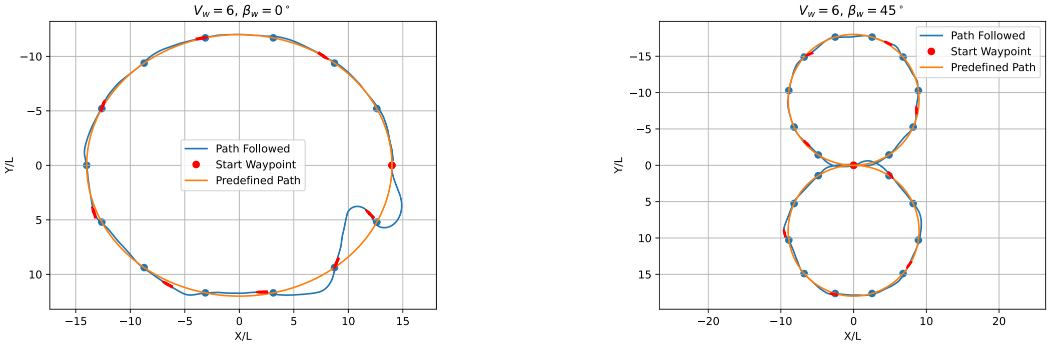
The wind speed and the direction are varied to simulate different scenarios. The



(a) Straight Line Path ($d_{RMSE} = 0.1227L$) (b) Straight Line Path ($d_{RMSE} = 0.1615L$)



(c) Straight Line Path ($d_{RMSE} = 0.9110L$) (d) Straight Line Path ($d_{RMSE} = 0.9938L$)



(e) Ellipse ($d_{RMSE} = 0.8456L$) (f) Eight ($d_{RMSE} = 0.3941L$)

Figure 5.12: Different cases for path following in presence of constant and uniform wind for DQN(3-actions) controller

Table 5.5: Wind force cases

| **Case** | **Path** | $v_w$ | $\beta_w$ | $d_{RMSE}$ |
|---|---|---|---|---|
| 1 | Line | 6 | $\pi/2$ | $0.1227L$ |
| 2 | Line | 6 | $-\pi/2$ | $0.1615L$ |
| 3 | Line | 9 | $\pi/2$ | $0.9110L$ |
| 4 | Line | 9 | $-\pi/2$ | $0.9938L$ |
| 5 | Ellipse maneuver | 6 | 0 | $0.8456L$ |
| 6 | Eight maneuver | 6 | $\pi/4$ | $0.3941L$ |

performance of the controller in the presence of constant wind is evaluated.

### 5.2.2 Results

The research focuses on examining the effect of wind speeds and directions on a vessel's path-tracking performance. Specifically, six different cases are considered, as described in Table 5.5. The first four cases involve straight-line paths, while the remaining two involve an ellipse and eight maneuvers. Table 5.5 reports the non-dimensional wind speed and direction for each case, along with the corresponding $d_{RMSE}$ values observed. Fig. 5.12 illustrates the path tracked for each case listed in Table 5.5. The outcomes indicate that in moderate winds ($V_w = 6U$), the vessel experiences minor initial deviations while tracking the straight-line paths but manages to recover later. Stronger winds lead to more significant initial deviations and substantially increase the $d_{RMSE}$ compared to moderate winds. For cases 5 and 6, the model can track the desired path successfully even in the presence of strong winds. Although the beam wind leads to significant initial deviation in the ellipse maneuver, the controller can eventually navigate through all the waypoints and complete the intended path. These simulations demonstrate the controller's ability to effectively reject disturbances.

### 5.2.3 Comparison with a PD-based controller

This section compares the path-following ability of the DQN agent with that of a PD controller. The study employs a Proportional Derivative (PD) controller to ensure that the vessel's heading angle $\psi$ converges to the desired heading angle $\psi_d$, which is determined

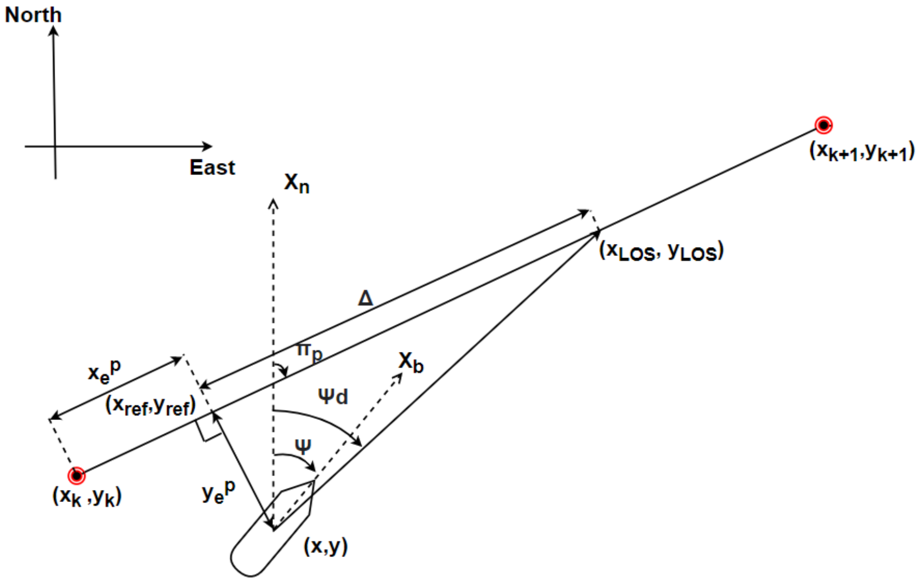


Figure 5.13: ILOS Guidance

using the integral line of sight (ILOS) guidance law as described in (Fossen, 2021). The error $e$ is the difference between the current heading angle $\psi$ and the reference value $\psi_d$ and is defined in (5.5). The derivative of the error with respect to time is expressed in (5.6).

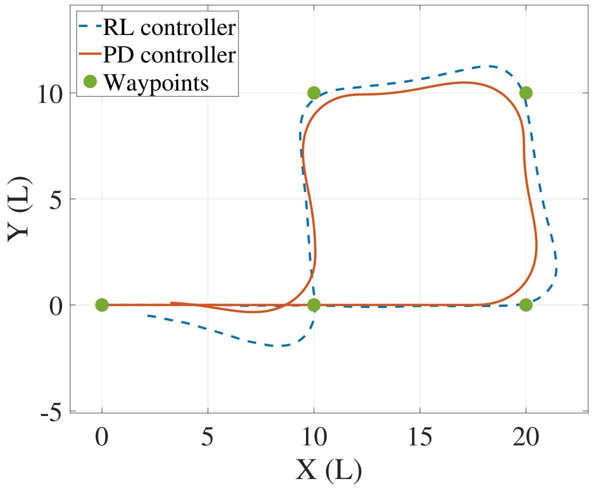
$$e = \psi_d - \psi \tag{5.5}$$

$$\dot{e} = \dot{\psi}_d - \dot{\psi} = -r \tag{5.6}$$

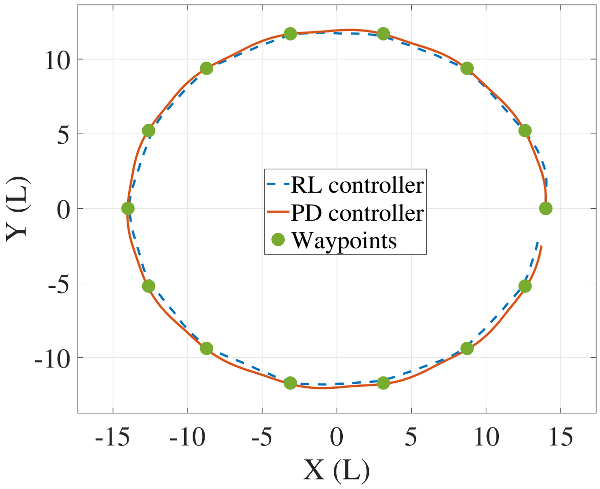
where $\dot{\psi}_d$ is taken to be 0 for the control implementation. This is a fair approximation as the rate of change of desired heading angle $\dot{\psi}_d$ is governed by the outer guidance loop and varies slowly. Thus the proportional derivative (PD) control law can be expressed as

$$\delta_c = K_p e + K_d \dot{e} = K_p(\psi_d - \psi) - K_d r \tag{5.7}$$

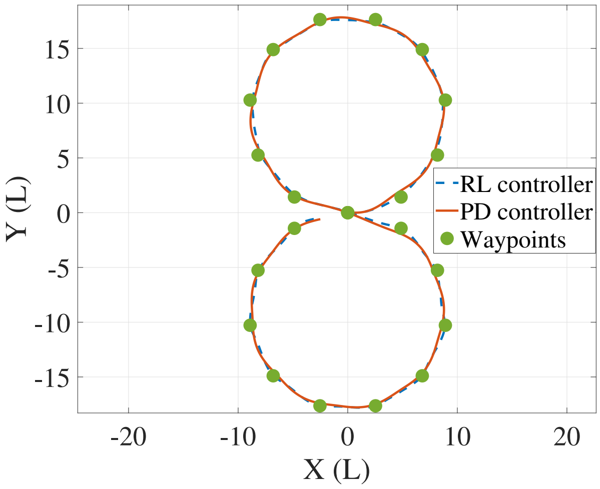
.



(a) Square Trajectory



(b) Ellipse Trajectory



(c) Eight Trajectory

Figure 5.14: Comparison of the path traversed by the model in PD controller and RL controller in simulation

where $\delta_c$ is the commanded rudder angle, $K_p$ is the proportional gain and $K_d$ is the derivative gain of the controller. These controller gains are tuned in such a way that deviation of the vessel's trajectory from the desired square trajectory as shown in Fig. 5.14a, is minimal. $K_d = 4.0$ and $K_p = 2.0$ were found to provide the best performance among the values investigated in the tuning process. The study tested the controllers against two other trajectories, an eight and an ellipse. The RL controller outperformed the PD agent by 54.56% for the eight maneuver, and by 29.68% for the ellipse trajectory. The plots are shown in Fig. 5.14 and the RMSE of cross track is listed in Table 5.6. Thus it can be seen that the traditional controller performance parallels the RL controller on paths for which the traditional controller gains are tuned. However, the same traditional controller is unable to perform as well as the RL controller on different paths for which its gains were not tuned.

Table 5.6: Comparison of the DQN and Traditional controller

| **Trajectory** | **DQN** | **PD** |
|---|---|---|
| Square | 0.8428L | **0.6859L** |
| Ellipse | **0.2537L** | 0.3290L |
| Eight | **0.3234L** | 0.5322L |

## 5.3 COMPARISON OF DRL-BASED CONTROLLERS

In this section, three different Deep Reinforcement Learning based controllers are compared in simulations. The RL-based controllers are based on DQN, PPO and DDPG algorithms. The DQN algorithm action space is extended to five i.e. ($\delta_c \in -35°, -20°, 0°, 20°, 35°$) for effectively comparing between the later two. The DDPG and PPO can operate with a continuous action space. The action space is taken to be continuous: $\delta_c \in [-35°, 35°]$. The relation between commanded rudder $\delta_c$ and the actual rudder angle $\delta$ is the same as defined in (2.15).

### 5.3.1 Waypoint Tacking

The study evaluates the performance of RL controllers by analyzing their ability to track waypoints and follow paths in various situations. The starting condition of the vessel was identical to the one discussed in Sec. 5.1.2.1. The DQN, DDPG, and PPO controllers were tested in all four quadrants, as depicted in Fig. 5.15, 5.16, and 5.17. The results show that the trained agent successfully reached the destination points in all four quadrants. Additionally, the path taken by the agent was found to be sufficiently smooth, and the agent simultaneously reduced the cross-track error and course angle error. The controller effort (C.E) is defined as the integral of abs(delta) over the simulation trajectory divided by the total time of the trajectory described in (5.8).

$$C.E = \frac{|\delta|}{N} \tag{5.8}$$

where N is the number of time steps in the trajectory and $\delta$ is the actual rudder angle.

### 5.3.2 Hyperparameters of the network

The agent is calibrated for satisfactory waypoint tracking by tuning the hyperparameters. The agent is then tested by simulating various maneuvers for waypoint tracking. The learning rate is chosen as an exponentially decaying function. The hyperparameters for the DQN, PPO and DDPG are shown in Table 5.7, 5.8 and 5.9. The returns vs episodes and loss vs episodes plots for DQN, PPO and DDPG are shown in Fig. 5.18, 5.19 and 5.20.

### 5.3.3 Path following through waypoint tracking

Since the RL agent is successful in being able to guide the ship to the destination waypoint, it can be directed to follow complex paths that are discretized into an adequate number of waypoints assuming that the chosen set of waypoints describes the path effectively.

To test the turning ability and path following ability of the agent, circles of radius $6L$ and $12L$, ellipse and eight are discretized into a number of points as done in Sec. 5.1.2.1. It

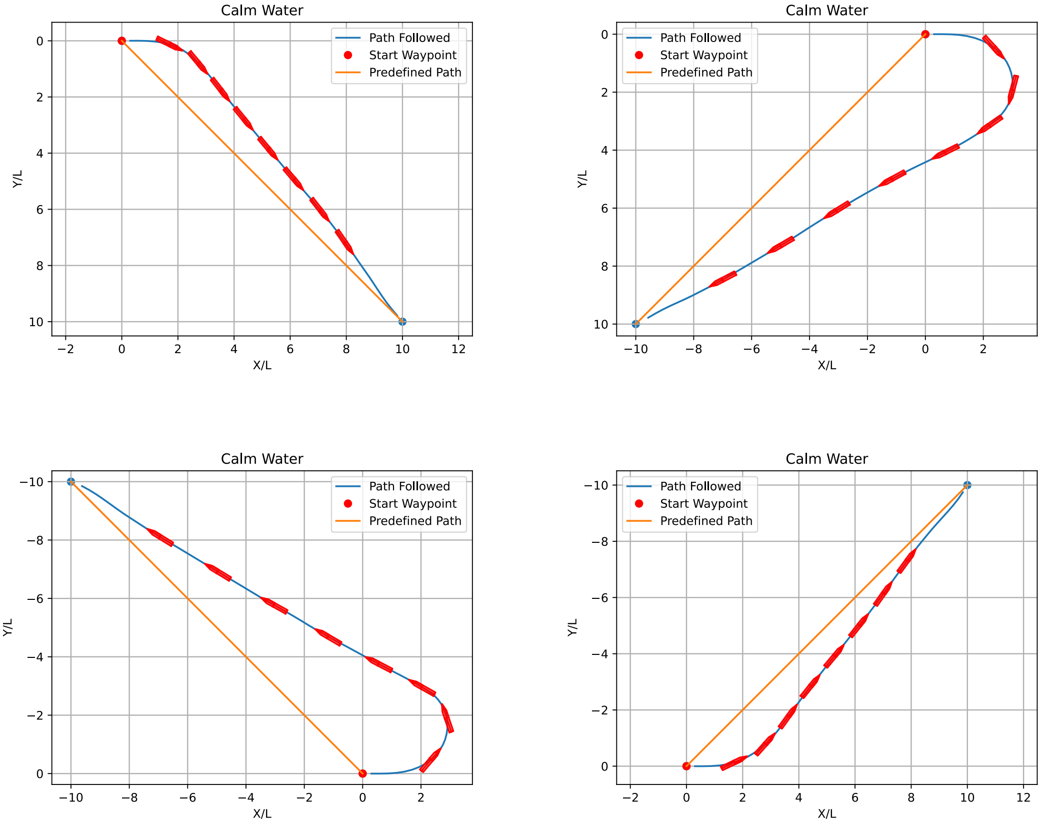


Figure 5.15: Single waypoint tracking for DQN

Table 5.7: Hyperparameters for DQN model

| **Hyperparameter** | **Value** |
|---|---|
| Initial Learning rate | 0.001 |
| Decay Steps | 50000 |
| Decay Rate | 0.4 |
| Hidden layers | (64,64) |
| Discount factor($\gamma$) | 0.96 |
| Sample batch size | 128 |
| Replay buffer size | 100000 |
| Activation function | tanh |
| Maximum time steps | 160 |
| Time step interval $\Delta t$ | 0.3 |
| Number of episodes | 10000 |
| Update frequency(time steps) | 10 |
| Time step interval $\Delta t$ | 0.3 |
| Target network update frequency(time steps) | 1 |
| Target update rate ($\tau$) | 0.01 |

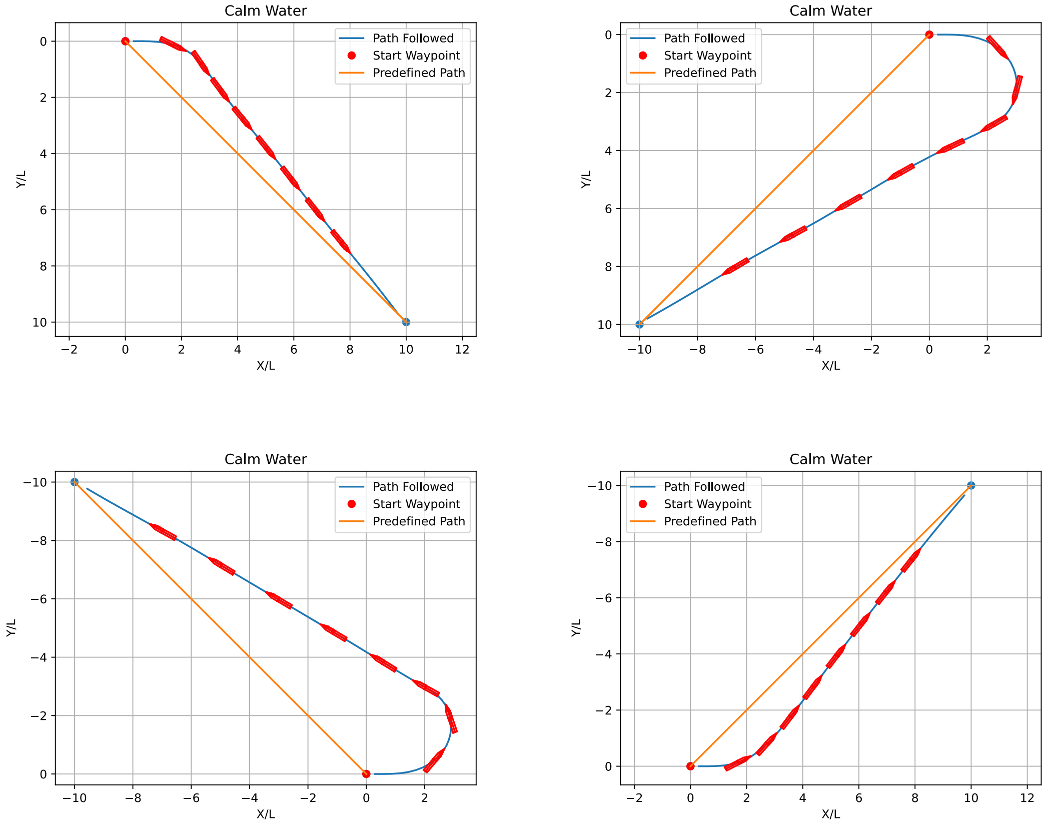


Figure 5.16: DDPG Single waypoint tracking

Table 5.8: Hyperparameters for PPO model

| **Hyperparameter** | **Value** |
|---|---|
| Initial Learning rate | 0.001 |
| Decay Steps | 3000 |
| Decay Rate | 0.5 |
| Actor layers | (128,128) |
| Critic layers | (128,128) |
| Discount factor($\gamma$) | 0.96 |
| Clip Ratio | 0.2 |
| Lambda | 0.95 |
| Entropy Regularisation | 0.01 |
| Replay buffer size | 8000 |
| Epochs | 10 |
| Episode per epochs | 50 |
| Number of Iteration | 100 |
| Time step interval $\Delta$t | 0.3 |

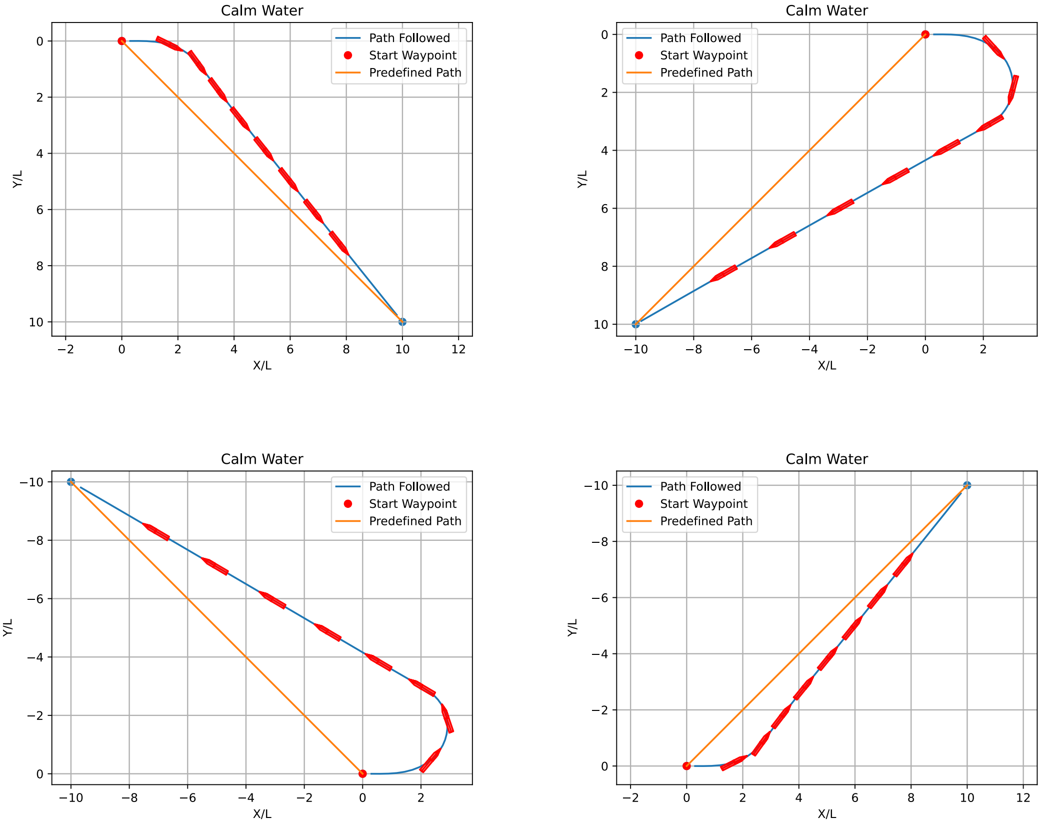


Figure 5.17: PPO Single waypoint tracking

Table 5.9: Hyperparameters for DDPG model

| **Hyperparameter** | **Value** |
|---|---|
| Actor Learning Rate | 0.001 |
| Critic Learning Rate | 0.001 |
| Actor Hidden Layers | (256,256) |
| Critic Hidden Layers | (256,256) |
| Discount factor($\gamma$) | 0.96 |
| Sample batch size | 128 |
| Replay buffer size | 100000 |
| Maximum time steps | 160 |
| Time step interval $\Delta$t | 0.3 |
| Update frequency(time steps) | 10 |
| Target network update frequency(time steps) | 1 |
| Target update rate ($\tau$) | 0.01 |

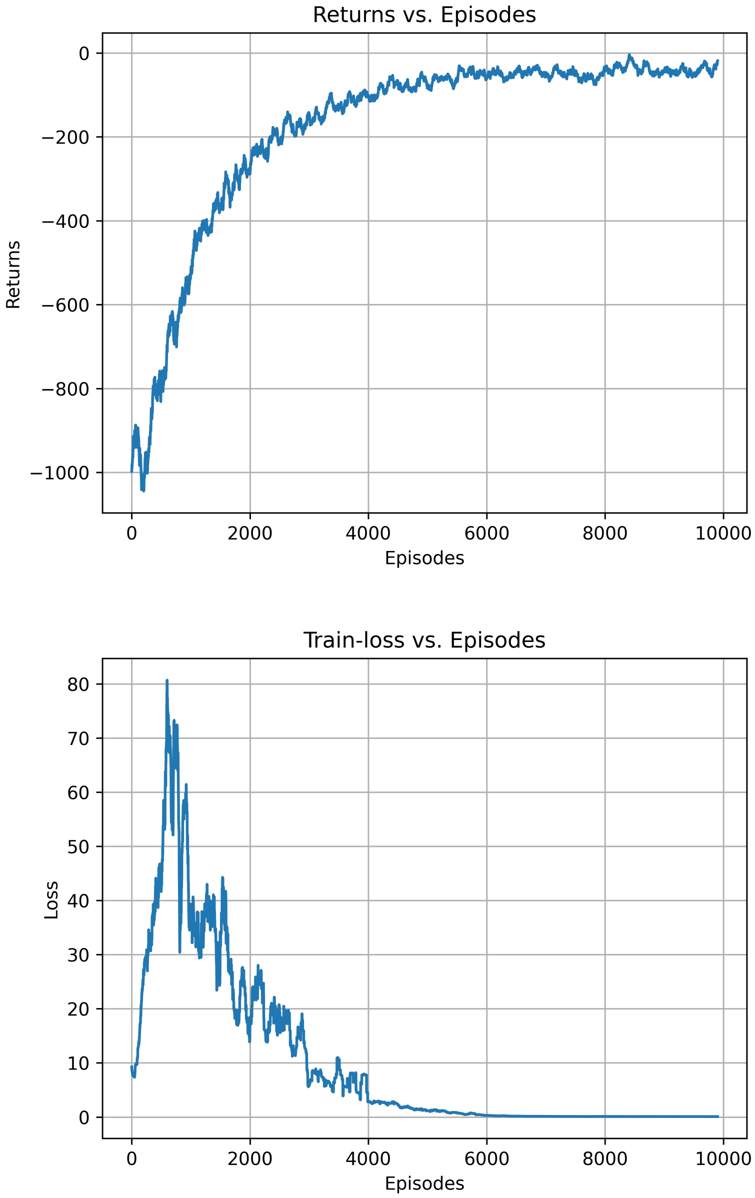


Figure 5.18: Training returns and loss for the DQN model

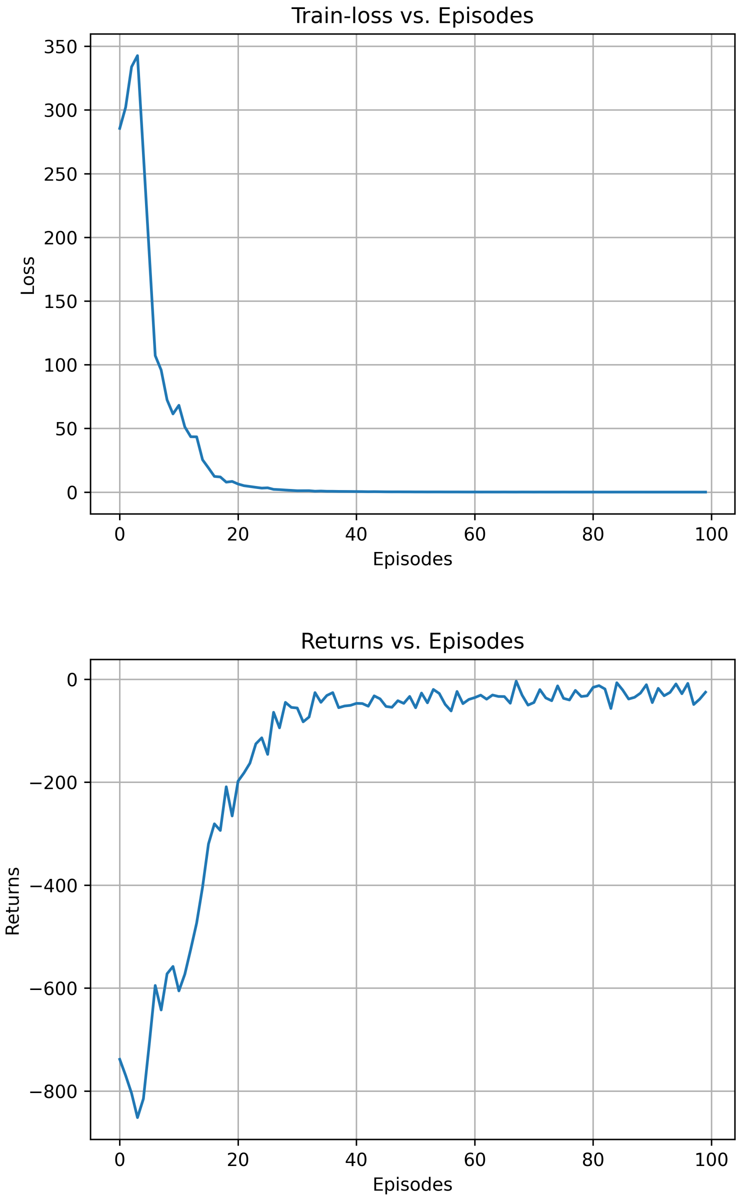


Figure 5.19: Training returns and loss for the PPO model

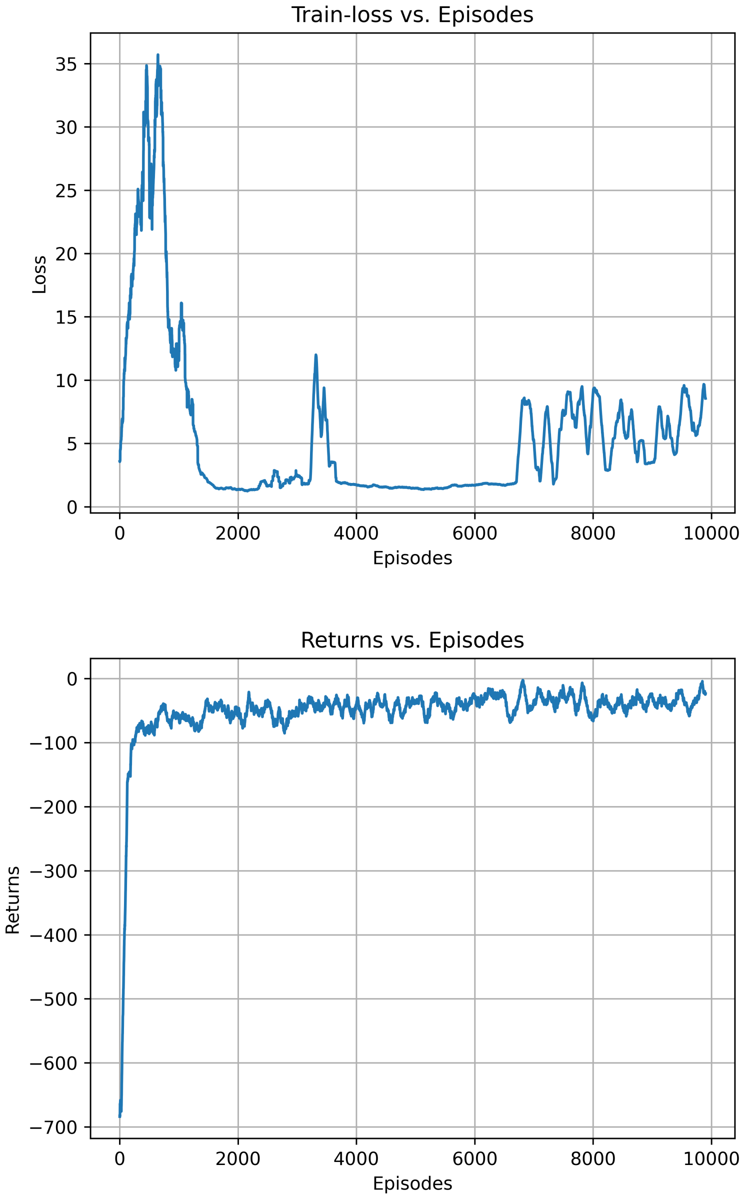


Figure 5.20: Training returns and loss for the DDPG model

can be seen that the agents were able to track the elliptical path successfully as shown in Fig. 5.21, 5.22, 5.23 and 5.24.

A path that resembles a “star" shape is ultimately established to assess the agent’s ability to make sharp turns at corners while travelling in a straight line. It should be noted that due to the ship’s significant inertia, the agent requires some time to execute the maneuver in the intended direction. Figure 5.25 illustrates that the agents were able to successfully execute the maneuver.

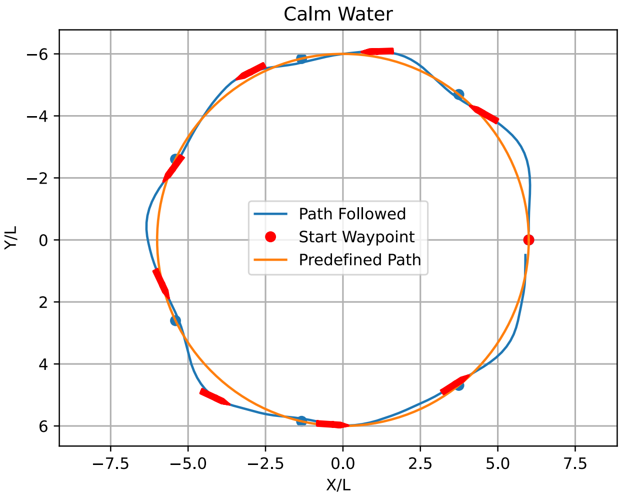
### 5.3.4 Performance with wind forces

The controllers are compared in the presence of uniform and constant wind. Though the controllers were able to track the waypoints successfully, their performance is compared with respect to RMSE and controller effort.

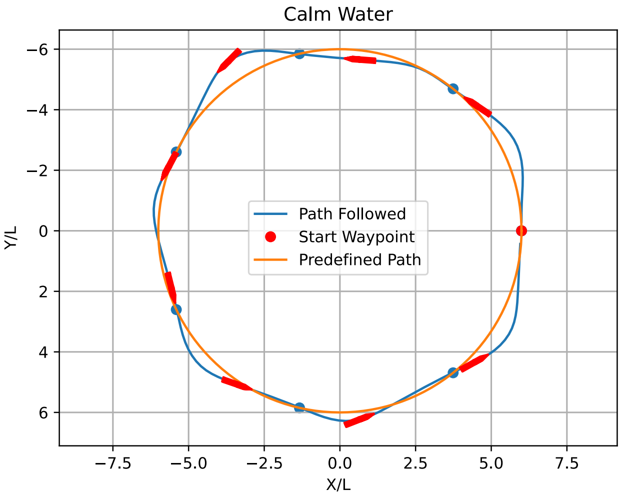
To simulate various scenarios, different wind speeds and directions are adjusted, and the performance of the controller in the presence of constant wind is assessed. While multiple wind speeds and directions are analyzed, specific cases are illustrated in Fig. 5.26, 5.27, 5.28, 5.29, 5.30 and 5.31, with the first four corresponding to straight line paths and the fifth and sixth showcasing ellipse and eight maneuvers under steady wind conditions. Despite the significant initial deviation caused by the beam wind in the ellipse and eight maneuvers, as demonstrated in Fig. 5.30 and 5.31, the model successfully follows all the waypoints and completes the desired path, indicating the controller’s effectiveness in rejecting disturbances.

### 5.3.5 Results

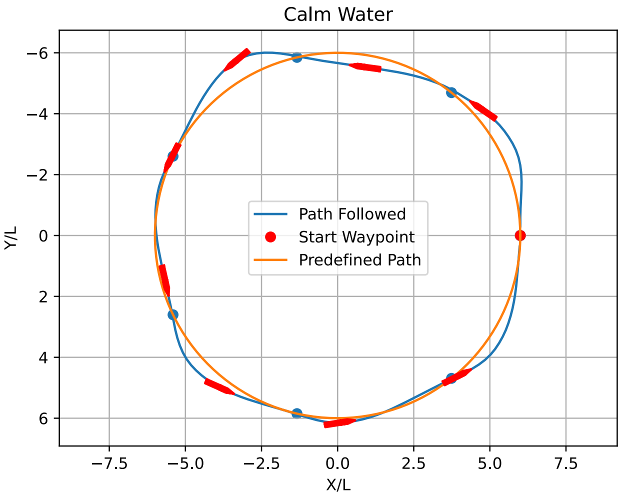
This section discusses the key features and outcomes of the data-driven RL controllers presented in previous sections. All three DRL algorithms have been successful in tracking waypoints and following specified paths. To facilitate a comparison, root mean squared errors (RMSE) and controller effort (C.E) are computed across the trajectory and listed in Table 5.10. Additionally, Fig. 5.32 presents a comparison of Loss and Returns for



(a) DQN

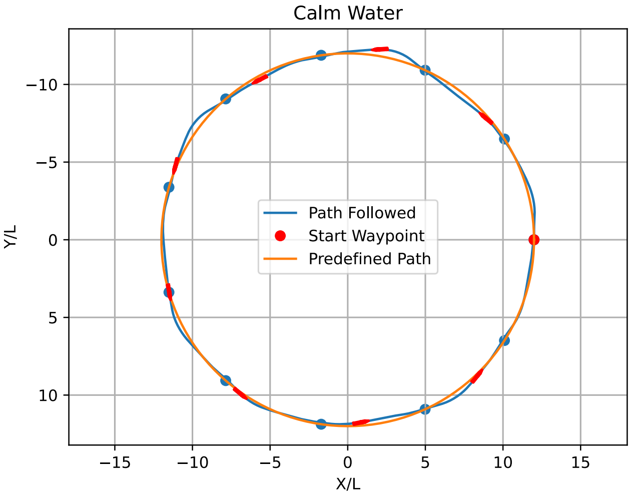


(b) PPO

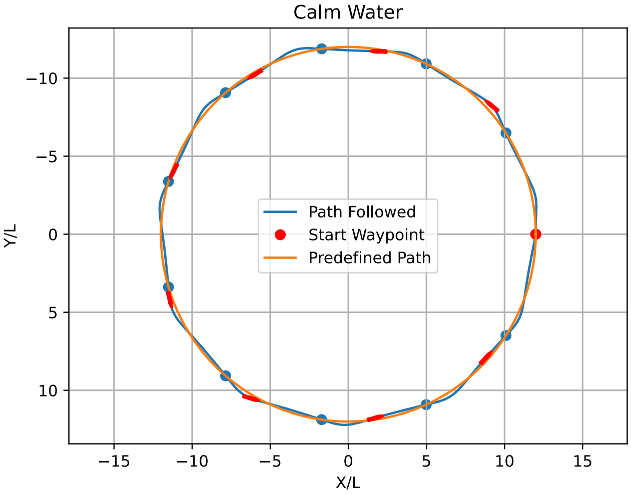


(c) DDPG

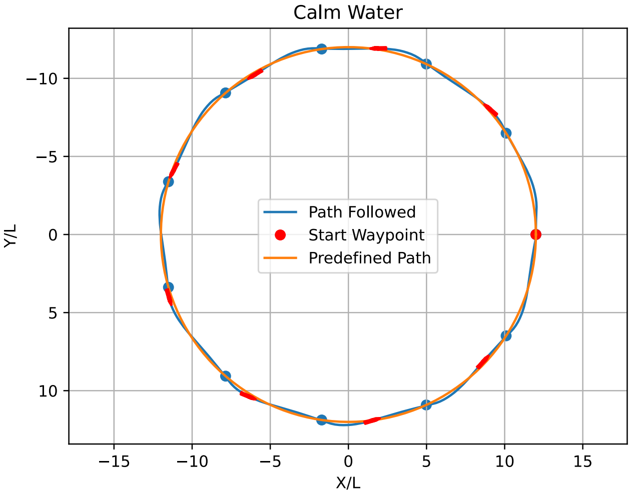
Figure 5.21: Circle (radius = 6L) maneuver waypoint tracking



(a) DQN

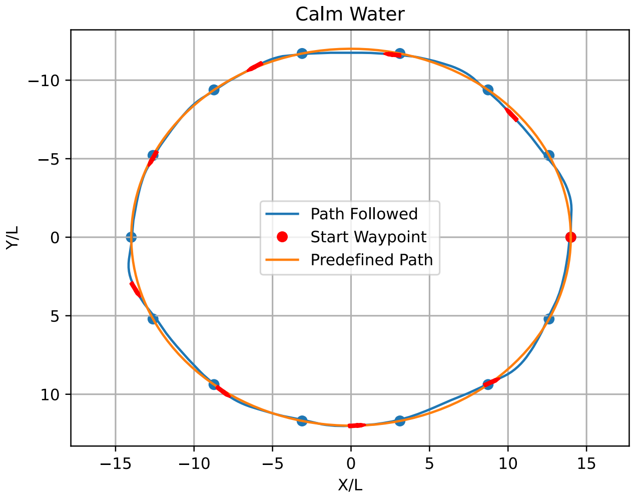


(b) PPO

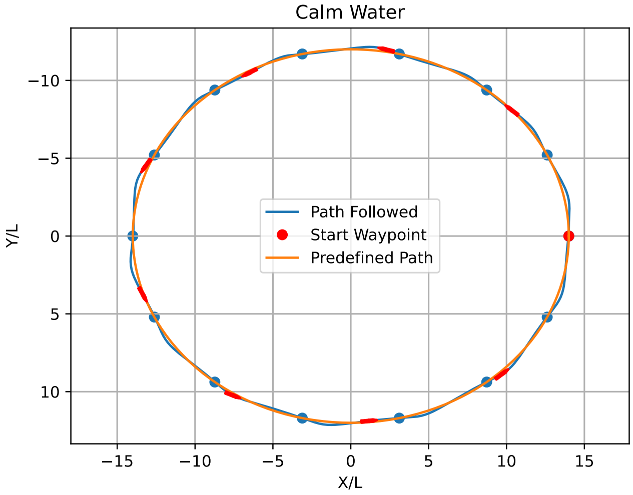


(c) DDPG

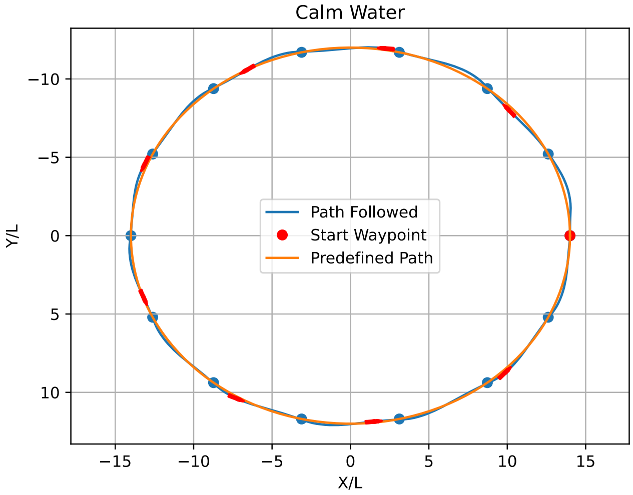
Figure 5.22: Circle (radius = 12L) maneuver waypoint tracking



(a) DQN

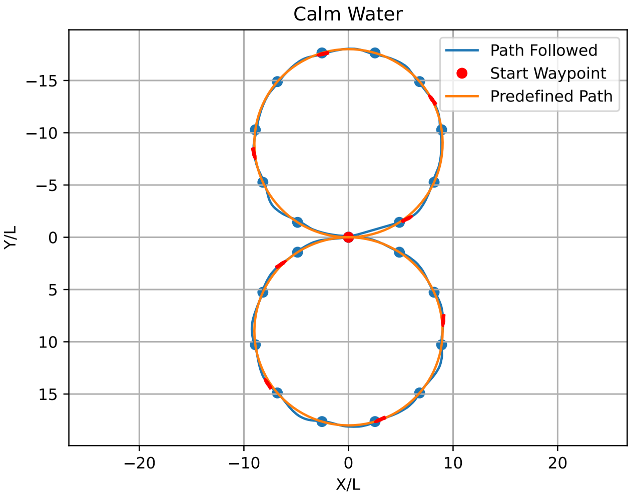


(b) PPO

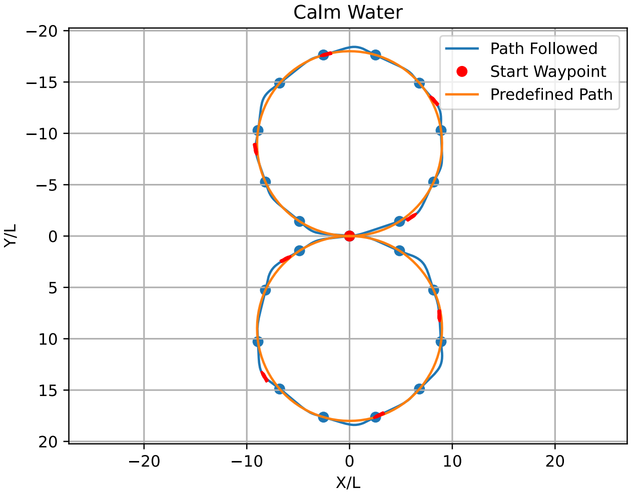


(c) DDPG

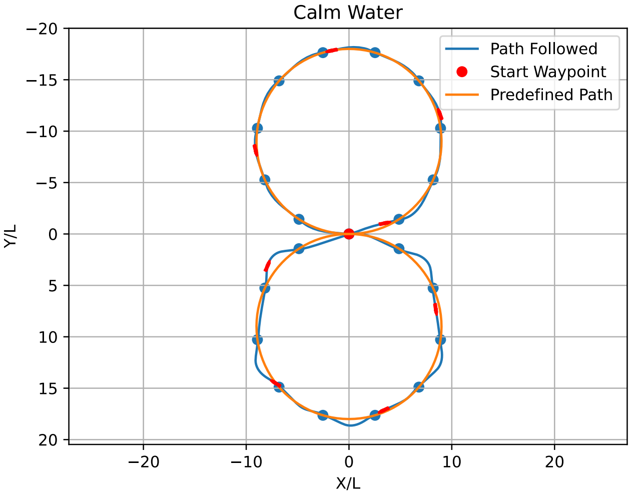
Figure 5.23: Ellipse maneuver waypoint tracking



(a) DQN

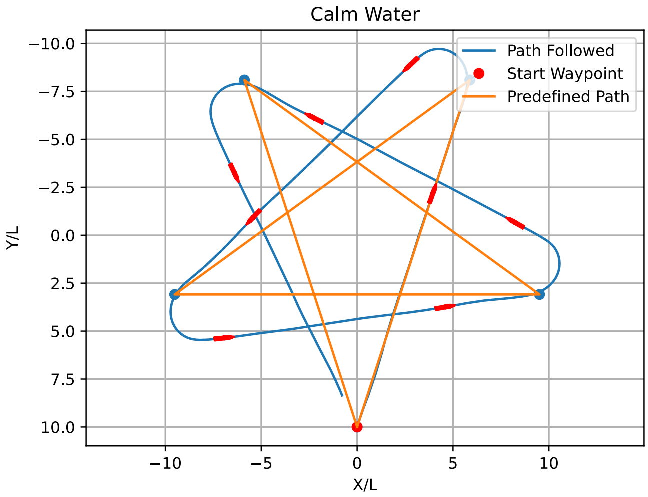


(b) PPO

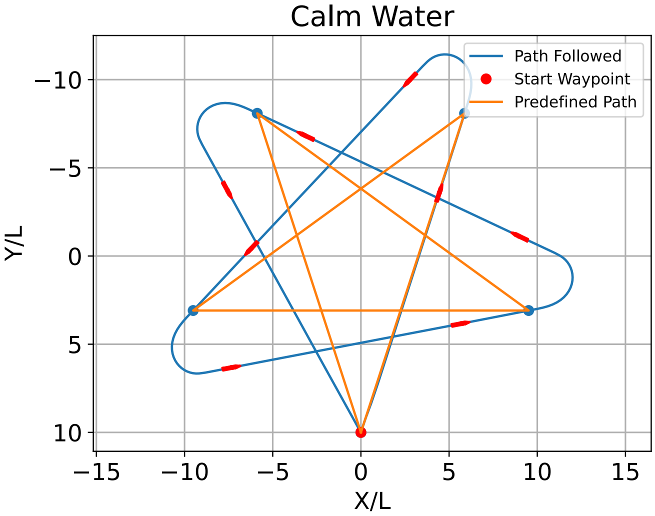


(c) DDPG

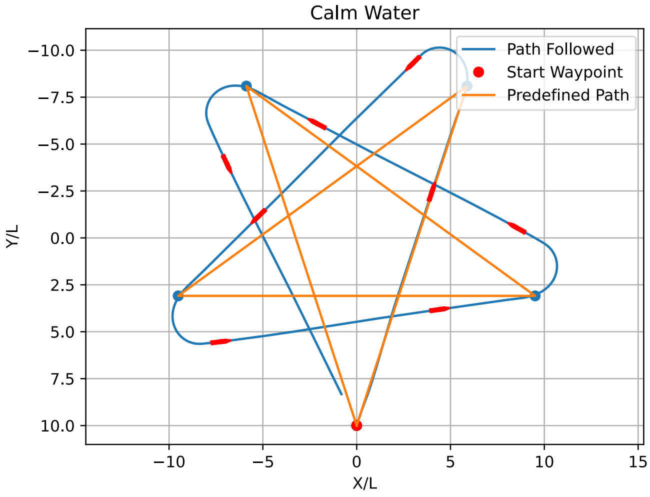
Figure 5.24: Eight maneuver waypoint tracking



(a) DQN

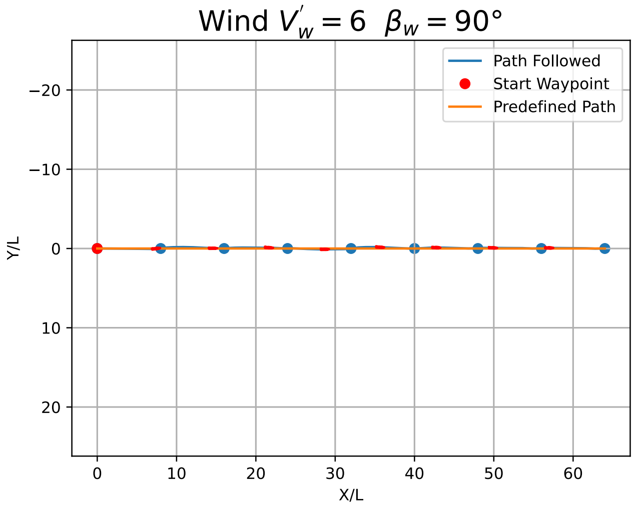


(b) PPO

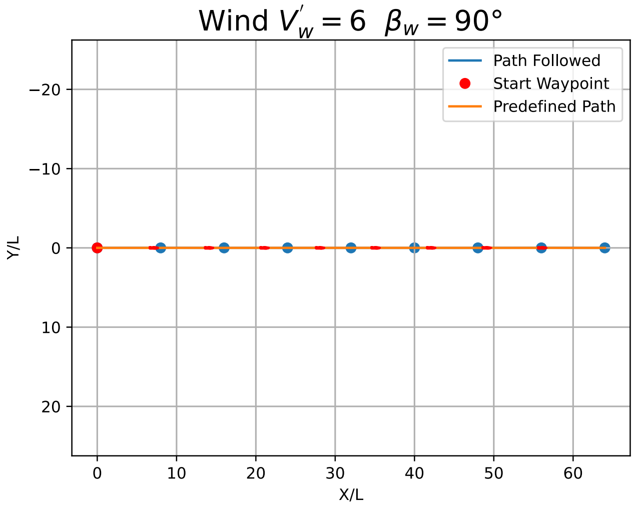


(c) DDPG

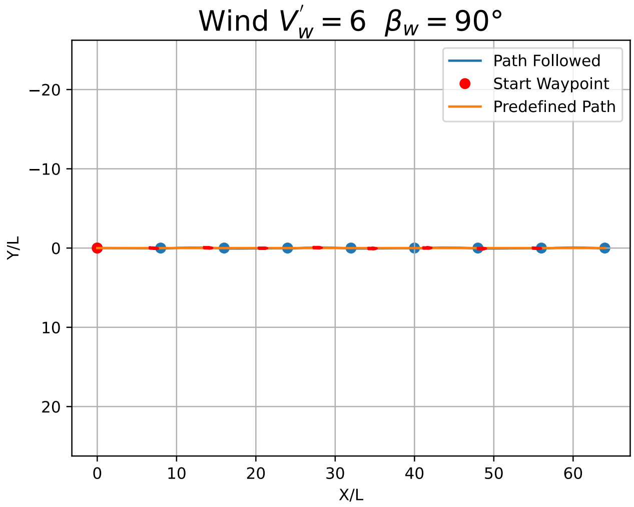
Figure 5.25: Star maneuver waypoint tracking



(a) DQN

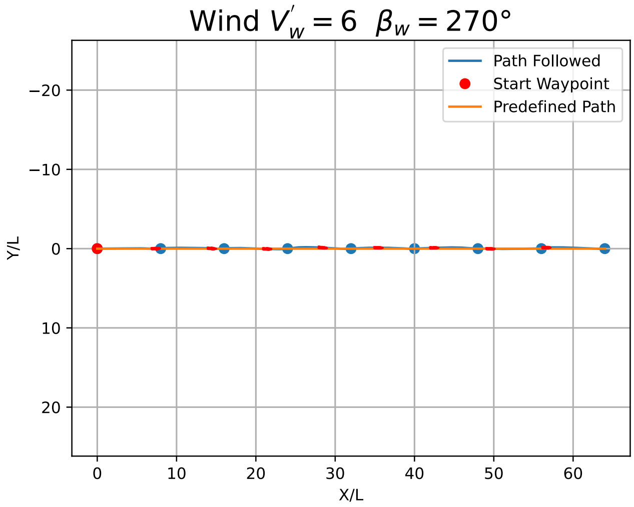


(b) PPO

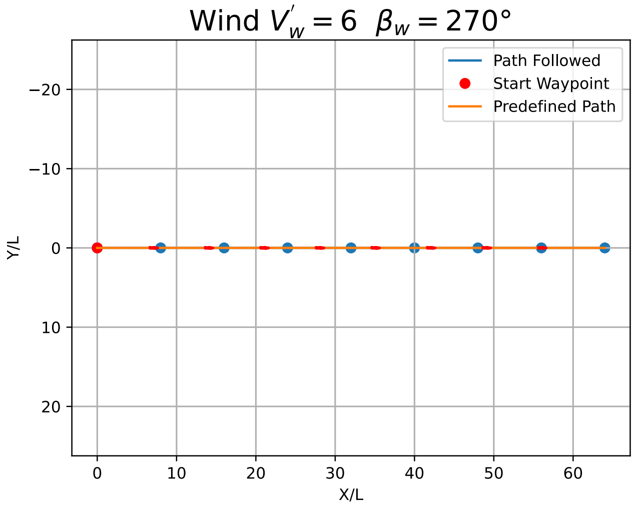


(c) DDPG

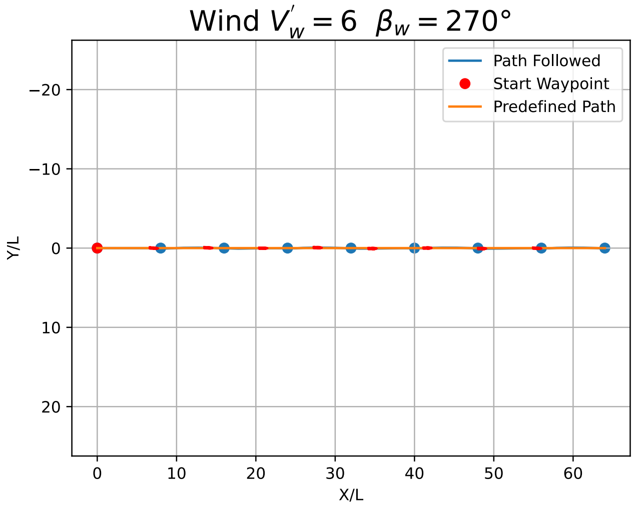
Figure 5.26: Straight Line Path with wind $V_w = 6L$ and $\beta_w = 90°$



(a) DQN

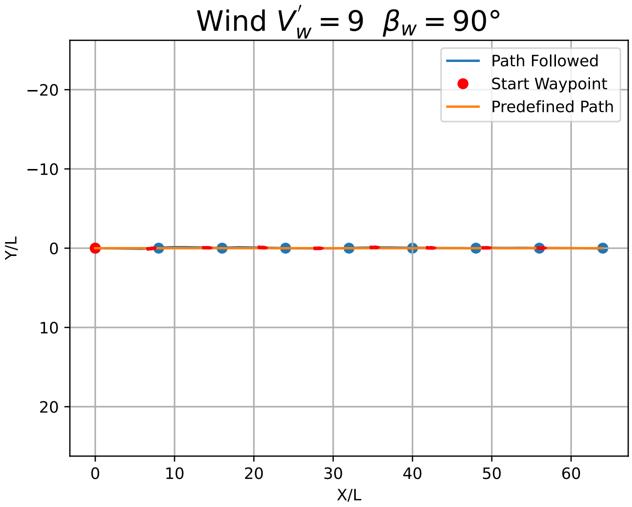


(b) PPO

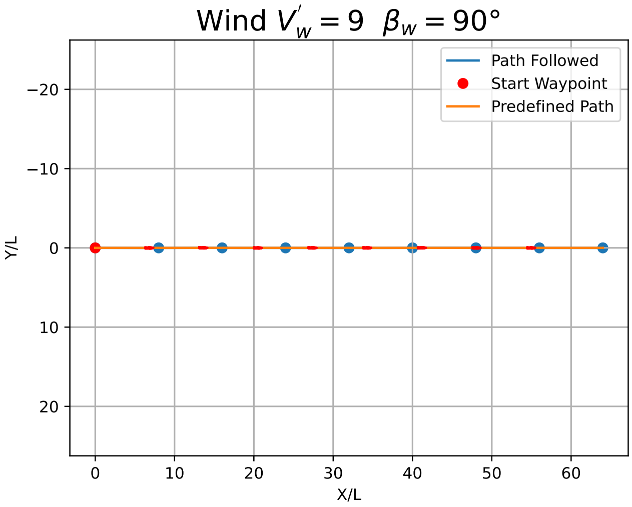


(c) DDPG

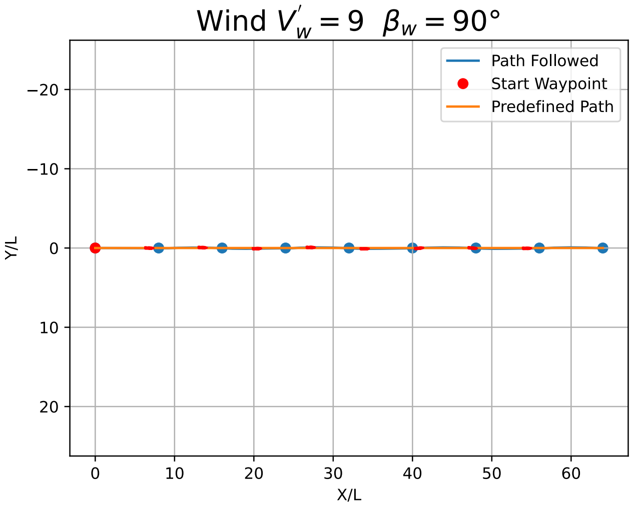
Figure 5.27: Straight Line Path with wind $V_w = 6L$ and $\beta_w = 270°$



(a) DQN

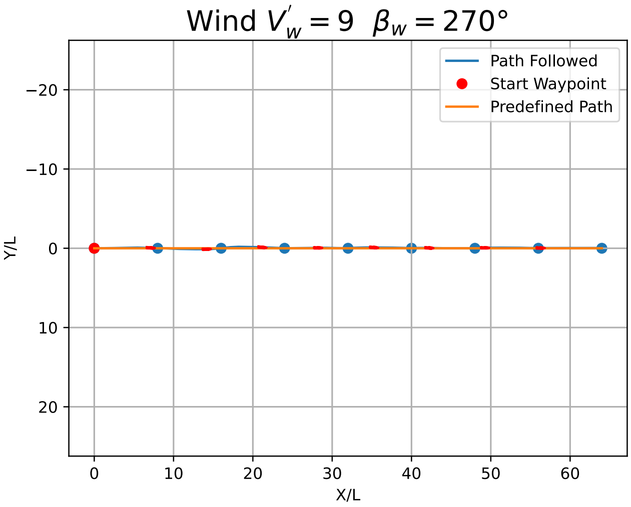


(b) PPO

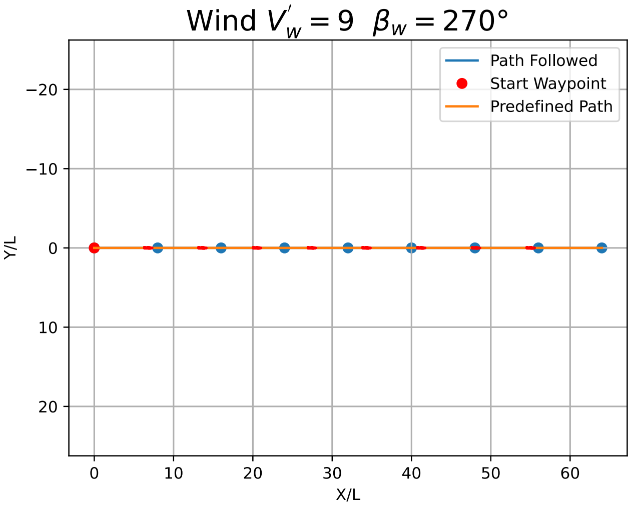


(c) DDPG

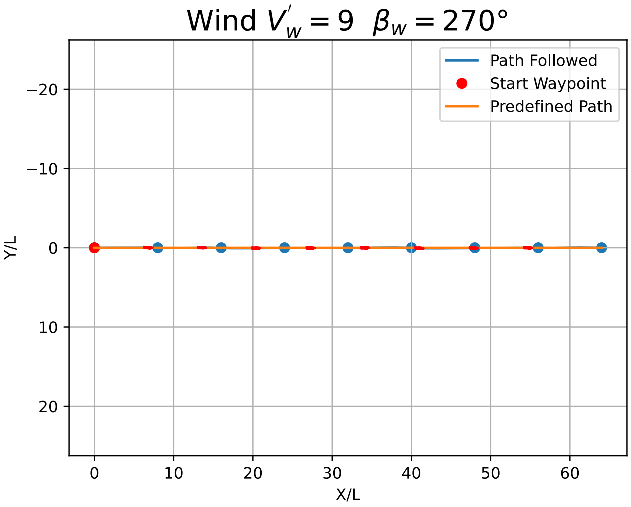
Figure 5.28: Straight Line Path with wind $V_w = 9L$ and $\beta_w = 90°$



(a) DQN

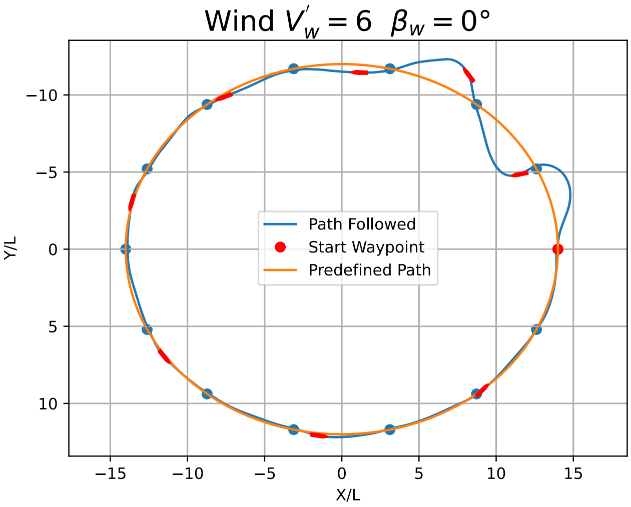


(b) PPO

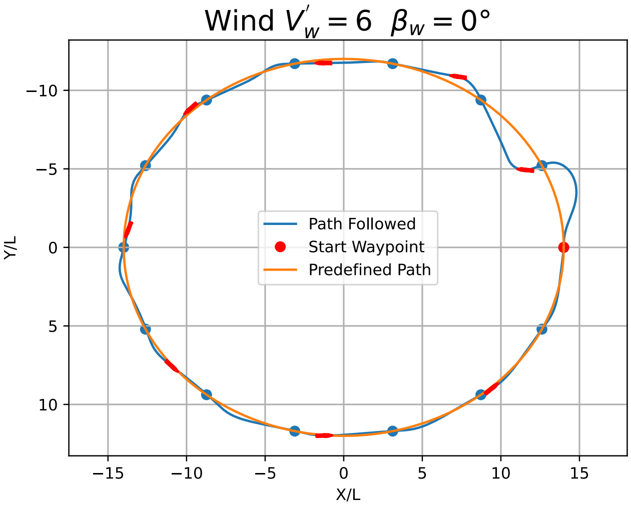


(c) DDPG

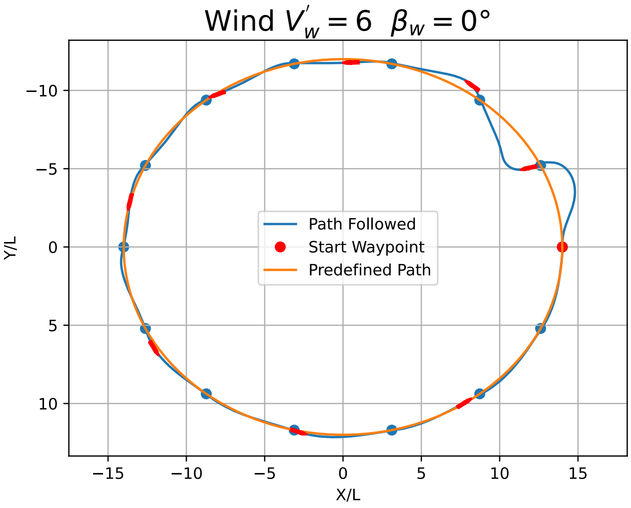
Figure 5.29: Straight Line Path with wind $V_w = 9L$ and $\beta_w = 270°$



(a) DQN

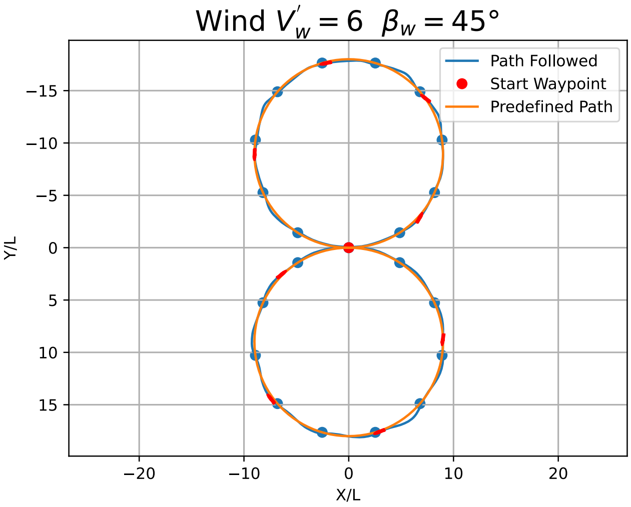


(b) PPO



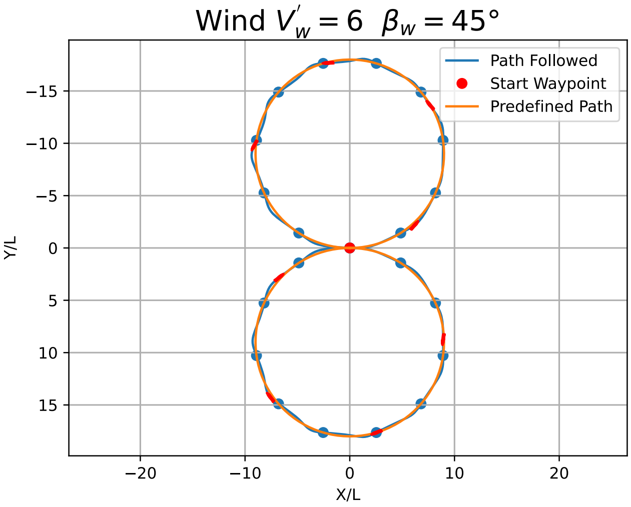
(c) DDPG

Figure 5.30: Ellipse Path with wind $V_w = 6L$ and $\beta_w = 0°$



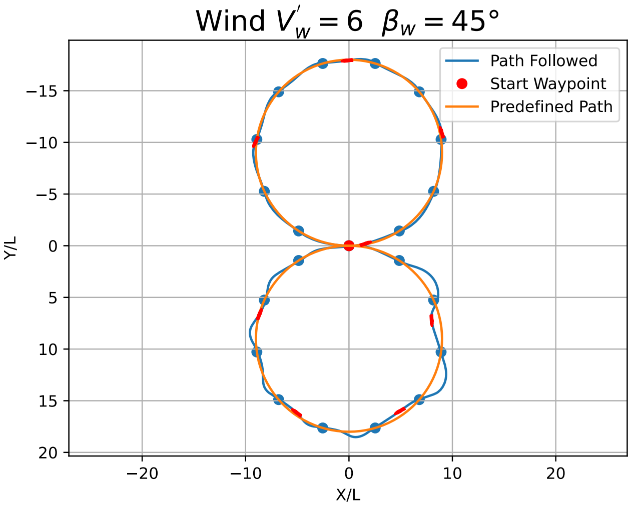
Wind $V_w' = 6 \;\; \beta_w = 45°$

(a) DQN



Wind $V_w' = 6 \;\; \beta_w = 45°$

(b) PPO



Wind $V_w' = 6 \;\; \beta_w = 45°$

(c) DDPG

Figure 5.31: Eight Path with wind $V_w = 6L$ and $\beta_w = 45°$

three controllers. It is observed that the PPO algorithm achieves convergence within 3000 episodes, while DDPG takes 5000 episodes and DQN takes 8000 episodes. The comparison plot is drawn based on these episode counts.

In terms of computational demand, DQN stands out as less demanding compared to DDPG and PPO since the latter two utilize multiple neural networks. However, in practical scenarios where continuous action is required, DDPG and PPO are more suitable choices.

Considering the findings in Table 5.10, the PPO controller consistently exhibits the lowest values of RMSE and C.E. Notably, when the ship attempts to move in a straight line, PPO demonstrates an RMSE approximately 10 times better than the other controllers. Additionally, PPO achieves convergence faster than DDPG and DQN, requiring fewer training scenarios.

Considering these aspects, the PPO algorithm emerges as the most promising among the three alternatives.

## 5.4 STATIC OBSTACLE AVOIDANCE

In this sub-section, DRL-based controllers are trained to avoid static obstacles as mentioned in Sec. 4.2.

### 5.4.1 Hyperparameters of the network

The agent is calibrated for satisfactory waypoint tracking by tuning the hyperparameters. The agent is then tested by simulating various maneuvers for waypoint tracking. The learning rate is chosen as an exponentially decaying function. The hyperparameters for the DQN and DDPG are shown in Table 5.11 and 5.12. The returns vs episodes and loss vs episodes plots for DQN and DDPG are shown in Fig. 5.33 and 5.34

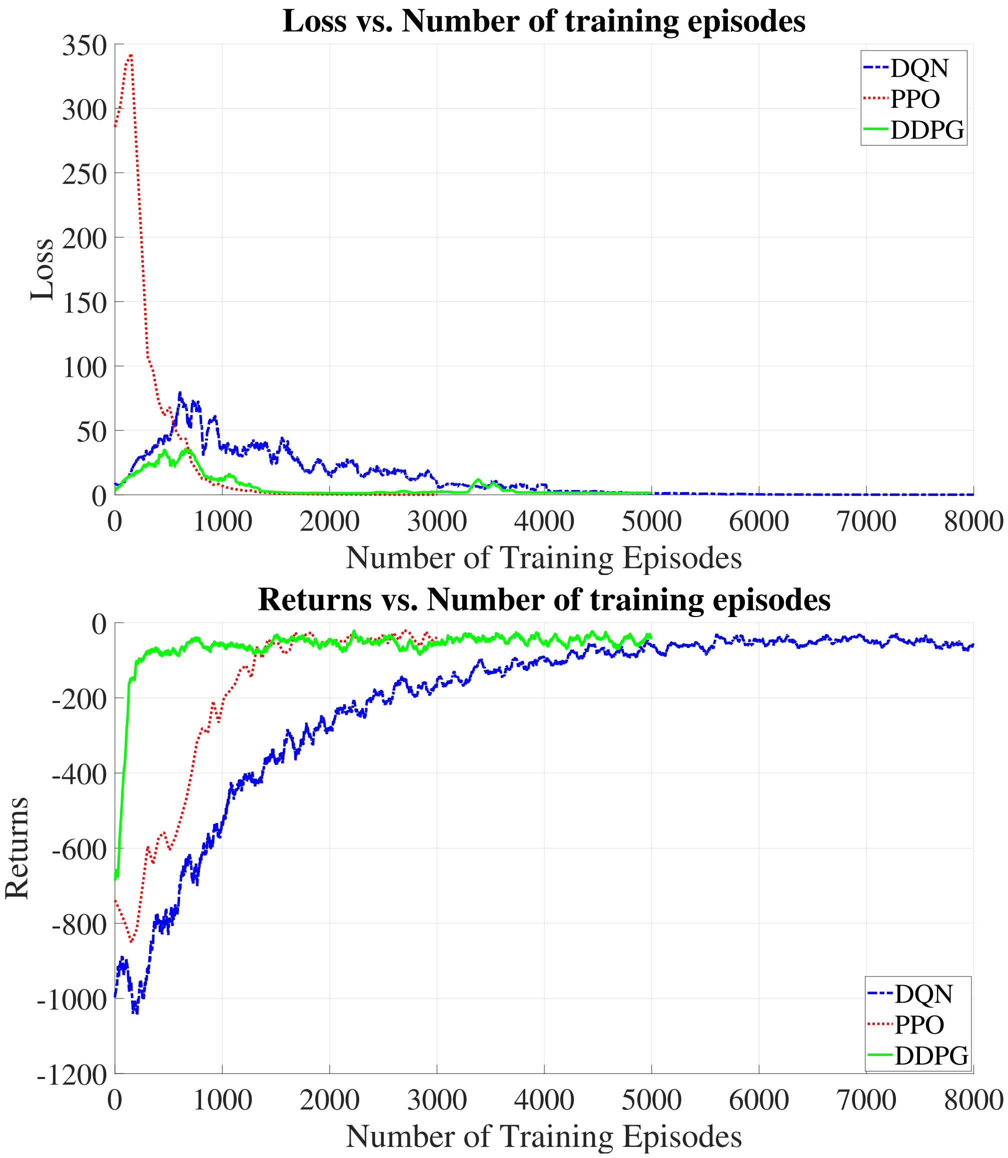


Figure 5.32: Comparison of Loss and Returns of different DRL-based controllers

Table 5.10: Comparison of the three DRL-based controllers in terms of RMSE and CE

| Maneuver | DQN | | PPO | | DDPG | |
|---|---|---|---|---|---|---|
| | RMSE | C.E | RMSE | C.E | RMSE | C.E |
| (10,10) | 0.9101 | 0.1632 | 0.8405 | **0.1214** | **0.8343** | 0.1444 |
| (-10,10) | 2.4795 | 0.2237 | 2.4427 | **0.1861** | **2.4111** | 0.2859 |
| (-10,-10) | **2.2915** | 0.2696 | 2.3404 | **0.1832** | 2.3636 | 0.2000 |
| (10,-10) | 0.9019 | 0.2091 | **0.8240** | 0.1230 | 0.8730 | **0.1082** |
| Ellipse | **0.2736** | 0.2411 | 0.3146 | 0.1971 | 0.3011 | **0.1493** |
| Eight | **0.3545** | 0.3128 | 0.4158 | 0.2375 | 0.4830 | **0.2333** |
| Star | **1.4354** | 0.2560 | 1.5489 | **0.1422** | 1.5472 | 0.1489 |
| Circle (Radius = 6L) | **0.6129** | 0.3749 | 0.6492 | 0.2787 | 0.6193 | **0.2292** |
| Circle (Radius = 12L) | 0.4186 | 0.2306 | **0.3921** | 0.1809 | 0.3981 | **0.1572** |
| Ellipse ($V_w = 6$,$\beta_w = 0°$) | 0.7463 | 0.2986 | **0.5453** | 0.2506 | 0.5726 | **0.2134** |
| Eight ($V_w = 6$,$\beta_w = 45°$) | 0.3591 | 0.3226 | **0.3256** | **0.2428** | 0.4905 | 0.2737 |
| Straight Line($V_w = 6$,$\beta_w = 90°$) | 0.0864 | 0.1611 | **0.0048** | **0.0191** | 0.0377 | 0.0727 |
| Straight Line($V_w = 6$,$\beta_w = 270°$) | 0.0799 | 0.1700 | **0.0032** | **0.0140** | 0.0390 | 0.0744 |
| Straight Line($V_w = 9$,$\beta_w = 90°$) | 0.0587 | 0.1701 | **0.0085** | **0.0402** | 0.0565 | 0.0784 |
| Straight Line($V_w = 9$,$\beta_w = 270°$) | 0.0688 | 0.1850 | **0.0027** | **0.0168** | 0.0337 | 0.0643 |

Table 5.11: Hyperparameters for DQN model

| **Hyperparameter** | **Value** |
|---|---|
| Initial Learning rate | 0.00075 |
| Decay Steps | 50000 |
| Decay Rate | 0.4 |
| Hidden layers | (128,128) |
| Discount factor($\gamma$) | 0.97 |
| Sample batch size | 128 |
| Replay buffer size | 100000 |
| Activation function | tanh |
| Maximum time steps | 160 |
| Time step interval Δt | 0.3 |
| Number of episodes | 9000 |
| Update frequency(time steps) | 10 |
| Time step interval Δt | 0.3 |
| Target network update frequency(time steps) | 1 |
| Target update rate ($\tau$) | 0.01 |

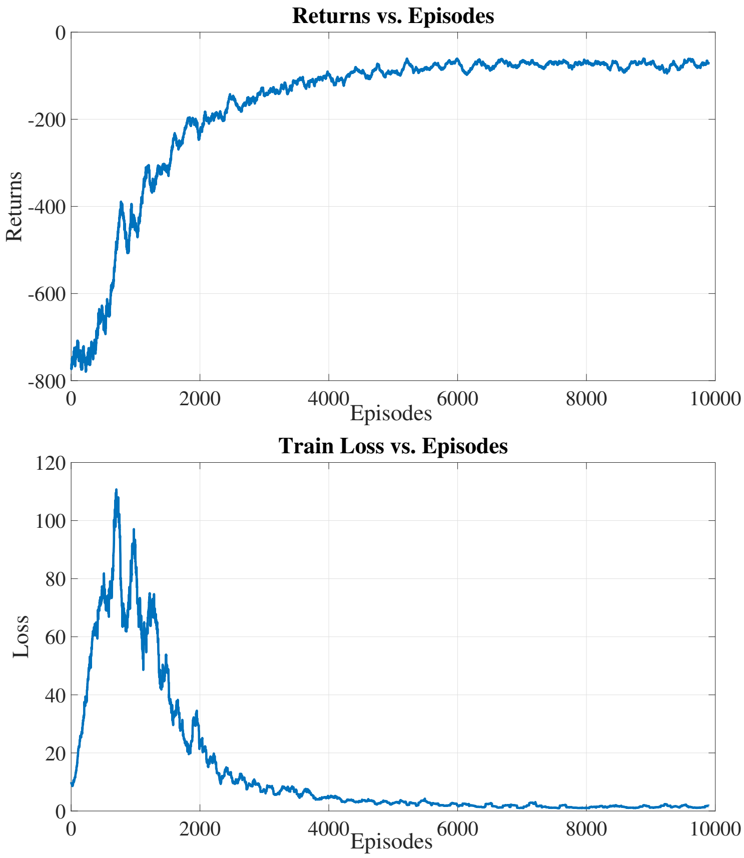


Figure 5.33: Training returns and loss for the DQN model

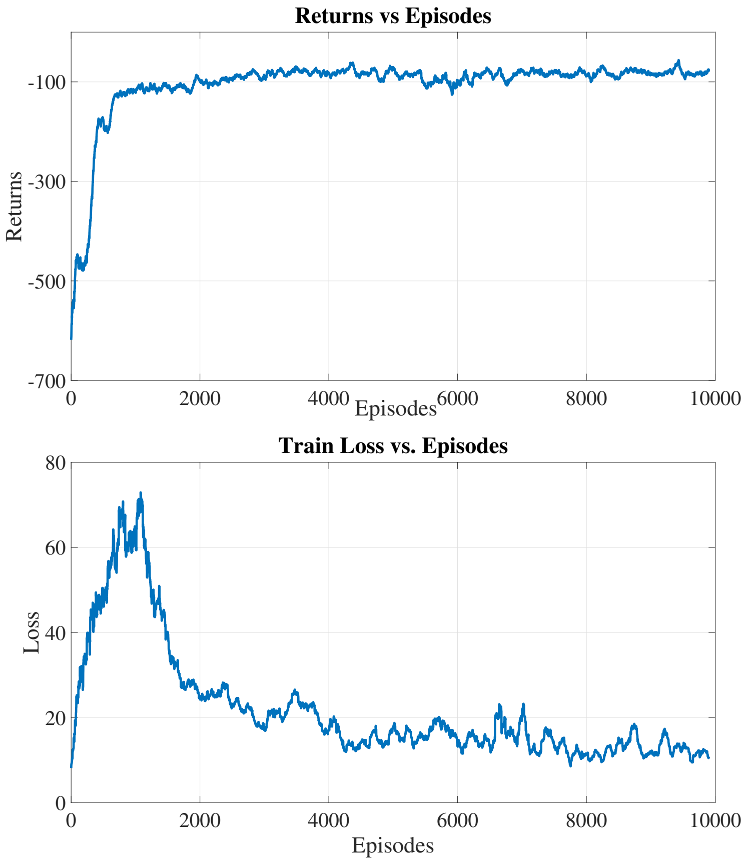


Figure 5.34: Training returns and loss for the DDPG model

Table 5.12: Hyperparameters for DDPG model

| **Hyperparameter** | **Value** |
|---|---|
| Actor Initial Learning Rate | 0.0001 |
| Actor Decay Steps | 30000 |
| Actor Decay Rate | 0.5 |
| Critic Initial Learning Rate | 0.00075 |
| Critic Decay Steps | 30000 |
| Critic Decay Rate | 0.5 |
| Actor Hidden Layers | (128,128) |
| Critic Hidden Layers | (128,128) |
| Discount factor($\gamma$) | 0.98 |
| Sample batch size | 128 |
| Replay buffer size | 100000 |
| Maximum time steps | 160 |
| Noise Multiplier | 0.15 |
| Noise Episodes | 12000 |
| Time step interval $\Delta$t | 0.3 |
| Update frequency(time steps) | 10 |
| Target network update frequency(time steps) | 1 |
| Target update rate ($\tau$) | 0.01 |

### 5.4.2 Single static obstacle avoidance

In this study, a model was trained with specific hyperparameters as listed in Table 5.11 and 5.12. The trained model was then tested on various path configurations, obstacle sizes and positions to evaluate its performance in obstacle avoidance and waypoint tracking. Results showed that both DQN and DDPG controllers were successful in performing these tasks, as evidenced by the trajectory plots in Fig. 5.35, 5.36, 5.40, and 5.41.

To evaluate the robustness of the model, an obstacle was placed in the path of the ship and the controllers were compared in terms of their RMSE of the cross-track error and Controller Effort (CE). The trajectory plots in Fig. 5.37, 5.38, 5.42, 5.43, 5.39, and 5.44 were used to compare the performance of the controllers.

Table 5.13, 5.14, 5.15, and 5.16 summarize the results of this comparison. Overall, it was observed that DDPG outperformed DQN in cases where the obstacle was in between the line joining the waypoints, especially when the size of the obstacle was large. In contrast,

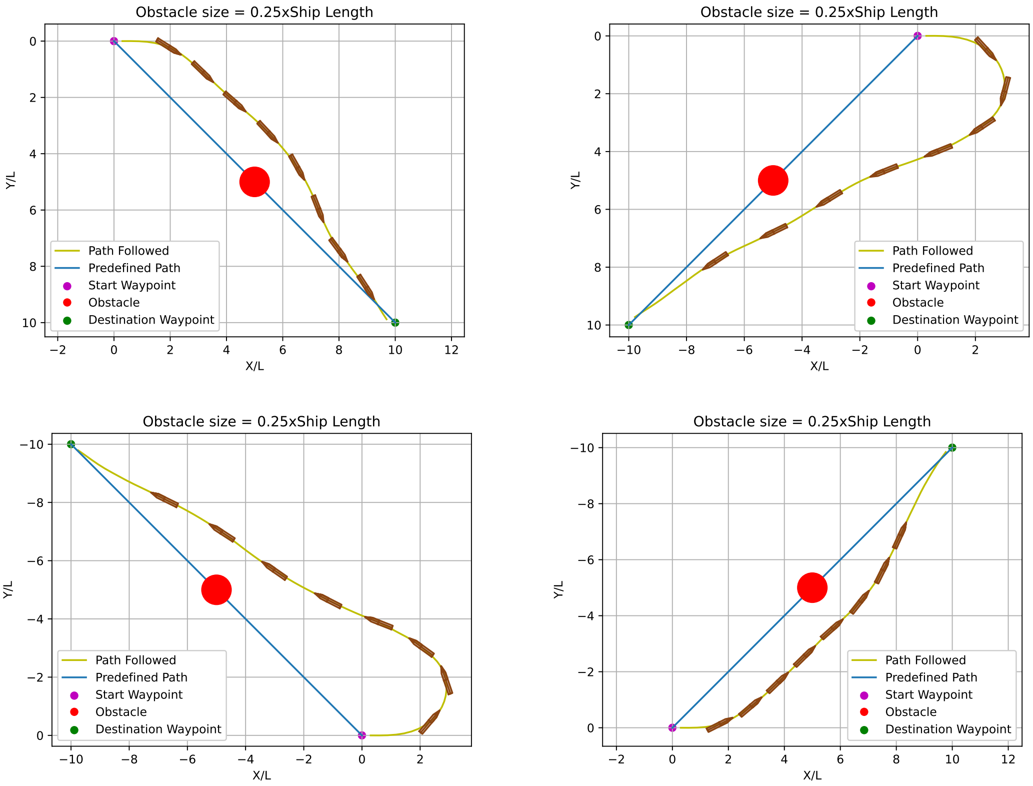


Figure 5.35: DQN agent successfully avoids obstacle of size 0.25L when present on line joining the waypoints

DQN performed better in cases where the obstacle was in the path of the ship, especially when the size of the obstacle was small. However, the difference in RMSE and CE between the two controllers was not significant and did not exceed 15%. Additionally, as the size of the obstacle increased, the RMSE value of DDPG decreased more compared to the DQN controller.

### 5.4.3 Multiple Static Obstacle Avoidance

The model was trained with a static obstacle in the environment, limiting the neural network's ability to consider only one obstacle at a time, with no option to adjust the network's size. Consequently, we utilized Collision Risk (CR) to identify the most critical obstacle with a potential collision risk and conveyed that obstacle's information to the neural network to evade it. Fig. 5.45 shows the DQN and DDPG agent to avoid multiple static obstacles of size $0.25L$ while performing a square maneuver. Clearly, the

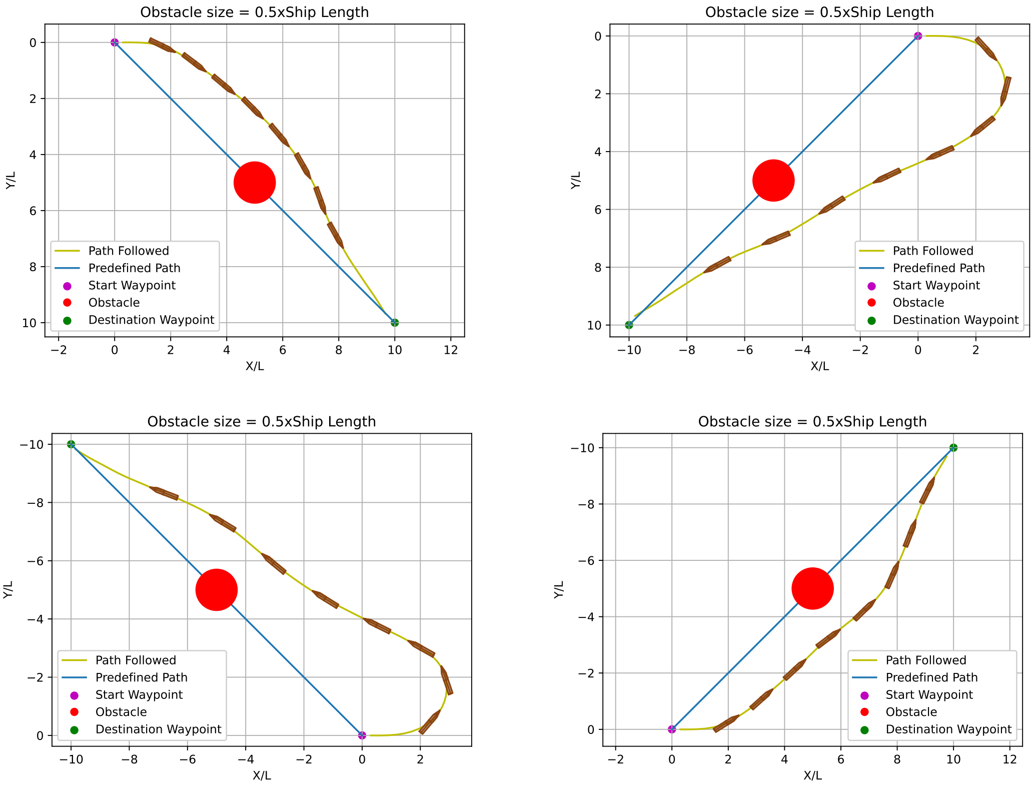


Figure 5.36: DQN agent successfully avoids obstacle of size 0.5L when present on line joining the waypoints

Table 5.13: Comparison of agent when obstacle of size 0.25L is in between line joining the waypoints

| Maneuver | DQN | | DDPG | |
|---|---|---|---|---|
| | RMSE | C.E | RMSE | C.E |
| (10,10) | 1.1343 | 0.3150 | **0.9800** | **0.2057** |
| (-10,10) | 2.3476 | **0.2885** | **2.2222** | 0.3023 |
| (-10,-10) | 2.3030 | **0.2891** | **2.2059** | 0.3461 |
| (10,-10) | 1.2163 | **0.2411** | **1.0171** | 0.3223 |
| (15,0) | 0.9902 | 0.2337 | **0.4945** | **0.1762** |

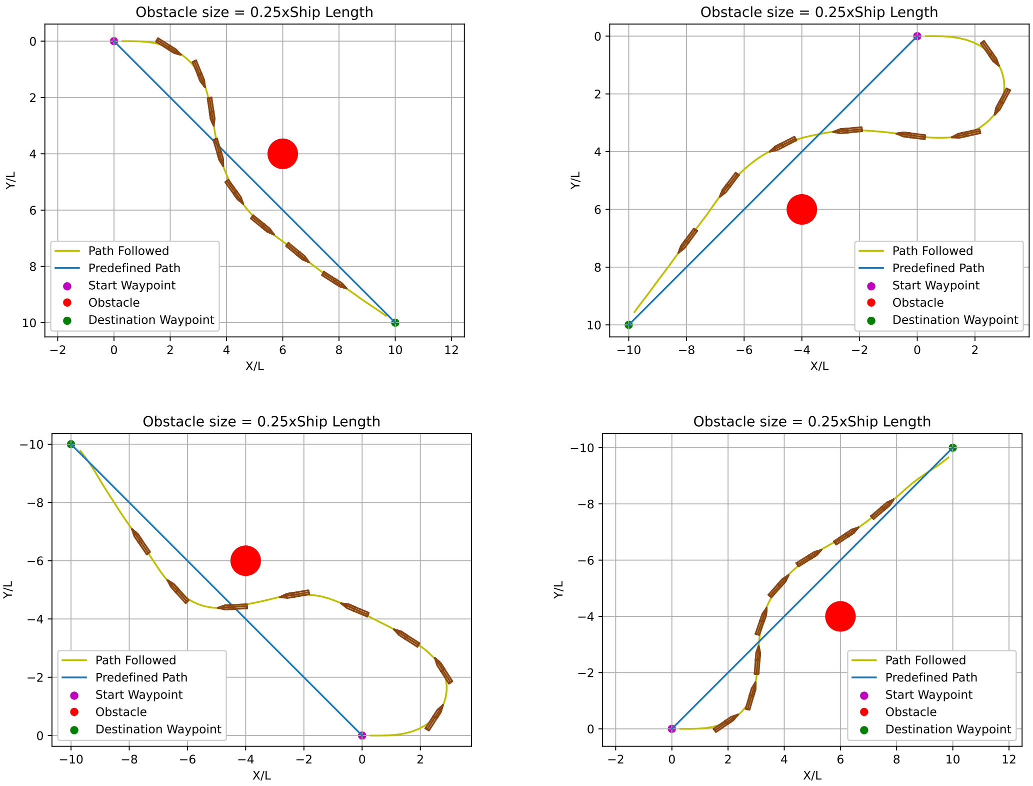


Figure 5.37: DQN agent successfully avoids obstacle of size 0.25L when present in path of ship

Table 5.14: Comparison of agent when obstacle of size 0.5L is in between line joining the waypoints

| Maneuver | DQN | | DDPG | |
|---|---|---|---|---|
| | RMSE | C.E | RMSE | C.E |
| (10,10) | 1.2944 | 0.2954 | **1.0624** | **0.1845** |
| (-10,10) | 2.4187 | 0.3096 | **2.2609** | **0.2926** |
| (-10,-10) | 2.3179 | **0.2737** | **2.2126** | 0.3709 |
| (10,-10) | 1.3530 | 0.2983 | **1.1100** | **0.2963** |
| (15,0) | 1.189 | 0.2756 | **0.6402** | **0.1580** |

Table 5.15: Comparison of agent when obstacle of size 0.25L is in path of the ship

| Maneuver | DQN | | DDPG | |
|---|---|---|---|---|
| | RMSE | C.E | RMSE | C.E |
| (10,10) | **0.7876** | 0.2874 | 0.8226 | **0.2837** |
| (-10,10) | **2.0379** | 0.3355 | 2.1053 | **0.3295** |
| (-10,-10) | 2.0965 | 0.3692 | **2.0563** | **0.2812** |
| (10,-10) | **0.7161** | 0.3132 | 0.8074 | **0.2038** |

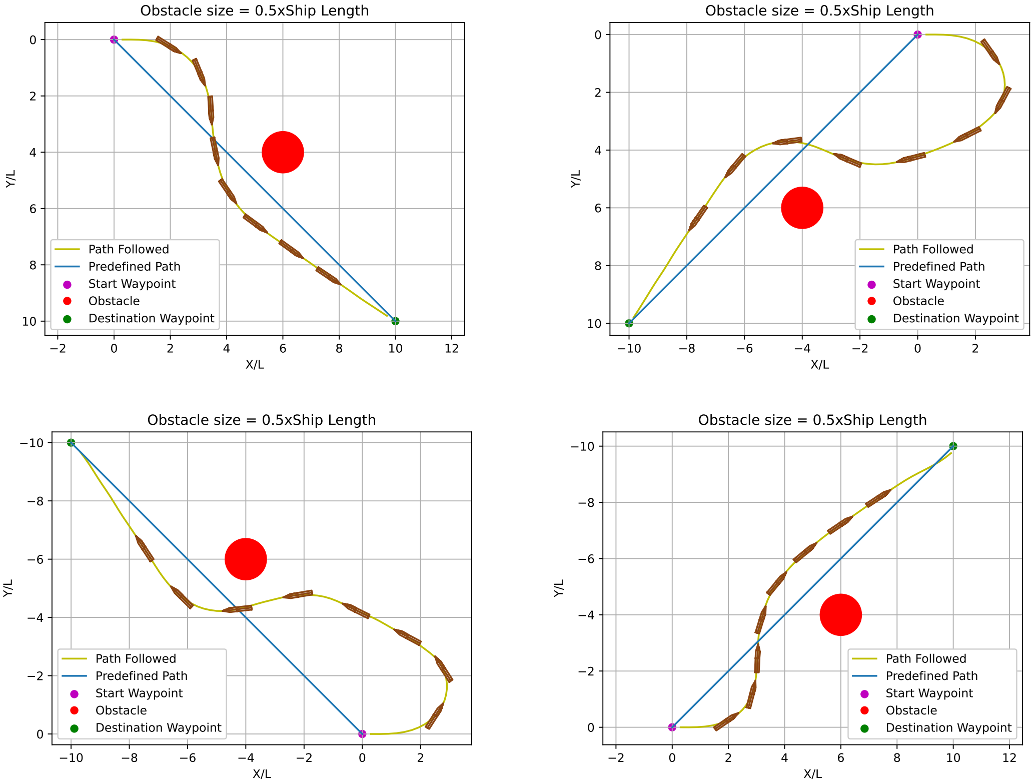


Figure 5.38: DQN agent successfully avoids obstacle of size 0.5L when present in path of ship

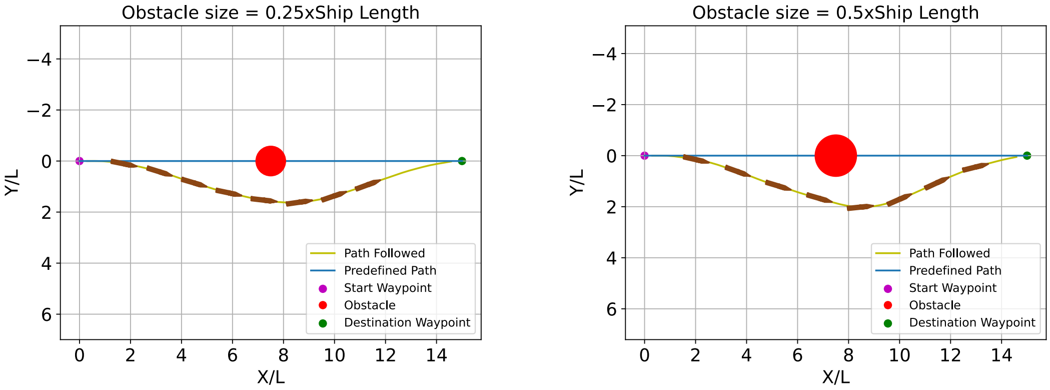


Figure 5.39: DQN agent successfully avoids obstacle on a straight line path

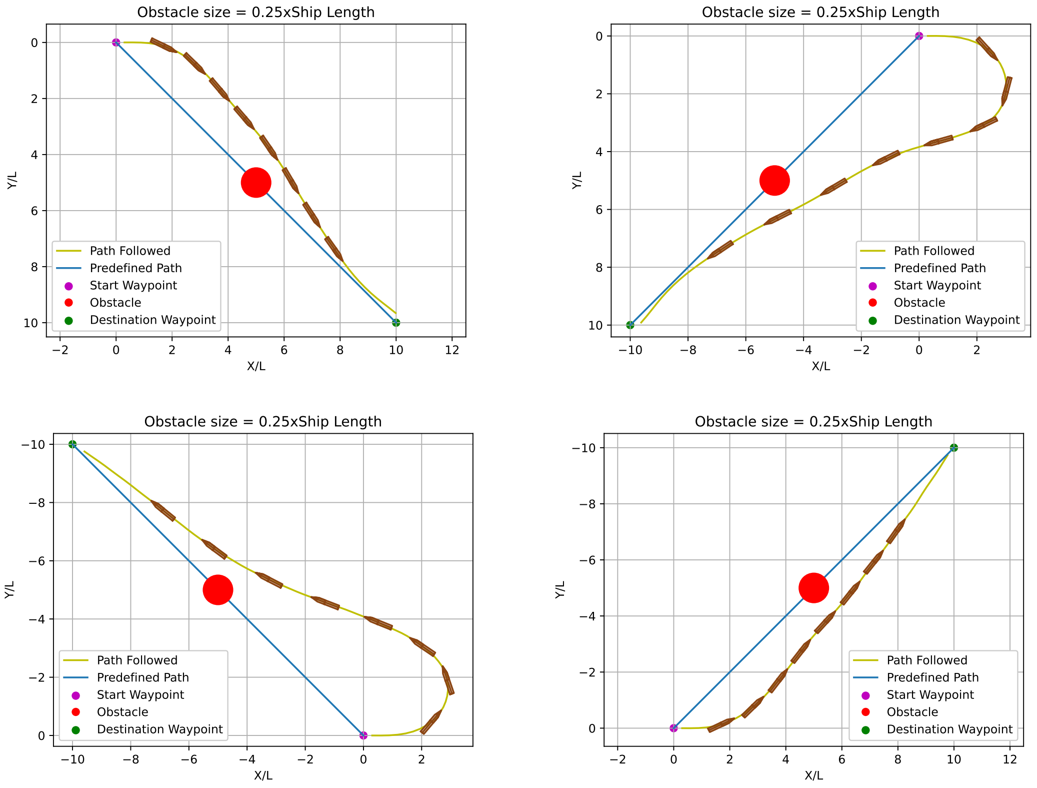


Figure 5.40: DDPG agent successfully avoids obstacle of size 0.25L when present on line joining the waypoints

Table 5.16: Comparison of agent when obstacle of size 0.5L is in is in path of the ship

| Maneuver | DQN | | DDPG | |
|---|---|---|---|---|
| | RMSE | C.E | RMSE | C.E |
| (10,10) | **0.8451** | 0.2839 | 0.8997 | **0.2539** |
| (-10,10) | 2.1433 | **0.3076** | **2.1273** | 0.3299 |
| (-10,-10) | 2.0908 | 0.3778 | **2.0598** | **0.2920** |
| (10,-10) | **0.8057** | 0.2794 | 0.9201 | **0.2185** |

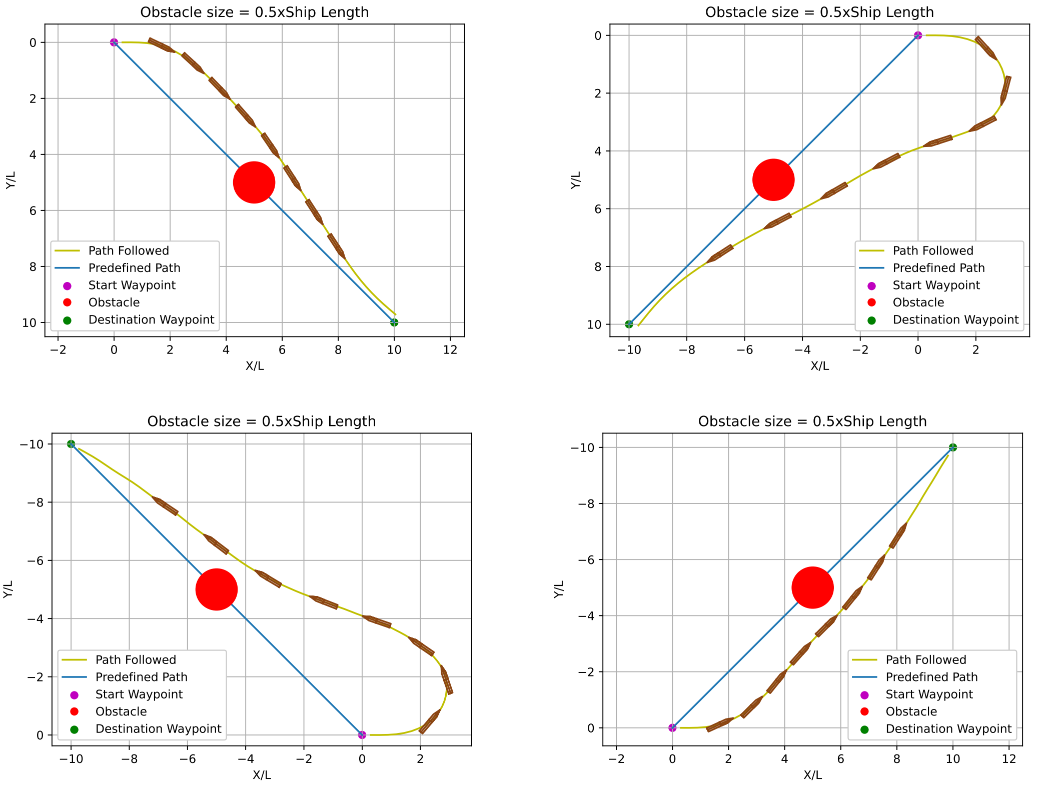


Figure 5.41: DDPG agent successfully avoids obstacle of size 0.5L when present on line joining the waypoints

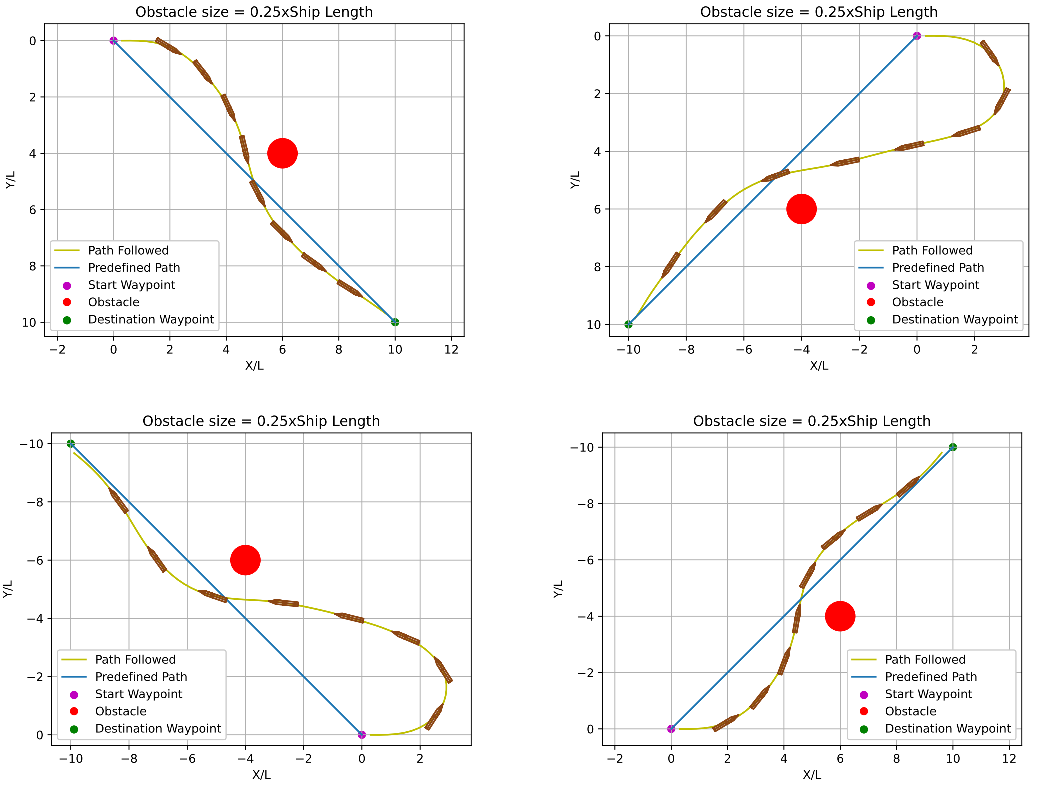


Figure 5.42: DDPG agent successfully avoids obstacle of size 0.25L when present in path of ship

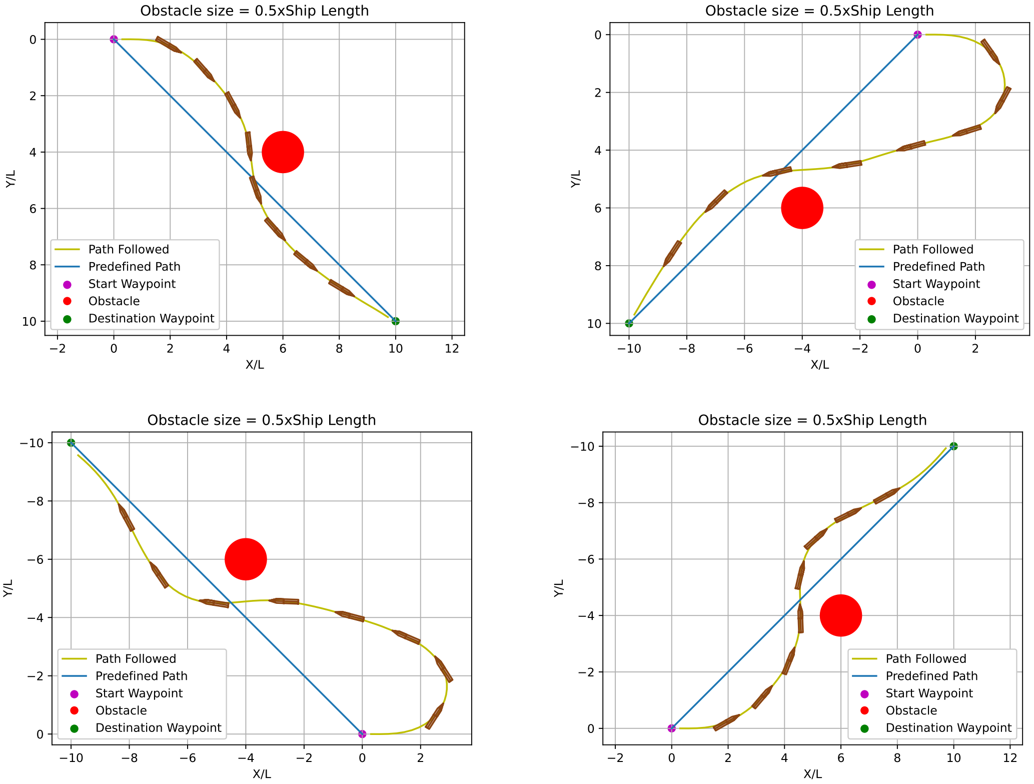


Figure 5.43: DDPG agent successfully avoids obstacle of size 0.5L when present in path of ship

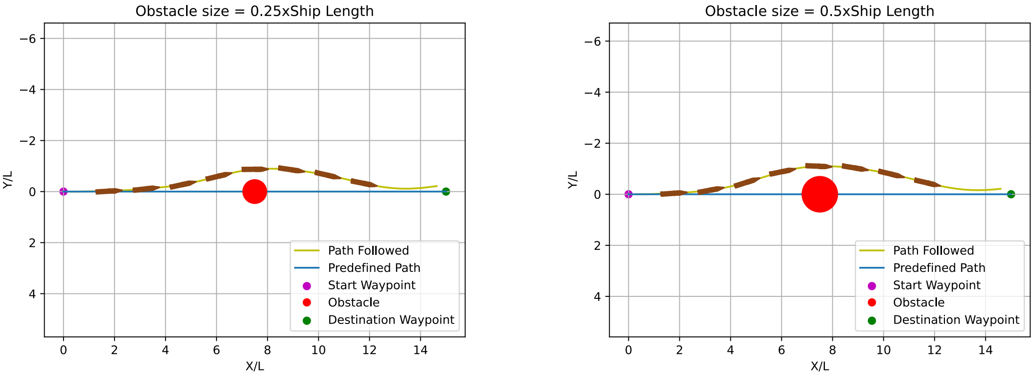


Figure 5.44: DDPG agent successfully avoids obstacle on a straight line path

DDPG agent is outperforming the DQN agent as the prior has less RMSE as well as less controller effort for the trajectory.

## 5.5 DYNAMIC OBSTACLE AVOIDANCE

In this subsection, DQN and DDPG-based controllers are trained to avoid dynamic obstacles as mentioned in Sec. 4.3.

### 5.5.1 Hyperparameters of the network

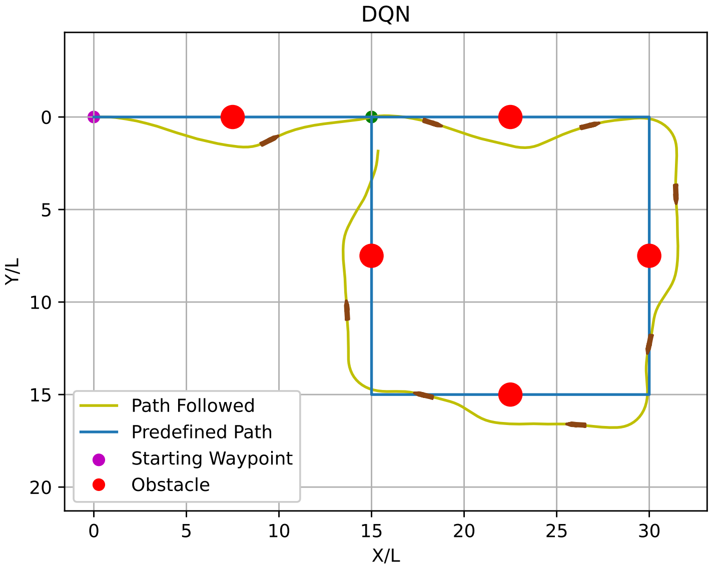
The agent is calibrated for satisfactory waypoint tracking by tuning the hyperparameters. The agent is then tested by simulating various maneuvers for waypoint tracking. The learning rate is chosen as an exponentially decaying function. The hyperparameters for the DQN and DDPG are shown in Table 5.17 and 5.18. The DQN agent trained for 8000 episodes is choosen for the task and 6000 episodes for DDPG agent. The returns vs episodes and loss vs episodes plots for DQN and DDPG are shown in Fig. 5.46 and 5.47.

It's important to note that reducing the loss value to zero may not be possible or necessary. The overall goal of a DRL is to learn an optimal policy that maximizes the long-term reward, not to minimize the loss value. As long as the agent is able to learn an effective policy that avoids obstacles and maximizes the reward, the loss value can be considered acceptable.

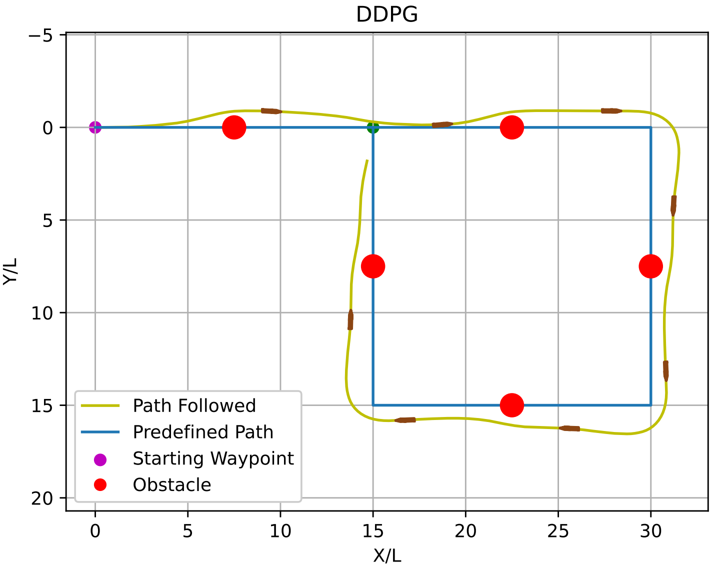
### 5.5.2 Obstacle Avoidance results

The results depicted in Link 1, Link 2, Link 3 and Link 4 shows that the DQN controller was successful in avoiding an obstacle in different cases with different size of obstacle and velocity of the obstacle.

The results depicted in Link 5, Link 6, Link 7 and Link 8 shows that the DDPG controller was successful in avoiding an obstacle in different cases with different size of obstacle and velocity of the obstacle.



(a) $d_{RMSE} = 1.0988L$
C.E = 0.3492 rad



(b) $d_{RMSE} = 0.9475L$
C.E = 0.2176 rad

Figure 5.45: Multiple Static Obstacle Avoidance

Table 5.17: Hyperparameters for DQN model

| **Hyperparameter** | **Value** |
|---|---|
| Initial Learning rate | 0.00075 |
| Decay Steps | 50000 |
| Decay Rate | 0.5 |
| Hidden layers | (128,128) |
| Discount factor($\gamma$) | 0.97 |
| Sample batch size | 128 |
| Replay buffer size | 100000 |
| Activation function | tanh |
| Maximum time steps | 160 |
| Time step interval $\Delta$t | 0.3 |
| Number of episodes | 8000 |
| Update frequency(time steps) | 5 |
| Time step interval $\Delta$t | 0.3 |
| Target network update frequency(time steps) | 1 |
| Target update rate ($\tau$) | 0.01 |

Table 5.18: Hyperparameters for DDPG model

| **Hyperparameter** | **Value** |
|---|---|
| Actor Initial Learning Rate | 0.0001 |
| Actor Decay Steps | 60000 |
| Actor Decay Rate | 0.4 |
| Critic Initial Learning Rate | 0.001 |
| Critic Decay Steps | 60000 |
| Critic Decay Rate | 0.4 |
| Actor Hidden Layers | (128,128) |
| Critic Hidden Layers | (128,128) |
| Discount factor($\gamma$) | 0.98 |
| Sample batch size | 128 |
| Replay buffer size | 100000 |
| Maximum time steps | 160 |
| Noise Multiplier | 0.15 |
| Noise Episodes | 11000 |
| Time step interval $\Delta$t | 0.3 |
| Update frequency(time steps) | 5 |
| Target network update frequency(time steps) | 1 |
| Target update rate ($\tau$) | 0.02 |

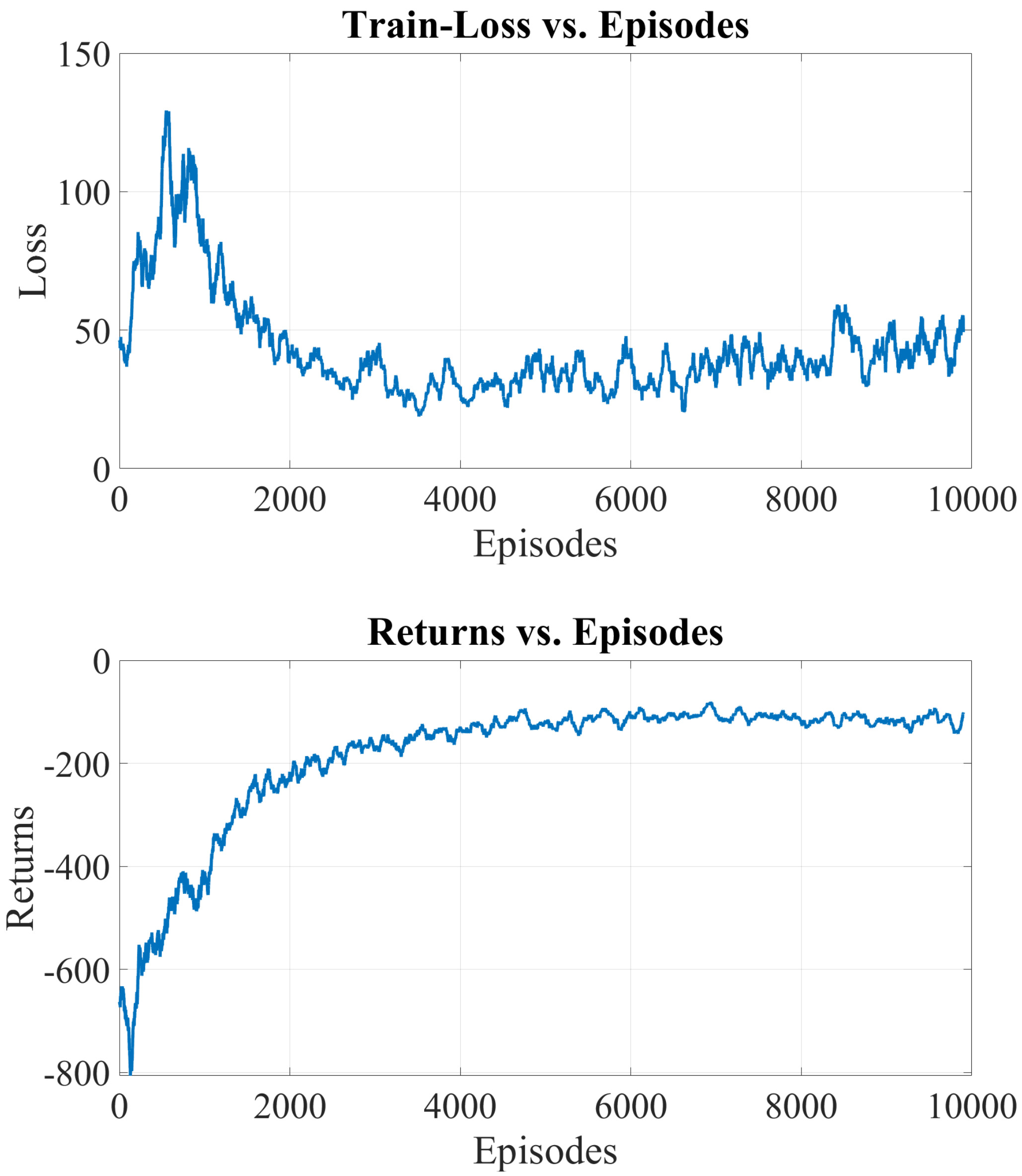


Figure 5.46: Training returns for DQN dynamic obstacle model

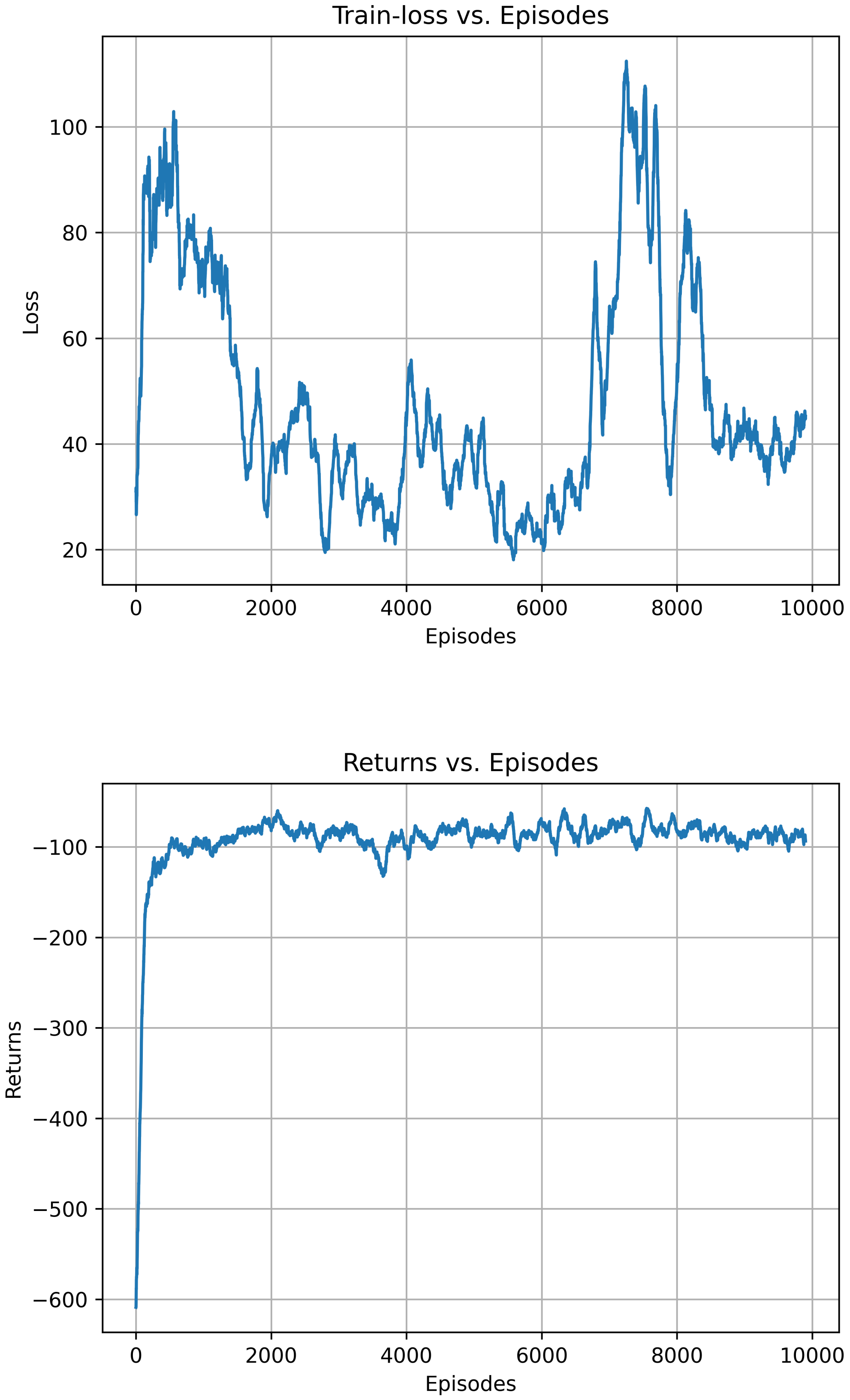


Figure 5.47: Training returns for DDPG dynamic obstacle model

Table 5.19: Comparison of agent in case of dynamic obstacle

| Maneuver | DQN | | DDPG | |
|---|---|---|---|---|
| | RMSE | C.E | RMSE | C.E |
| Head on | 1.136 | 0.323 | **0.8712** | **0.245** |
| Velocity along same direction | **1.126** | 0.325 | 1.167 | **0.166** |
| Starboard Side | **0.5186** | 0.3305 | 0.587 | **0.2204** |
| Port Side | **0.772** | **0.305** | 1.1034 | 0.3979 |

In the testing of DQN and DDPG controllers for autonomous ship control, four different scenarios were considered - starboard side obstacle, portside obstacle, head-on obstacle, and along the same direction velocity of obstacle. The performance of both controllers was evaluated based on two key metrics: RMSE of cross-track error and Controller Effort (CE). The results indicated that both controllers performed well in most scenarios. However, the DDPG controller exhibited a lower CE in most cases, indicating that it required less effort to navigate the ship through the obstacles. On the other hand, the DQN controller had a lower RMSE of cross-track error in most scenarios, which implies that it was more effective at keeping the ship on course. It is noteworthy that the choice of controller would ultimately depend on the specific requirements of the autonomous ship, such as the nature of the obstacle and the level of precision required in navigation. Nevertheless, these results provide valuable insights into the strengths and limitations of the DQN and DDPG controllers, which can be used to optimize autonomous ship control in real-world scenarios.

# CHAPTER 6

# DISCUSSION

## 6.1 WAYPOINT TRACKING

This section covers the noteworthy characteristics of the data-driven RL controllers exhibited in the previous sections.

In order to assess the performance of the RL agent in controlling the vessel under windy conditions, various wind speeds and directions were simulated for a straight-line maneuver with the vessel starting at $(0, 0)$ and $\psi = 0$, while attempting to follow a straight line along the x-axis. Fig. 6.1 presents the $d_{RMSE}$ ratio for two wind speeds and nine wind directions in comparison to calm water conditions. The results indicate that the RL agent can effectively track waypoints for all wind directions with an error of less than 20% compared to calm water conditions when the wind speed is low ($V_w = 6U$). However, for a wind speed of $V_w = 9U$, the wind directions of ±90° show greater deviation compared to calm water conditions. Furthermore, Fig. 5.26 demonstrates that the maximum deviation occurs at the beginning of the episode when the wind is perpendicular to the vessel, causing significant deviation before the vessel can regain its course and accurately follow the path.

The figure displayed in Figure 5.12d indicates that in the presence of high wind speeds ($V_w = 9U$) while attempting to follow a straight line path, there is a 10% variation in $d_{RMSE}$ when comparing a wind heading direction of $-90°$ with $+90°$. The difference in performance can be attributed to the non-symmetrical rudder dynamics caused by the asymmetry in flow past the rudder when applying port and starboard rudder angles. This effect is taken into account in the rudder dynamics (as discussed in Section 2.3). The effect is not significant for moderate wind speeds of $V_w = 6$, but becomes more pronounced for higher wind speeds of $V_w = 9$, where the wind forces and moments are stronger.

### 6.1.1 Yaw rate

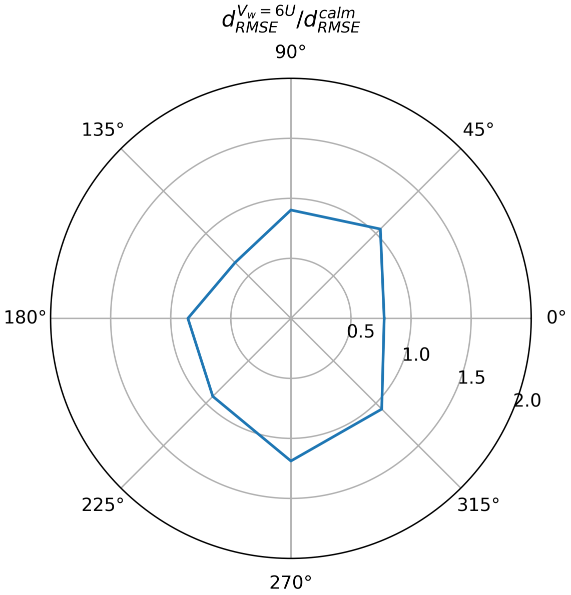
The ship's observation state vector includes the yaw rate '$r$' which has a significant influence on the ship's trajectory, despite not contributing to the reward. To analyze this effect, an agent is trained separately by removing the yaw rate from its observation state, while keeping all other hyperparameters constant between the models, as stated in Table 5.1. The tracked trajectory of the agent without the yaw rate is shown in Fig. 6.2. The results demonstrate that the agent with the yaw rate, as depicted in Fig. 5.2, performs better than the agent without it. This implies that the yaw rate is a crucial factor for the agent to consider when making decisions regarding ship control, despite its non-contribution to the reward.

### 6.1.2 Reward Function

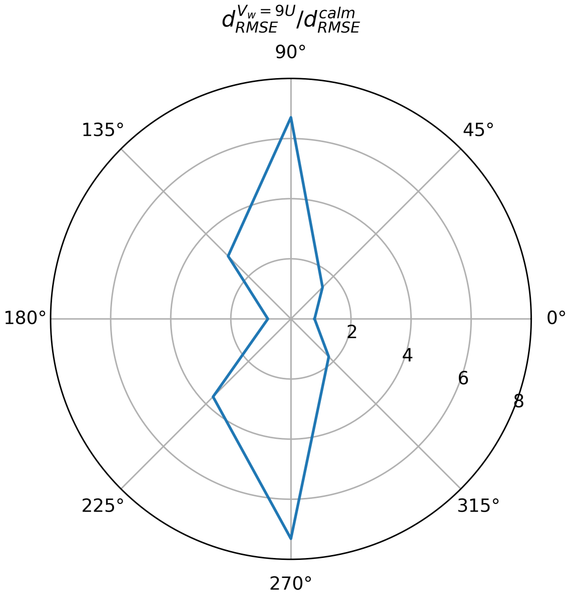
The reward at each time step should be negative because it drives the agents to reach the destination as soon as possible. Since, the RL agent wants to maximise the reward at each episode, if the agent has a negative reward at each timestep, the cumulative sum of the reward at the end of the episode will be negative and hence the agent tries to finish the episode soon. After training with the same hyperparameters as given in Table 5.1 and changing the reward function as :

$$\begin{aligned} r_1 &= 2\exp\left(\frac{-d_c^2}{12.5}\right) - 1 + 2 \\ r_2 &= 1.3\exp\left(-10\,|\chi_e|\right) - 0.3 + 2 \\ r_3 &= 0 \end{aligned} \tag{6.1}$$

The rewards $r_1$ and $r_2$ are now positive at each time step and hence the overall reward at each timestep will become positive. Fig. 6.3 shows the results of the modified reward



(a) For wind speed $V_w = 6U$



(b) For wind speed $V_w = 9U$

Figure 6.1: Relative cross track error for straight line maneuver with wind speeds $V_w = 6, 9$

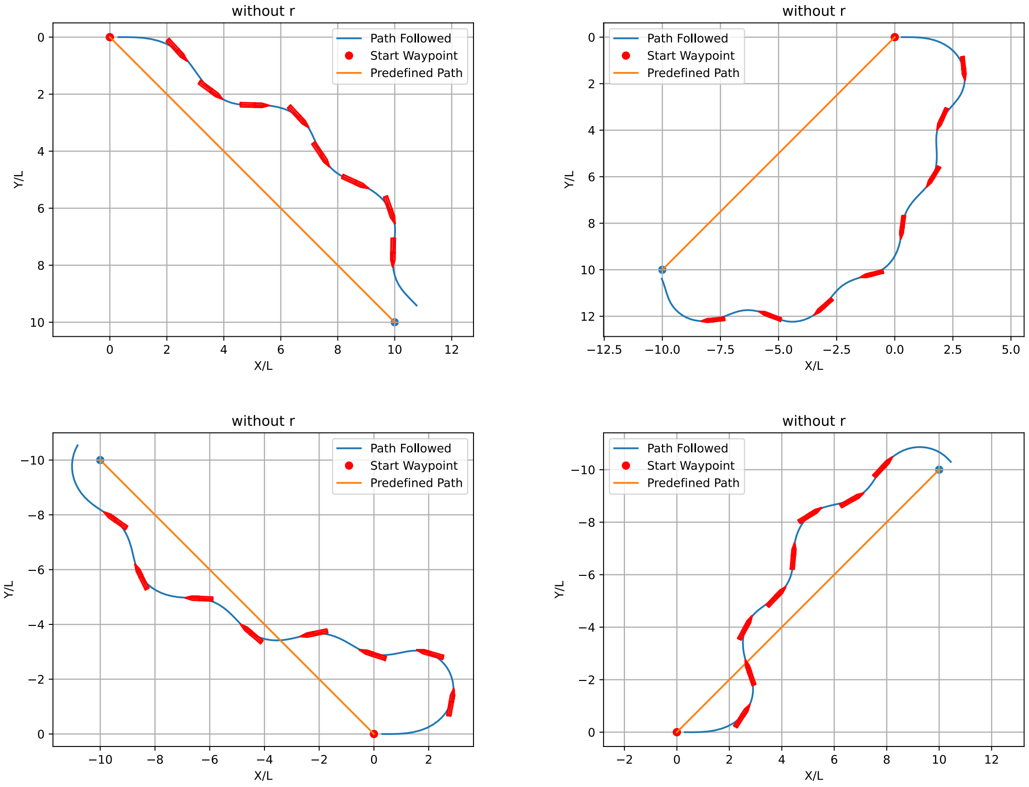


Figure 6.2: Without yaw rate

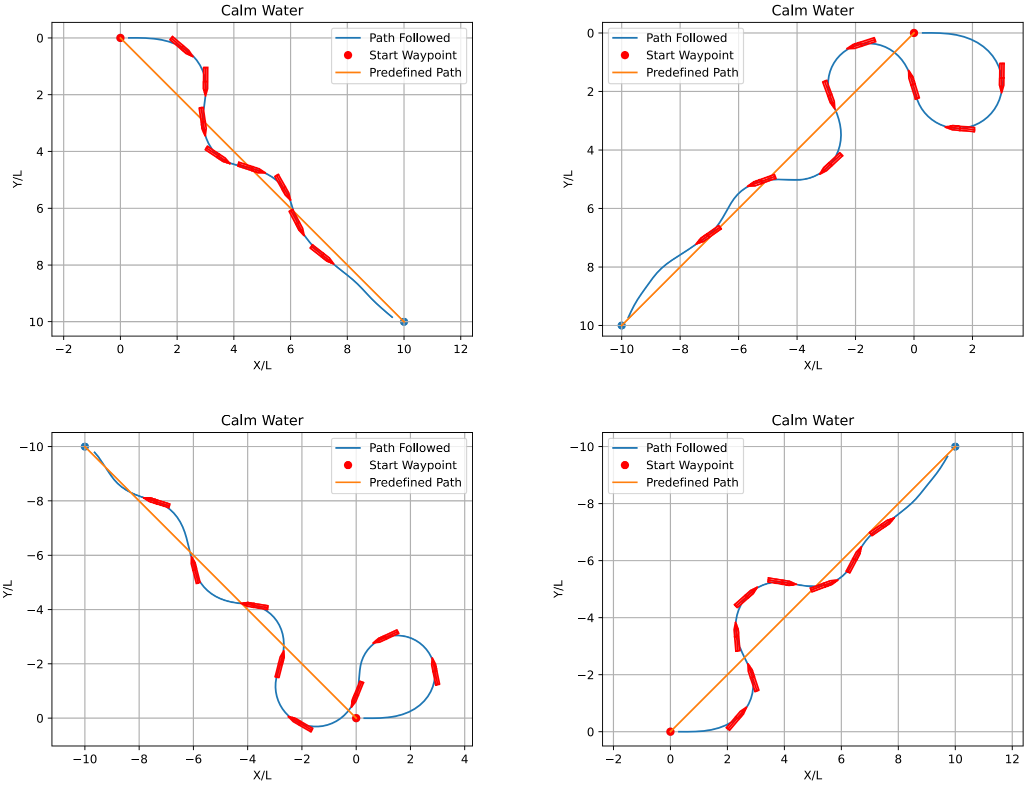


Figure 6.3: Modified Rewards

function on the training. It can be seen that the agent is still reaching the destination to get a +100 reward, but the path followed by the agent has become large as the agent want to accumulate more positive reward. Since the agent always has a velocity (due to constant propeller r.p.m) the agent can not stay at a particular location and accumulate a reward. Therefore the agent follows this zig-zag trajectory to spend more time steps before reaching the destination and maximising the overall reward.

### 6.1.3 Effect of reward $r_3$

The inclusion of the term $\frac{-d_{wp}}{4}$ in the reward function (4.7) is intended to make the overall reward negative, which will encourage the agent to try to reach the destination as quickly as possible. The restrictions on the values of $r_1$ and $r_2$ are in place to ensure that the agent focuses on reducing the distance to the goal when it is far away but shifts its attention to the course angle error and cross-track error as it gets closer to the destination.

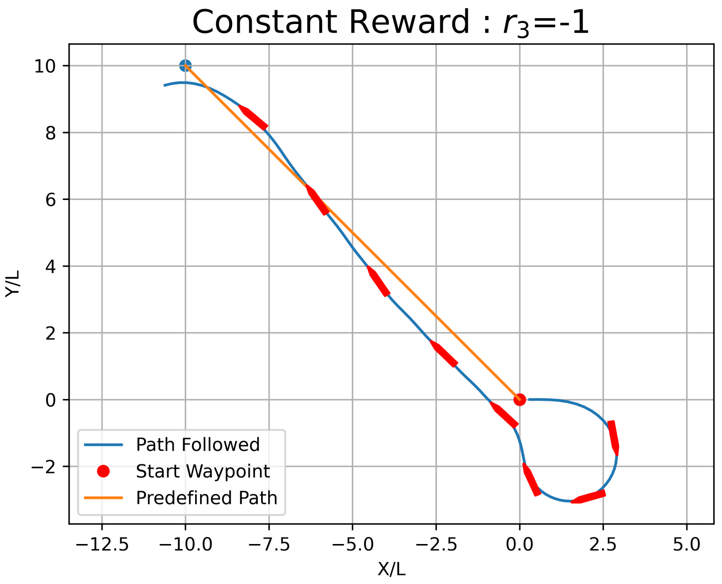


Figure 6.4: Constant Reward $r_3$

This is intended to produce a smooth trajectory that gradually decreases these errors as the distance to the goal decreases. It has been found that the specific values chosen for these terms, including the factor of 4 in the distance term, result in a path that effectively balances the importance of these different factors. Other values for these terms may produce different trade-offs between distance and angle errors.

Both $r_1$ and $r_2$ are bounded to ensure that far away from the goal, the distance should be the governing factor. Training the model with a constant negative $r_3$ does not give a smooth curve as what we get if there is a dependence of distance to the waypoint. Fig. 6.4 shows the trajectory of the agent with constant negative reward. The hyperparameters are the same as described in Table 5.1 along with the same seed number.

## 6.2 STATIC OBSTACLE AVOIDANCE

In conclusion, this study trained and evaluated a model for obstacle avoidance and waypoint tracking using DQN and DDPG controllers. The controllers were successful in performing these tasks, and the model's robustness was tested by placing an obstacle in the path of the ship. The comparison of the controllers in terms of RMSE and CE showed that DDPG outperformed DQN in cases where the obstacle was in between the line joining the waypoints, while DQN performed better in cases where the obstacle was in the path of the ship, especially when the obstacle was small. Overall, the difference in performance between the two controllers was not significant, and the study provides insights for selecting an appropriate controller based on the obstacle's size and position.

### 6.2.1 Hybrid Path Following and Obstacle Avoidance Architecture

During agent training with an obstacle present, the distance between the agent and the obstacle was restricted to between 25% and 75% of the distance between the starting and ending points. As a result, the agent was not trained when the obstacle was far away and was unable to optimize its path during training. Although the likelihood of a collision with the obstacle was low, the trajectory for the path following was not optimized, as illustrated in Fig. 6.5.

The agent can successfully reach the destination in certain scenarios if it detects obstacles at specific distances during training. However, this approach is not practical for generalizing to different environments, as obstacles can be located anywhere and training for each trajectory would be computationally expensive. Therefore, a hybrid architecture is proposed that incorporates both the trained network for the path following described in Sec. 5.3 and the obstacle avoidance network presented in Sec. 5.4.2. The path-following network will be utilized unless an obstacle with CR>0 is detected within a 5L distance from the ship. Fig. 6.6 shows the DDPG agent was successful in reaching the destination quite effectively.

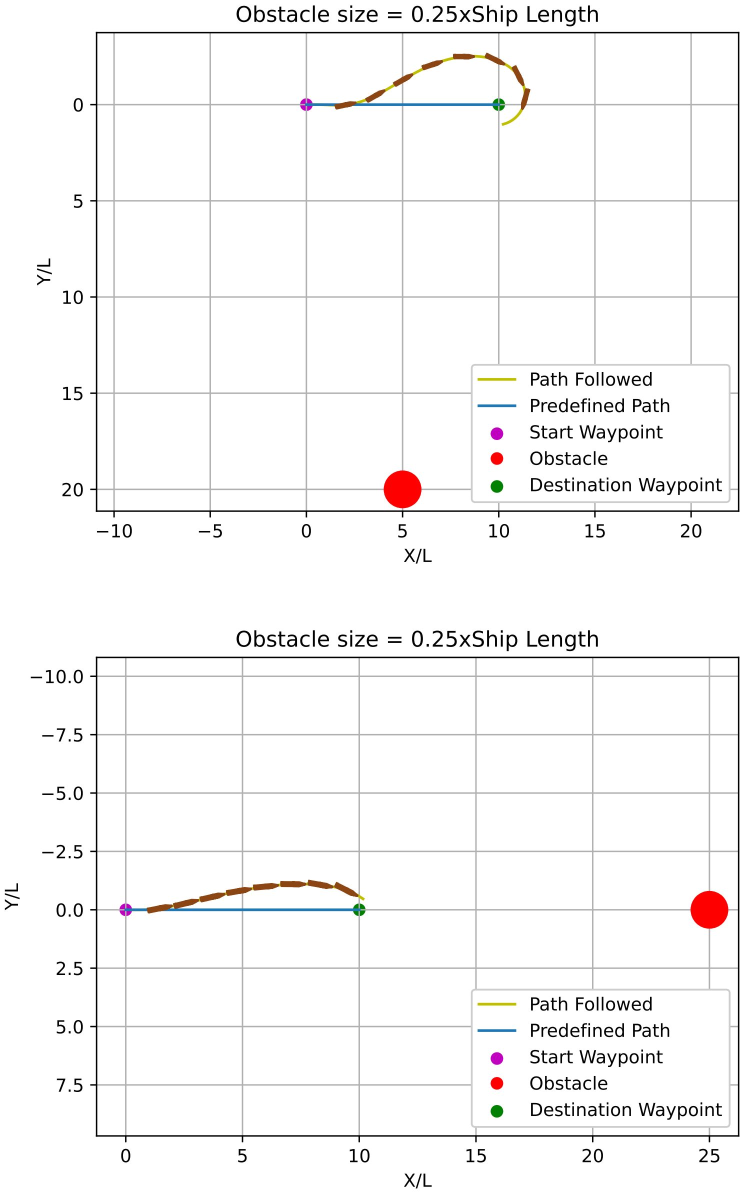


Figure 6.5: Agent unable to reach destination

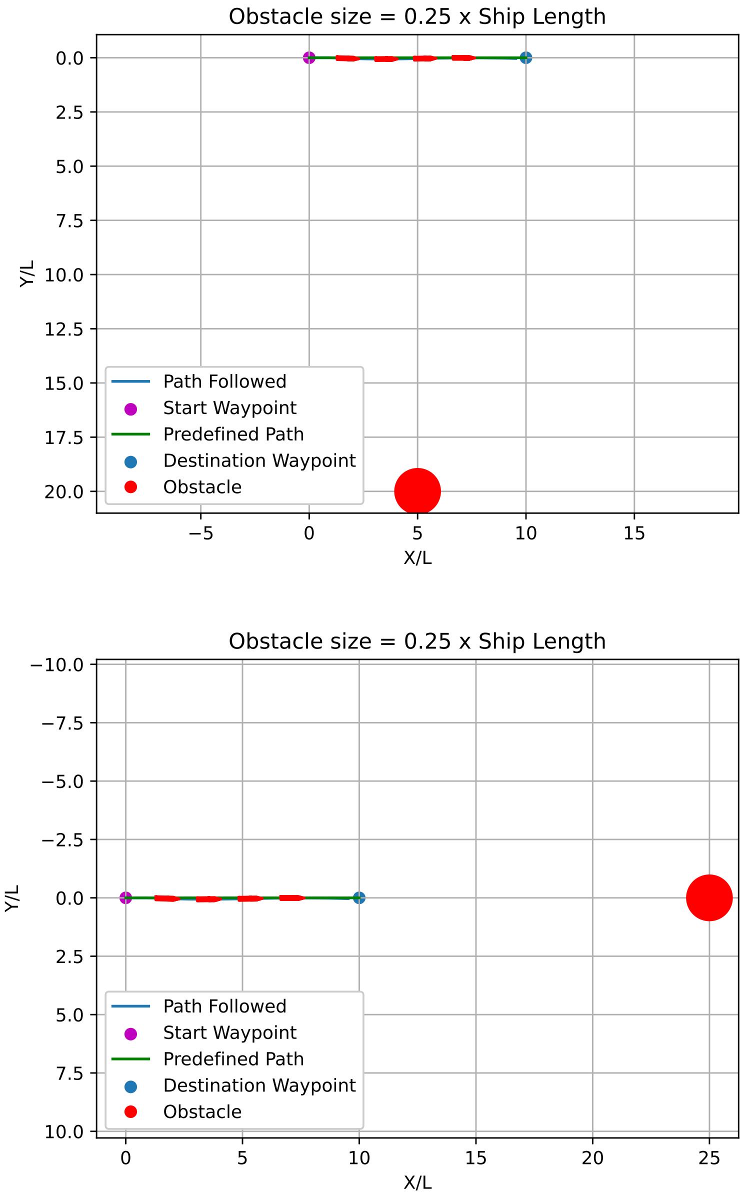


Figure 6.6: Hybrid architecture

# CHAPTER 7

# CONCLUSION AND FUTURE STUDIES

In conclusion, this research successfully developed a DRL-based controller for autonomous ship navigation, showcasing its effectiveness in path following, waypoint tracking, and compensation for wind forces. The DRL controller's performance was verified through simulations and validated with experiments. A comparison with the traditional PD controller revealed that the RL-based approach performed equally well or even better in certain cases.

The study evaluated multiple DRL algorithms, including DQN, DDPG, and PPO, and found them capable of effectively tracking waypoints and executing maneuvers in different trajectories, even in the presence of external disturbances. The PPO controller generally exhibited the least cross track error and controller effort.

The implications of the findings in the context of the maritime industry could be significant. By exploring and developing RL-based path planning and navigation methods for autonomous surface vehicles (ASVs), there is potential to improve the safety, efficiency, and operational costs in maritime operations. RL algorithms that can effectively handle complex and dynamic environments, adapt to changing conditions, and make optimal decisions could enhance the autonomous capabilities of ASVs, leading to safer and more efficient navigation. Moreover, the integration of RL with traditional methods could provide a balance between established techniques and innovative approaches, potentially improving the overall performance and practical applicability of autonomous navigation systems in the maritime industry.

Additionally, the research explored the training of DQN and DDPG controllers for handling both static and dynamic obstacles, yielding promising results. The DDPG controller outperformed the DQN controller for static obstacles, while both controllers successfully avoided dynamic obstacles. Moreover, a hybrid model utilizing multiple neural networks was introduced, allowing for effective obstacle avoidance by dynamically switching between networks based on obstacle location.

Future research endeavors may focus on integrating dynamic obstacles and training reinforcement learning (RL) agents to adhere to Collision Regulations at Sea (COLREGS), posing a more significant challenge. However, several specific challenges and considerations arise when ensuring compliance with maritime regulations while employing RL techniques. One critical challenge involves developing RL algorithms that can effectively learn and encode the complex and context-dependent rules of COLREGS. Striking the right balance between exploration and exploitation is another challenge to ensure compliance without overly conservative behavior. Additionally, it is vital to verify and validate RL-based systems' adherence to COLREGS, demonstrating their safety and reliability to regulatory authorities and stakeholders. Enhancing the system can be achieved by conducting comparisons with traditional controllers that utilize path planning algorithms. It is important to note that the current research assumed static and dynamic obstacles with predetermined sizes and velocities. However, real-world scenarios may involve encounters with obstacles of varying sizes and speeds. Furthermore, the maximum rudder rate ($\dot{\delta}_{max}$) in the study is 5° per second, which may overload the Steering Gear and require significant powering at full scale. To address this, the rudder rate can be limited to 1.5 - 2.5° per second. Therefore, future studies should encompass evaluating the RL agent's performance under such diverse conditions.

To validate the practical applicability of the control algorithms, further steps involve testing and verification on an actual unmanned surface vessel for collision avoidance. This real-world experimentation will provide valuable insights into the RL agent's performance. Furthermore, while the simulations demonstrated satisfactory performance, future research should also explore the behavior of the RL agent under additional environmental factors, such as currents and wave forces, to ensure its robustness and reliability in diverse conditions.

Finally, there are potential limitations and risks associated with relying solely on RL-based controllers for autonomous surface vehicles. A major concern is the lack of transparency and interpretability in RL algorithms, which makes it challenging to comprehend and clarify how the controller arrives at its decisions. This lack of understanding undermines trust and poses obstacles in guaranteeing the system's safety and dependability. Moreover, RL-based controllers may exhibit subpar performance when confronted with uncommon or novel scenarios that were not encountered during training, potentially resulting in unpredictable behavior and safety risks. Furthermore, integrating DRL-based controllers into existing maritime systems and ensuring compliance with regulatory frameworks necessitates careful deliberation and adjustment.

To effectively address these risks, it is imperative to establish a comprehensive safety framework that encompasses various measures. This framework should include rigorous testing and validation procedures, incorporating diverse and realistic simulation environments as well as extensive field trials. It is essential to integrate fail-safe mechanisms, such as human override capabilities or backup systems, to handle critical situations and ensure human intervention when necessary.

While RL-based controllers can bring significant benefits to autonomous surface vehicles (ASVs), relying solely on them presents potential limitations and risks. One notable limitation is the need for a substantial amount of training data and computational resources, which may not be practical or feasible to acquire and utilize in real-time scenarios. Additionally, RL algorithms can be sensitive to environmental changes and system dynamics, making them susceptible to failures or suboptimal behavior in unforeseen circumstances.

Furthermore, it is crucial to explore transfer learning techniques that leverage knowledge from related tasks to enhance the interpretability and explainability of the decision-making process of the RL-based controllers. These research avenues are vital for advancing the ongoing development and improvement of DRL-based controllers. As RL algorithms become increasingly complex and autonomous, it is essential to address potential biases and unintended consequences.

To mitigate biases, researchers are exploring techniques such as careful dataset curation, fairness-aware reward shaping, and algorithmic transparency. Efforts are also being made to develop explainable AI methods that provide insights into the decision-making process of RL algorithms. Moreover, establishing robust regulatory frameworks, standards, and guidelines for the deployment and operation of RL-based systems in the maritime industry is crucial to ensure accountability and responsible use of these technologies.

# APPENDIX A

# MAKING THE EPSILON GOES TO A CONSTANT VALUE WHICH IS GREATER THAN 0

While training the DQN model in Sec. 3.2, the epsilon (exploration factor) goes to zero at 5000 episodes, but in the DQN paper (Mnih *et al.* (2013)), the epsilon was constant for the whole episodes. This also tried to see the difference between the two methods of exploration and exploitation.

$$\epsilon = \begin{cases} 1 - \frac{\text{current episode number}}{5000} & \text{if } \epsilon \geq 0.2 \\ 0.2 & \text{else} \end{cases} \tag{A.1}$$

After using the same set of hyperparameters along with the same seed number, both agents were compared against each other. The training losses and episode returns averaged over 100 episodes are shown in Fig. A.1. For comparing the model, we are calculating the RMSE value of the cross-track error for different trajectories.

In the case of Deep Q-Networks (DQN), the choice of exploration strategy can have a significant impact on the learning process. When the value of epsilon does not go to zero, the agent is more likely to take random actions, which can help the agent to

Table A.1: Comparison of the DQN controller

| **Trajectory** | **Original($\epsilon$ goes to 0)** | **Modified($\epsilon$ goes to 0.2)** |
|---|---|---|
| Circle (radius = 6L) | **0.6095L** | 0.7142L |
| Circle (radius = 12L) | 0.5395L | **0.4632L** |
| Ellipse | **0.2969** | 0.344L |
| Eight | **0.4393L** | 0.4557L |
| Ellipse ($V_w = 6$ and $\beta_w = 0°$) | 0.8456L | **0.5640L** |
| Eight($V_w = 6$ and $\beta_w = 45°$) | 0.3941L | **0.3485L** |

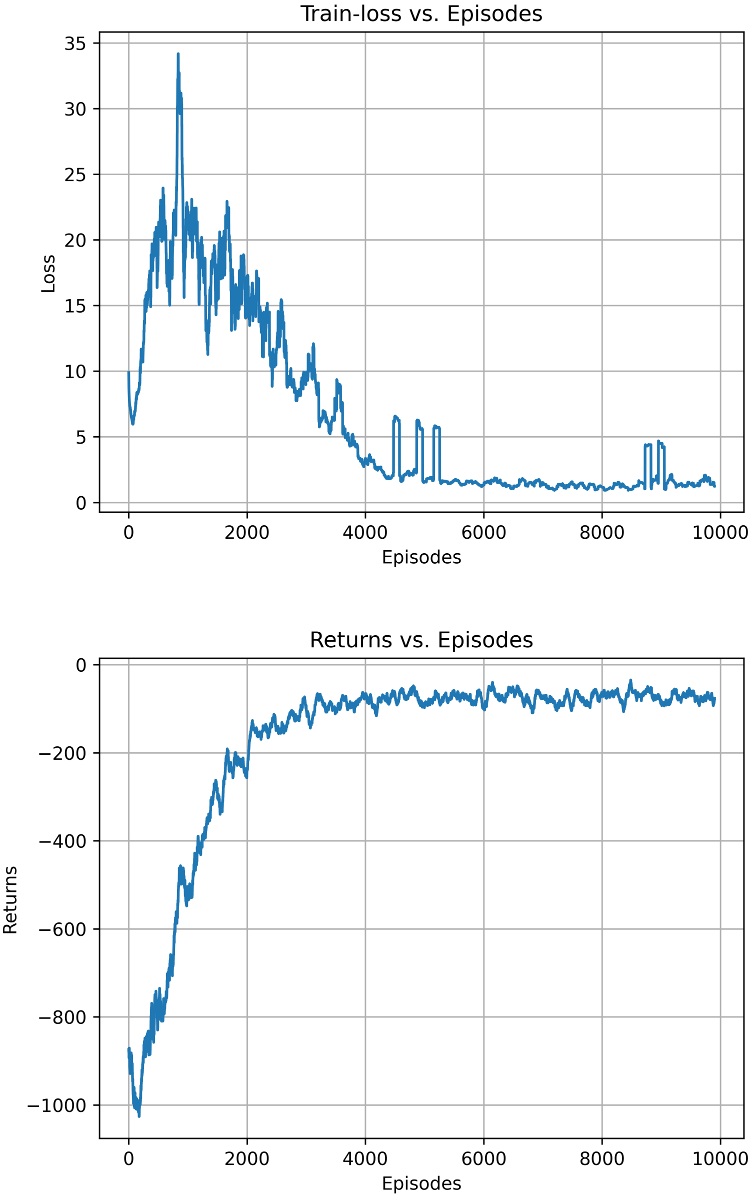


Figure A.1: Training loss and returns for DQN model

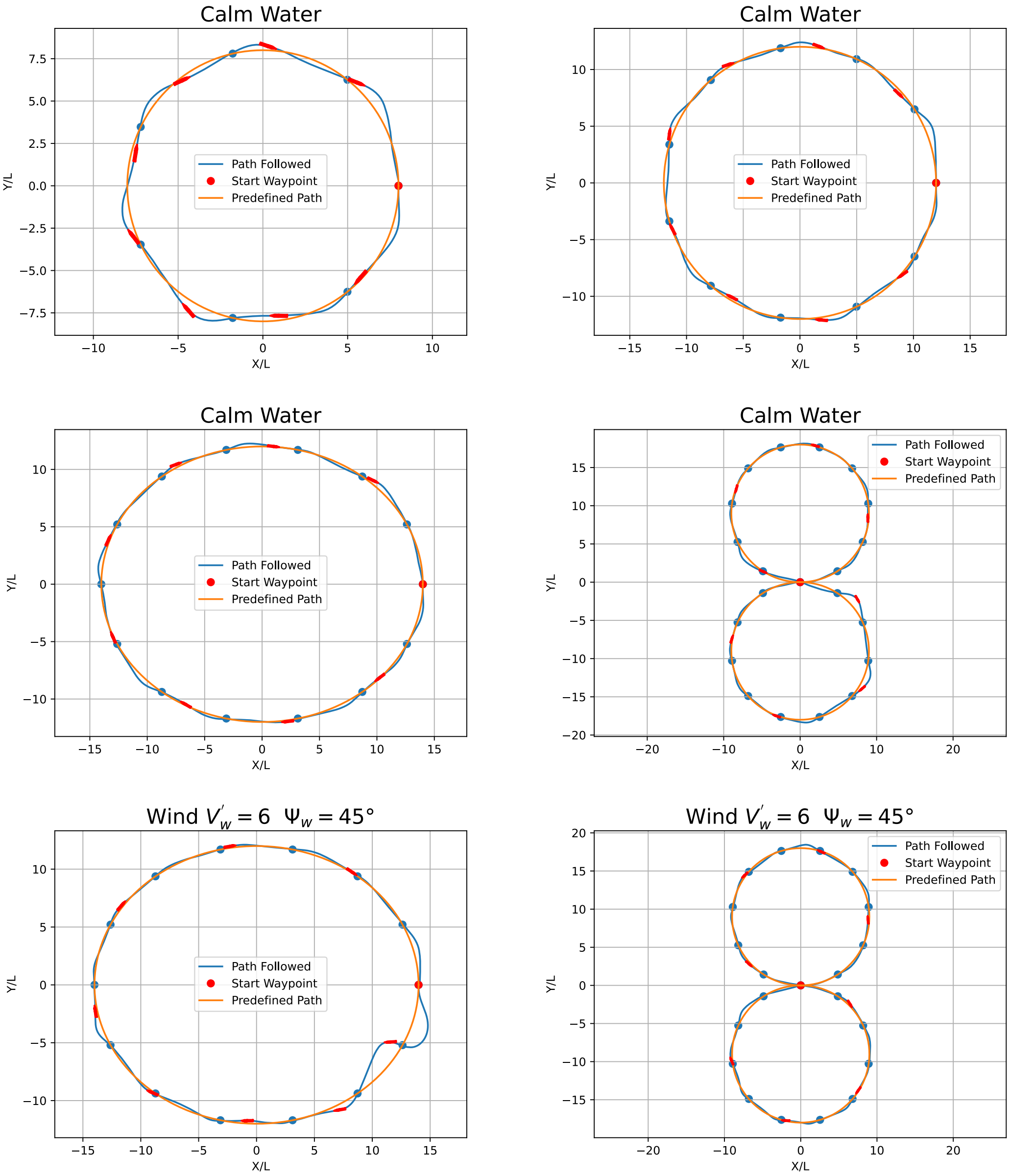


Figure A.2: Path Following of the DQN model

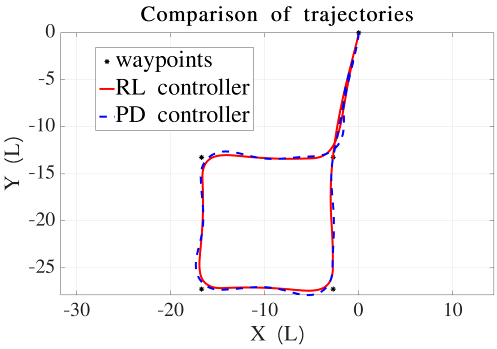
explore different states and learn from new experiences. This can be particularly useful in environments with external disturbances, such as the presence of winds. In such situations, the agent needs to be able to adapt to changes in the environment, and random exploration can help it to do so. On the other hand, when the value of epsilon goes to zero after a certain number of episodes, the agent relies more on its learned policy and takes more deterministic actions. This can lead to better performance in calmer environments, where the agent can exploit its learned policy more effectively. Ultimately, the choice of exploration strategy will depend on the specific characteristics of the environment, and a balance between exploration and exploitation needs to be struck to achieve optimal performance.

# APPENDIX B

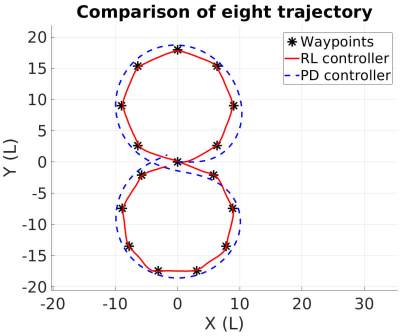
# TUNING OF THE PD CONTROLLER

The comparison between the DQN controller and the PD controller with ILOS was conducted, and the results showed that the DQN controller initially outperformed the PD controller as shown in Sec. 5.2.3. However, before that the study used another set of values of $K_p$ and $K_d$ set at 1.7 and 4.0, respectively. It can be seen from the Fig. B.1a that the RL controller performance is similar to the PD controller with the root mean square cross track error being marginally (4%) smaller for the DRL controller. Further simulations were performed on an ellipse and eight trajectory and it was found that the root mean square cross track error for PD controller is 5 times the corresponding value from DRL controller in case of eight trajectories (Fig. B.1b). This same value is about 4 times for the ellipse trajectory (Fig. B.1c).

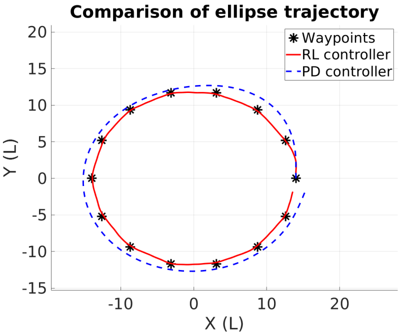
Overall, the results suggest that the choice of the controller may depend on the specific trajectory and task at hand, and the PD controller can be tuned for specific trajectories to achieve superior performance. Future studies can explore the optimization of controllers for different trajectories and tasks to enhance performance further.



(a) Square Trajectory (RL: $d_{RMSE} = 0.6693L$ PD: $d_{RMSE} = 0.6959L$)



(b) Eight Trajectory (RL: $d_{RMSE} = 0.369L$ PD: $d_{RMSE} = 1.990L$)



(c) Ellipse Trajectory (RL: $d_{RMSE} = 0.246L$ PD: $d_{RMSE} = 1.019L$)

Figure B.1: Comparison of the path traversed by the model in PD controller and RL controller in simulation

# APPENDIX C

# HYBRID ARCHITECTURE OF DRL AND PD BASED CONTROLLER

The control system for a vessel's movement can be implemented using a hybrid architecture of PD and DRL-based controller. This approach combines classical control theory and reinforcement learning techniques. The PD controller is responsible for ensuring stable and predictable movement when there are no obstacles. However, when the system detects an obstacle nearby, the DRL controller takes over and uses a model-free approach to learn how to navigate around the obstacle. This hybrid architecture has a faster output speed compared to using just one controller. Although the PD controller is faster than the RL controller in waypoint tracking (the computation speed for one control decision of the RL based controller is 0.110 ms while for a PD controller is 0.0212 ms.), it requires an online path planner (ex. Velocity Obstacle Method, Artificial Potential Field, A* algorithm, etc) to handle obstacle avoidance, which consumes more computation power. On the other hand, the RL controller can take observation state data and output control actions needed to avoid obstacles. Once the obstacle is passed, the system switches back to the PD controller. This hybrid architecture takes advantage of both PD control and DRL, resulting in a more robust and adaptive control system capable of handling various real-world scenarios. Fig. C.1 illustrates the plot for the hybrid architecture.

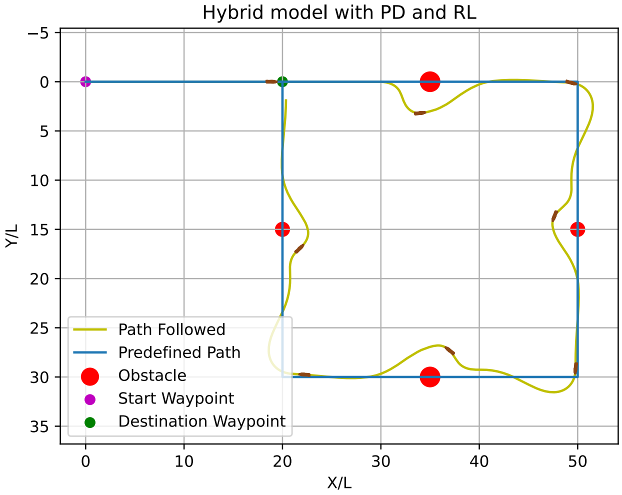


Figure C.1: Hybrid architecture of PD and DRL controller

# BIBLIOGRAPHY

1. **Abadi, M.**, **A. Agarwal**, **P. Barham**, **E. Brevdo**, **Z. Chen**, **C. Citro**, **G. S. Corrado**, **A. Davis**, **J. Dean**, **M. Devin**, **S. Ghemawat**, **I. Goodfellow**, **A. Harp**, **G. Irving**, **M. Isard**, **Y. Jia**, **R. Jozefowicz**, **L. Kaiser**, **M. Kudlur**, **J. Levenberg**, **D. Mané**, **R. Monga**, **S. Moore**, **D. Murray**, **C. Olah**, **M. Schuster**, **J. Shlens**, **B. Steiner**, **I. Sutskever**, **K. Talwar**, **P. Tucker**, **V. Vanhoucke**, **V. Vasudevan**, **F. Viégas**, **O. Vinyals**, **P. Warden**, **M. Wattenberg**, **M. Wicke**, **Y. Yu**, and **X. Zheng** (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

2. **Baker, C.** and **D. McCafferty**, Accident database review of human element concerns: What do the results mean for classification? *In Proc. Int Conf.'Human Factors in Ship Design and Operation, RINA Feb*. Citeseer, 2005.

3. **Chen, C.**, **X.-Q. Chen**, **F. Ma**, **X.-J. Zeng**, and **J. Wang** (2019). A knowledge-free path planning approach for smart ships based on reinforcement learning. *Ocean Engineering*, **189**, 106299.

4. **Chen, Y.**, **H. Peng**, and **J. Grizzle** (2017). Obstacle avoidance for low-speed autonomous vehicles with barrier function. *IEEE Transactions on Control Systems Technology*, **26**(1), 194–206.

5. **Chun, D.-H.**, **M.-I. Roh**, **H.-W. Lee**, **J. Ha**, and **D. Yu** (2021). Deep reinforcement learning-based collision avoidance for an autonomous ship. *Ocean Engineering*, **234**, 109216.

6. **Deogaonkar, V.**, **A. Jadhav**, **K. R**, and **A. Somayajula**, Data driven identification of ship maneuvering coefficients. *In International Conference on Offshore Mechanics and Arctic Engineering*. American Society of Mechanical Engineers, 2023.

7. **Fossen, T. I.** (1999). Guidance and control of ocean vehicles. *University of Trondheim, Norway, Printed by John Wiley & Sons, Chichester, England, ISBN: 0 471 94113 1, Doctors Thesis*.

8. **Fossen, T. I.**, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

9. **Fossen, T. I.**, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2021.

10. **Fox, D.**, **W. Burgard**, and **S. Thrun** (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, **4**(1), 23–33.

11. **Garrido, S.**, **L. Moreno**, **M. Abderrahim**, and **F. Martin**, Path planning for mobile robot

navigation using voronoi diagram and fast marching. *In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006.

12. **Guo, S.**, **X. Zhang**, **Y. Zheng**, and **Y. Du** (2020). An autonomous path planning model for unmanned ships based on deep reinforcement learning. *Sensors*, **20**(2), 426. ISSN 1424-8220.

13. **Heiberg, A.**, **T. N. Larsen**, **E. Meyer**, **A. Rasheed**, **O. San**, and **D. Varagnolo** (2022). Risk-based implementation of colregs for autonomous surface vehicles using deep reinforcement learning. *Neural Networks*, **152**, 17–33. ISSN 0893-6080.

14. **Layek, A.**, **N. A. Vien**, **T. Chung**, *et al.*, Deep reinforcement learning algorithms for steering an underactuated ship. *In 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2017.

15. **Lazarowska, A.** (2020). A discrete artificial potential field for ship trajectory planning. *The Journal of Navigation*, **73**(1), 233–251.

16. **Lekkas, A. M.** and **T. I. Fossen** (2012). A time-varying lookahead distance guidance law for path following. *IFAC Proceedings Volumes*, **45**(27), 398–403.

17. **Lillicrap, T. P.**, **J. J. Hunt**, **A. Pritzel**, **N. Heess**, **T. Erez**, **Y. Tassa**, **D. Silver**, and **D. Wierstra** (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971.*

18. **Lyu, H.** and **Y. Yin** (2019). Colregs-constrained real-time path planning for autonomous ships using modified artificial potential fields. *The Journal of navigation*, **72**(3), 588–608.

19. **Martinsen, A. B.** and **A. M. Lekkas**, Curved path following with deep reinforcement learning: Results from three vessel models. *In OCEANS 2018 MTS/IEEE Charleston*. IEEE, 2018.

20. **Mnih, V.**, **K. Kavukcuoglu**, **D. Silver**, **A. Graves**, **I. Antonoglou**, **D. Wierstra**, and **M. Riedmiller** (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602.*

21. **Moreira, L.**, **T. I. Fossen**, and **C. G. Soares** (2007). Path following control system for a tanker ship model. *Ocean Engineering*, **34**(14-15), 2074–2085.

22. **Mou, J. M.**, **C. Van Der Tak**, and **H. Ligteringen** (2010). Study on collision avoidance in busy waterways by using ais data. *Ocean Engineering*, **37**(5-6), 483–490.

23. **Perera, L. P.**, **V. Ferrari**, **F. P. Santos**, **M. A. Hinostroza**, and **C. Guedes Soares** (2015). Experimental evaluations on ship autonomous navigation and collision avoidance by intelligent guidance. *IEEE Journal of Oceanic Engineering*, **40**(2), 374–387.

24. **Schulman, J.**, **S. Levine**, **P. Abbeel**, **M. Jordan**, and **P. Moritz**, Trust region policy

optimization. *In International conference on machine learning*. PMLR, 2015.

25. **Schulman, J.**, **F. Wolski**, **P. Dhariwal**, **A. Radford**, and **O. Klimov** (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

26. **Shen, H.** and **C. Guo**, Path-following control of underactuated ships using actor-critic reinforcement learning with mlp neural networks. *In 2016 Sixth International Conference on Information Science and Technology (ICIST)*. IEEE, 2016.

27. **Shen, H.**, **H. Hashimoto**, **A. Matsuda**, **Y. Taniguchi**, **D. Terada**, and **C. Guo** (2019). Automatic collision avoidance of multiple ships based on deep q-learning. *Applied Ocean Research*, **86**, 268–288.

28. **Sivaraj, S.**, **S. Rajendran**, and **L. P. Prasad** (2022). Data driven control based on deep q-network algorithm for heading control and path following of a ship in calm water and waves. *Ocean Engineering*, **259**, 111802. ISSN 0029-8018. URL `https://www.sciencedirect.com/science/article/pii/S0029801822011489`.

29. **Takei, R.**, **R. Tsai**, **H. Shen**, and **Y. Landa**, A practical path-planning algorithm for a simple car: a hamilton-jacobi approach. *In Proceedings of the 2010 American control conference*. IEEE, 2010.

30. **TensorFlow** (). Tensorflow blog. URL `https://blog.tensorflow.org/2022/05/whats-new-in-tensorflow-29.html`.

31. **Vijay, A.** and **A. Somayajula** (2022). Identification of hydrodynamic coefficients using support vector regression. *Oceans Conference 2022, Chennai.*

32. **Wang, C.**, **X. Zhang**, **R. Li**, and **P. Dong**, Path planning of maritime autonomous surface ships in unknown environment with reinforcement learning. *In International Conference on Cognitive Systems and Signal Processing*. Springer, 2018.

33. **Woo, J.** and **N. Kim** (2020). Collision avoidance for an unmanned surface vehicle using deep reinforcement learning. *Ocean Engineering*, **199**, 107001.

34. **Woo, J.**, **C. Yu**, and **N. Kim** (2019). Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. *Ocean Engineering*, **183**, 155–166.

35. **Xu, X.**, **P. Cai**, **Z. Ahmed**, **V. S. Yellapu**, and **W. Zhang** (2022). Path planning and dynamic collision avoidance algorithm under colregs via deep reinforcement learning. *Neurocomputing*, **468**, 181–197.

36. **Yasukawa, H.** and **Y. Yoshimura** (2015). Introduction of mmg standard method for ship maneuvering predictions. *Journal of Marine Science and Technology*, **20**(1), 37–52.

37. **Yoshimura, Y.** and **Y. Masumoto** (2012). Hydrodynamic force database with medium high speed merchant ships including fishing vessels and investigation into a manoeuvring

prediction method. *Journal of the Japan Society of Naval Architects and Ocean Engineers*, **14**, 63–73.

38. **Zhao, L.** and **M.-I. Roh** (2019). Colregs-compliant multiship collision avoidance based on deep reinforcement learning. *Ocean Engineering*, **191**, 106436.

39. **Zhao, L.**, **M.-I. Roh**, and **S.-J. Lee** (2019). Control method for path following and collision avoidance of autonomous ship based on deep reinforcement learning. *Journal of Marine Science and Technology*, **27**(4), 1.

40. **Zhen, R.**, **M. Riveiro**, and **Y. Jin** (2017). A novel analytic framework of real-time multi-vessel collision risk assessment for maritime traffic surveillance. *Ocean Engineering*, **145**, 492–501.

41. **Zhou, X.**, **P. Wu**, **H. Zhang**, **W. Guo**, and **Y. Liu** (2019). Learn to navigate: cooperative path planning for unmanned surface vehicles using deep reinforcement learning. *IEEE Access*, **7**, 165262–165278.

42. **Zinage, S.** (2021). *COMPARATIVE ANALYSIS OF DIFFERENT CONTROL STRATEGIES FOR ACTIVE HEAVE COMPENSATION*. Ph.D. thesis.

43. **Zinage, S.** and **A. Somayajula** (2020). A comparative study of different active heave compensation approaches. *Ocean Systems Engineering*, **10**(4), 373.

44. **Zinage, S.** and **A. Somayajula** (2021). Deep reinforcement learning based controller for active heave compensation. *IFAC-PapersOnLine*, **54**(16), 161–167.

# CURRICULUM VITAE

| | |
|---|---|
| **NAME** | Md Shadab Alam |
| **DATE OF BIRTH** | 09 December 1997 |

**EDUCATION QUALIFICATIONS**

**2020** **The previous degree**

| | |
|---|---|
| Institution | Jamia Millia Islamia |
| Specialization | Mechanical Engineering |

**Master of Science**

| | |
|---|---|
| Institution | Masters of Science |
| Specialization | Ocean Engineering |
| Registration Date | 2nd August 2021 |

# GENERAL TEST COMMITTEE

**Chairperson**

Dr. Nilanjan Saha
Professor
Ocean Engineering
IIT Madras

**Guide(s)**

Dr. Abhilash Sharma Somayajula
Assistant Professor
Ocean Engineering
IIT Madras

**Member(s)**

Dr. Suresh Rajendran
Associate Professor
Ocean Engineering
IIT Madras

Dr. Nirav Patel
Assistant Professor
Engineering Design
IIT Madras

View publication stats