REINFORCEMENT LEARNING BASED SHIP AUTOPILOT FOR TRANSIT IN ICE

by

© Michael Dawson

A Thesis submitted to the

School of Graduate Studies

in partial fulfillment of the

requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

October 2022

St. John’s Newfoundland and Labrador

Table of Contents

Abstract

Aided navigation systems are constantly evolving and lending themselves to a growing number of applications. With a range of benefits from improved safety to reduced costs, they are an important concept within the shipping industry. This work aims to improve safety at sea in northern regions through demonstrating the potential for a ship autopilot specifically designed for transit in ice. Autopilots on ships are not a new concept – PID (proportional-integral-derivative) controller-based autopilots are already implemented onboard vessels. PID controllers, however, do not perform well when dealing with ice floes. This is because they are designed to maintain a process – such as ship speed and heading – without proper information regarding the ice around the vessel. Instead, this work aims to implement an adaptive autopilot that adjusts based on ice information ahead of the ship.

The autopilot is a reinforcement learning based model that is trained using proximal policy optimization. Within a simulated setting, the ship autopilot is shown to be able to safely navigate through ice under controlled conditions. Both the ability to travel through ice at reasonable speeds as well as avoid individual ice floes is demonstrated. Due to the limited generalizability of the current method, further development is necessary before being applied in real world scenarios.

Key terms: machine learning, reinforcement learning, ship autopilot, safety at sea, marine simulation, ice loads

Acknowledgements

My supervisors, Dennis Peters and Sarah Power, played a significant role in the focus and development of the work. I would like to take this opportunity to thank the effort and interest that they put in throughout the course of my Masters. I would also like to acknowledge the great help that I received from those in the MUN Safety at Sea project group. Special thanks goes to Mashrura Musharraf and Jennifer Smith. Mashrura was extremely helpful when it came to organizing the thoughts that I brought forward as well as providing general assistance. Jennifer provided aid in many aspects including some of the big picture questions of making sure that the Masters was heading in a direction that worked well for everyone involved.

The author acknowledges with gratitude the support of the NSERC-Husky Energy Industrial Research Chair in Safety at Sea.

List of Tables

List of Figures

List of Abbreviations

COLREG	Convention on the International Regulations for Preventing Collisions at Sea
GUI	Graphical User Interface
NFAS	Norwegian Forum for Autonomous Ships
PPO	Proximal Policy Optimization
TRPO	Trust Region Policy Optimization
PID	Proportional-Integral-Derivative
SOLAS	Safety of Life at Sea
TD	Temporal-Difference

List of Appendices

1 Introduction

1.1 Relevance of Work

Safety at sea is a constantly evolving area of study with many different factors to consider and avenues for improvement. Aided navigation is one form of potential development in this field. Aided navigation can range from providing the crew with additional information or methods that improve their ability to carry out tasks. More extreme versions of aided navigation can involve controlling aspects of the vessel's function – such as automatic adjustments of thrust and rudder angle. This can lead to crew having additional knowledge of their situation or even alleviate them of some of the tasks that need to be performed. This work looks at ship autopilot in ice as a means for accomplishing this outcome. The goal is to work towards an improved understanding of desirable behavior for a ship travelling through ice in order to provide usable information to the crew, or even provide a functioning autopilot for the most common scenarios. In addition to the improved safety of the crew, a pilot assist can improve fuel efficiency and generally reduce the cost of transit.

1.2 Forms of Ship Autopilots

PID (proportional-integral-derivative) controllers have been used for a basic form of ship autopilot for a long time. This form of autopilot allows the ship to maintain course and speed by continuously analyzing recent and current data in order to determine what adjustments need to be made to thrust and rudder angle. There are a few key limitations to this style of autopilot given the nature of their functionality. Basic forms of PID controller-based autopilots do not take into account any information regarding what is ahead of the vessel. This makes it impossible for such controllers to avoid specific obstacles or deal with rapidly changing conditions.

Some more recent forms of autopilots have been developed with more complex tasks in mind; including obstacle avoidance [1, 2]. A key difference between this style of task and maintaining course is that with obstacle avoidance, information about future events is required. That is, the autopilot must be capable of considering objects that it has not yet interacted with. One method for accomplishing this is to program a model with the desired behavior that takes into account the required information. For certain problems, this is not practical due to a non-perfect understanding of the desired behavior. The complexity of hull-ice interactions makes ship transit in ice one such scenario where it is hard to define the exact desired behavior.

Using machine learning, a model can be trained to interpret information about objects ahead of the vessel along with what to do with it. One method for doing this is to apply supervised learning with a dataset of pre-compiled cases where ships have encountered these objects. Convolutional neural networks can be trained to recognize obstacles from image – or other – data. Rules can then be applied to tell the model what to do in the different situations. For example, if an object is found to be ahead of the vessel on the port side, then the vessel should turn to starboard. A comprehensive set of rules would allow for the model to have a complex set of behavior appropriate for any situation. One downside with this form of autopilot is the challenge with developing a comprehensive set of rules for any situation. Conversely, machine learning can be applied to train a model without the need of expert knowledge. This style of learning has the downside of requiring more data but benefits from not needing to be able to succinctly define the behavioral criteria while also avoiding being biased by a defined set of rules. It is this general style of aided ship navigation that is applied in this work.

1.3 Levels of Autonomy

Levels of autonomy have been classified in many different ways depending on the application. Autonomous shipping is still in its infancy, but other industries – such as the automotive industry – have been increasingly implementing higher levels of autonomy in vehicles.

Five levels of autonomy (six counting level zero) for cars have been well laid out [3, 4]. Level zero represents no automation meaning that a human is responsible for all aspects of control. Level one is "driver assistance", where the driver is the primary controller but some assisting features – such as cruise control or lane assist – are incorporated into the system. Level 2 starts having more functions that are controlled autonomously, but the driver still plays an active role in control of the vehicle. At the third level, all aspects of control are autonomously controlled; however, the driver must be capable of taking control of the system at any time. At level 4, the vehicle can be controlled entirely without the driver under certain conditions, with the driver often provided with the option to take over control if desired or if encountering conditions that the system cannot handle. Finally, the fifth level of automation in cars is full automation where the human is no longer required, or even capable, of having any level of involvement in the control of the vehicle. While these levels of autonomy are laid out for cars, the same levels can apply to autonomous shipping. It is valuable to consider the level of autonomy that is being created in order to understand the scope of the work.

Autonomy levels for ships can be described in a similar fashion. The Norwegian Forum for Autonomous Ships (NFAS) [5] specifies four levels of operational autonomy. The lowest level of autonomy defined is "decision support" in which the crew controls the ship with the aid of advanced technology systems. The second level is "automatic" where certain operations can be completed without human interaction, but the general operation of the vessel still relies on the

actions of the crew. “Constrained autonomous”, the third level, involves the ship being fully autonomous in most situations but still requires a crew being present in order to step in as required by the autonomous system. Finally, the fourth level of autonomy discussed by NFAS is “fully autonomous”. In this level, there is no crew at all; the ship is capable of safely completing all required tasks without direct human control or observation.

There are a lot of similarities between the levels defined for cars and the levels defined by NFAS for ships; however, there are some nuances as to some of the ways they are viewed. In the automotive industry, there is a major consideration of the risk of injury to those not in the automated vehicle. This is because cars are used primarily on roads with plenty of other people. While this is true for ships in some situations – such as when docking – there are many situations at sea where no other vessels are present. For these situations, the focus is on the capabilities of the ship to completing its objective without needing to consider the risk to individuals outside of the vessel being considered. Additionally, these situations are much more controllable as it is just the environment that need be examined rather than also needing to factor in the decisions of other people or crews. This means that there are not as many factors. This work specifically deals with situations where no other vessels are present primarily for this reason, since – as the levels of autonomy suggest – it is possible to have only some tasks handled by the autonomous systems.

There are some valuable takeaways from the levels of automation in relation to this work. First of all, there are benefits from viewing it from a technical standpoint in terms of the purpose of the model. Conversely, the levels can also be considered from a more human-oriented standpoint. Here, the levels can be used both to consider what level of autonomy would be most beneficial for those involved. Both aspects are important to consider during the development of an autopilot as they govern design decisions regarding the way that the model will be developed and

applied. The capability side of things relates to what the model will be trained to do and how it will accomplish the objectives that are laid out. This can affect the information being passed to the model, the information being produced by the model, and the overall purpose of the model. The second part – the human-oriented view – can also greatly affect the development of the model. The levels are laid out in a way that deal with the roles that the crew will play on board the vessel. When considering this, the process revolves around making sure that the end product is something that is actually desired. It is all too easy to develop a new technology that has impressive capabilities but lacks practical use. This work is an early stage look into what is possible. As such, the human factors side is considered merely in terms of presenting some of the ways in which the model could be used. Determining the way that it could be implemented or applied is considered future work for expansion on this work. This is deemed acceptable since the work performed is not restrictive in how it is applied. This means that the work can be adapted in order to be applied at various levels of autonomy. Additionally, as this work is a proof of concept, the idea is to demonstrate the capabilities of the model so that real-world applications can be considered with a better understanding of what is accomplishable.

1.4 Approach

Navigation in ice has several challenges that need to be addressed. Here, mitigation of ice loads on the hull is considered. This is broken down into two main domains using two different simulated environments. The first involves navigation of a ship through constant level ice while the second deals with sparse ice floes – scattered sheets of floating ice – that are intended to be avoided completely on the way to the destination. Machine learning methods are used to create a model capable of controlling a ship's speed and heading to reach a specified destination. Criteria are pre-set to measure the successfulness of the approaches. This includes ensuring that a

reasonable route is taken as well as minimizing the number of damage instances between the ship and the ice. In the case of level ice, the vessel speeds need to be kept below a set threshold, while the second area of work requires avoidance of ice floes altogether. Certain leniencies are taken with the strictness of the criteria since this work is a proof of concept. This equates to a low number of allowable errors on the part of the model while still being considered successful. The intent is to keep the number of errors low enough to ensure that they are outliers to the overall approach. Further tuning of the methods could filter out these outliers so long as the governing characteristics of the trained models are sound.

1.5 Research Objectives

The intent of this work is to create an automated method of controlling a ship in ice. Specifically, this work involves the creation and validation of models developed using reinforcement learning – where the model is trained through repeated interaction with the environment – to navigate through ice in a reasonable fashion. This means that the ship must reach the destination in a reasonable amount of time while not incurring any excessive hull loads that would cause damage. While this work is merely a proof of concept, a fully developed model using the same methods could be used in the development of various forms of aided navigation. The information could be used to provide route planning suggestions to the crew. Alternatively, simple autopilot capabilities for typical ice conditions could function in a similar fashion to conventional aircraft autopilots; the standard cases that make up the bulk of transit scenarios are handled by the autopilot, while the crew take over for unusual cases as well as more complex maneuvers. In the long term, a full autopilot is even possible, allowing for reduced operational costs and improved safety for those operating such vessels.

2 Literature Review

There are many ways to approach the development of an aided navigation system for ships. The most appropriate form of aided navigation depends on the problem. This literature review starts by looking at some of the forms of aided navigation which have been studied. Choosing the most appropriate method requires consideration of several factors as discussed in section 2.1. In this case, a reinforcement learning based model is chosen. Section 2.2 of this review looks at why this model is appropriate as well as various ways that reinforcement learning can be applied. Finally, there are a number of restrictions that limit the effectiveness of a reinforcement learning based model to application in the real world. In section 2.3, a review of these restrictions is performed with an intent to determine whether or not such a model is transferable to applications on board actual ships.

2.1 Types of Aided Navigation in Ships

PID controllers are a conventional control method for ships that allows for automated maintenance of speed and heading. Most initial forms of ship autopilots used this type of controller, though they were not reliable in a number of scenarios such as high sea states [6]. There are a variety of methods that can boost the capabilities of PID controllers to make them more robust in ship control applications. One such method is to include a wave filter to improve control reliability in waves [7]. A sliding mode controller can be added for better roll stabilization [8]. Kalman filters improve effectiveness when dealing with high frequency noisy measurements [9, 10]. Implementing neural networks with Kalman filters can further improve the controllability and trajectory tracking capability of the autopilot [6]. A fuzzy controller can be implemented with the PID controller for better performance in sea currents [11]. These methods each aim to improve upon a specific aspect of ship autopilot where standard PID controllers are inadequate.

Most of these styles of controllers are designed to improve ship autopilot for travelling in open water. PID controllers and similar forms of autopilot are not capable of any kind of obstacle avoidance due to using purely historic data with no knowledge of obstacles ahead of the ship. This requires a different style of controller that is aimed at preventing or mitigating collisions. Making adjustments based on the frequency of recent interactions is not enough for transit in ice; while ice can remain fairly constant at times, the size, thickness, and frequency of ice interaction is non-constant and generally less predictable than even that of wind and waves. Additionally, the penalty for improper control in these situations – hitting ice too quickly – can be much more severe. At the very least, the above controllers would need to be modified to incorporate additional information for the objects that need to be avoided.

There are a number of aided navigation methods that can take provided ice information into account. This can involve modification of previously mentioned methods or a completely new style of controller. For instance, if an ice load prediction model can be created, an algorithm for adjusting speed and heading based on the predicted loads can be developed. Part of the issue with this is that accurately predicting ice loads is complicated due to the inconsistencies in the ice [12] [13]; however, many studies have worked towards this goal [13, 14, 15]. Alternatively, machine learning could be applied to develop an autopilot using pre-existing data of ship-ice interactions. The primary issue with this is that the amount of data needed for such training does not currently exist. This is largely true because of the need for both safe and unsafe ice interaction data. It is the lack of sufficient amounts of hull-damaging data that makes machine learning challenging to apply properly. Without having data for the unsafe interactions, machine learning models would be incapable of learning what variables lead to unsafe conditions. However, given a simulation environment, reinforcement learning can be used to develop a controller capable of navigating

through ice. This method has the benefit of being able to create its own training data, thereby removing the limitation of needing previously collected data. Additionally, simulation-based learning methods can learn from catastrophic behavior that are impractical to examined in real-life settings – such as hitting very thick ice at high speeds. This makes reinforcement learning a good choice for learning about complex problems for which simulators can be developed.

2.2 Reinforcement Learning

Reinforcement learning is a form of machine learning in which a model is trained to map states to actions through repeated interaction with the environment [16]. The interactions are displayed in Figure 1 with *S* representing the state space, *R* representing the reward, *A* representing the chosen action, and *t* being the timestep.

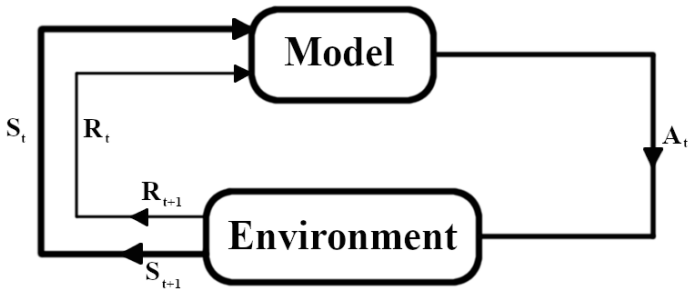


Figure 1: Reinforcement Learning Environment Interaction [17]

The learner is provided with a state space representation and selects an action to take, from an available action space, based on the current state. This action is passed as a command to the environment which carries out the appropriate task and returns a value – referred to as a reward – that denotes how good the result of the chosen action was. The function which determines the value to be passed back to the model is called the reward function. It is a critical aspect of reinforcement learning that dictates good and bad behavior and, therefore, determines the trends learned by the model. This trial-and-error approach to learning is similar to how any naive

individual would go about learning a task from scratch. The process of interacting with the environment through taking actions over and over again is the basis of reinforcement learning. The rewards can be presented to the learner in various ways depending on the environment and the type of reinforcement learning algorithm being applied. For simple problems, a basic reward function can be applied effectively. The reward function gets increasingly complex with more complex environments and objectives. The selected algorithm also affects the method for updating the decision-making model used for selecting actions.

Large or continuous state or action spaces make many learning algorithms impractical, or even impossible, to apply [18]. Algorithms such as Q-Learning, in their base form, only function in discrete state and action spaces [19]. These methods can often still be applied by modifying them to use function approximation [20]. Deep Q-learning (DQN), for example, utilizes neural networks as function approximators in order to apply Q-learning to large or non-discrete state spaces [21] [22]. Alternatively, the state space can be reduced from a continuous state space to something much more manageable using discretization methods such as Tile Coding [23]. Finally, there are a variety of algorithms designed to function for large or continuous state and action spaces. Among others, this includes gradient-based temporal-difference learning, policy-gradient algorithms, and continuous forms of actor-critic methods [24, 25]. Proximal Policy Optimization (PPO) – one of the primary algorithms applied in this work – is a policy gradient reinforcement learning method aimed at being scalable, data efficient, and robust when dealing with continuous state spaces using neural networks as function approximators [26].

2.2.1 State Space

The state space is the representation of the environment that is passed to the model. It is not necessarily a comprehensive description of the environment. Rather, the state space is merely

a representation of the current situation that allows the model to make a choice as to the action that should be taken. Creating a practical state space representation can be challenging. It is important to provide enough information that the model will be able to learn the required trends, but this should be done as simply as possible to facilitate learning.

2.2.2 Action Space

The action space defines the way in which the model can affect the environment. For discrete action spaces, there is a list of actions that can be taken at any given timestep. Similarly, for continuous action paces, there is a range of acceptable values for one or more parameters that, as with discrete environments, can be selected for a given timestep.

An action is selected at the start of each timestep by the model. The selected action determines the interaction with the environment. For example, in a grid-world environment – that is, a checkerboard style environment where each square is a separate state – the action could determine an orthogonal direction of travel for that timestep. Comparatively, for a ship navigation problem, the action space can be set up to specify the adjustments made to both the ship thrust and the rudder angle. In general, the action space is defined based on the control options that are desired for the environment.

2.2.3 Reward Function

In reinforcement learning, a model is trained to perform actions that result in the maximum overall reward over the course of a trial. The reward is a numeric value that denotes the desirability of an outcome. A reward function is the equation that is used to determine the reward for any actions taken. The model's behavior is adjusted directly based on the reward received.

Reward functions can be extremely simple or very complicated depending on the complexity of the goal that is desired. For example, the reward function can be as simple as $R = -1$. This singular value reward function states that any action taken is equally bad regardless of the state or any other factors. As a result, the only influence on the overall reward for the trial is the duration of the run; the longer the trial lasts, the more negative the reward. Due to this, a correctly trained model would learn to ensure the trial lasts for as few timesteps as possible. This would be a functional reward function for a simple grid-world problem where the objective is to reach the destination in as few steps as possible. As the problems get more complex, or the objectives themselves are less one-dimensional, the reward function tend to get more complex as well. In the case of a ship travelling through ice, there are multiple factors at play that each need to be appropriately factored into the reward function.

2.2.4 Exploration vs Exploitation

A key concept in reinforcement learning is the idea of exploration versus exploitation. Exploration involves taking actions that are not well understood in order to gain better knowledge of the possibilities. Alternatively, exploitation uses the knowledge that has previously been acquired to determine the next action. An algorithm that focuses too much on exploration will fail to fully analyze the more optimal choices to develop more complex ideal behavior whereas algorithms that too heavily favor exploitation will fail to discover certain good behaviors that may be less likely to occur randomly. A common method is to focus more on exploration during the early phases of training the model followed by focusing on exploitation during the later phases. This allows the model to find the trends that tend to give better results and then refine them to reach the local or global optima.

An example of an exploitation-based algorithm would be a simple greedy algorithm. Greedy algorithms always go with the action that the model predicts to provide the largest overall reward. One method for improving the exploration of the state space for greedy methods is to add a randomness variable, epsilon [27]. This leads to a form of algorithm referred to as an epsilon-greedy algorithm. With epsilon-greedy, a random action is taken at a specified percentage of frequency (based on the epsilon variable), while the rest of the time the option that the model predicts will provide the greatest reward is taken. This encourages exploration by preventing the model from settling on a local optimum too quickly. Over the course of training, the frequency of random actions being taken can be decreased by reducing the value of epsilon. In doing so, the model starts to have an easier time converging once as the training gets closer to completion. The net outcome is that the model learns to explore early on – so that the state space is better explored – and then learns to improve upon what was deemed to be the best general course of action to optimize the solution. In other words, the training process starts focused on exploration and slowly becomes increasingly exploitation driven over the course of the training.

It is also possible to learn the optimal value of one policy while using a separate policy for selecting actions. This allows for the policy used for selecting actions to be more focused on exploration than the policy being updated for optimal behavior. For example, action selection can be performed using an algorithm such as random walk (completely random actions) or epsilon-greedy for exploration purposes, while a pure greedy algorithm is applied for the policy being learned. This style of learning is referred to as off-policy learning.

2.2.5 Tiered Learning

For many complex problems, multiple phases of learning have been observed [28, 29]. During the early stages of training, the model learns to accomplish the simple behaviors that lead

to decent results. As training continues, the model will learn to optimize these simple rules. Due to exploration, the model will occasionally happen upon a new idea or concept that it had not yet attempted or deemed acceptable. This can lead to jumps in the overall effectiveness of the model. In general, this corelates to local and global optima. Models can get stuck in local optima and never figure out alternate strategies that would lead to even better results, ultimately the global optima. Proper exploration can help to discover these alternative strategies. New strategies frequently create a jump in model capability. The development of the model with successive strategies being found creates a sort of tiered learning. A great demonstration of this is with a multi-agent hide-and-seek reinforcement learning problem developed by OpenAI [28]. Here, one group of agents is tasked with catching a second group of agents that themselves are attempting to avoid being caught. Over the course of the training, distinct strategies get developed that result in the other side needing to look for new methods of accomplishing the goal.

2.2.6 Types of Reinforcement Learning

Learning algorithms can be policy-based, value-based, or Actor-Critic which is a combination of the two. Algorithms can also be either model-based or model-free. Policy-based algorithms map states to actions based on probabilities of different actions being taken from a given state. These can be either deterministic or stochastic based on the type of problem being solved. By comparison, value-based algorithms map states to values for each of the potential actions that can be taken from a given state. Policy-based algorithms tend to be more sample efficient than value-based algorithms but typically provide more stable learning [30].

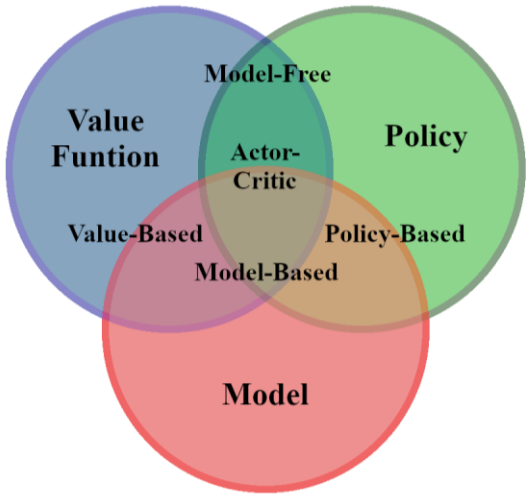


Figure 2: Types of Reinforcement Learning [31]

A policy is a function that maps states to actions. Policy-based algorithms update the model's policy directly using methods such as gradient ascent. With gradient ascent, the policy is updated using the gradient of the difference between the old policy and the new policy based on the reward. [32] With value-based learning, state-action pairs are assigned values that are updated over the course of the training process. A policy uses these values to select actions, but it is the values, not the policy, that gets updated.

2.2.7 Reinforcement Learning Algorithms

There are a large variety of learning algorithms that have been developed for solving myriad problems. Some examples of algorithms include temporal-difference (TD) learning, TD(λ), Proximal Policy Optimization (PPO), and TD(λ) with Tile Coding [20, 17].

2.2.7.1 Temporal Difference Learning

Temporal-difference (TD) learning is a key concept in reinforcement learning upon which many other algorithms are based. The premise of the algorithm is to update the model based on the difference between the outcome – in terms of reward – that is achieved compared to that which

was expected by the model. At each timestep, the model selects the action to be taken based on the expected reward for each possible action given the current state. After taking the action, the model is provided with the actual reward received. The difference between the expected and actual rewards creates an error, referred to as the TD error. TD(λ) is an extension of TD learning that introduces a lambda decay parameter. This parameter is used to create a weighted average of the rewards that makes short term rewards more relevant than long term rewards. The addition of the lambda parameter also allows the learning to be performed on-line rather than off-line, meaning that the model can be updated continuously rather than needing to wait for the end of a given trial.

2.2.7.2 Proximal Policy Optimization

Proximal policy optimization (PPO) is a policy-based algorithm developed by OpenAI Five – an AI research company. The method is a form of model-free deep reinforcement learning and is explained in depth in section 3.3.5.2.

2.3 Simulated Environments for Reinforcement Learning

Simulators can play an important role in the safe development and validation of ship navigation models [33]. For reinforcement learning, a simulated setting is necessary in order to train a model for most applications. This is due to the massive amount of data needed that makes it impractical to train a model in real time. Additionally, for ship navigation in ice, part of the negative behavior that the model needs to learn comes from simulated hull damage from unacceptable ice loads – further making it impractical to learn through real world training.

In some applications, the simulation environment can be the intended environment of the final product. In other instances, such as this work, the environment is merely a medium for creating a model that is intended to solve a problem. The transition between simulation and real-

world environments creates a source of error referred to as the simulation-reality gap [34]. Any discrepancies between the physics or control parameters creates a source of error that can lead to issues with the reliability of the trained model. Because of this, it is critical that the environment representation be as realistic as possible. For ship transit in ice, this largely comes down to development of an accurate physics engine for simulating vessel control, ice-floe movement, and ship-ice interactions. From the reinforcement learning side, it is important to use a state space representation that is comparable to that which could be achieved in the associated real-world scenario. It is also important that the control be representative of proper vessel control.

Obtaining ice data ahead of a ship in real time is a non-trivial task, however, there are a number of methods that can be applied to obtain basic ice data. Multiple studies have used sonar to obtain ice thickness data using statically located sonars above or below the ice being measured [35, 36, 37]. These conditions are not transferable to application on board a ship. LiDAR is another type of sensor that can be applied to obtain ice data. One study used a downward facing LiDAR sensor in order to get ice thickness [38], while another study looked into the potential use of LiDAR for measuring sea ice surface roughness [39]. Finally, pure image data could be analyzed to determine details about the ice floes near the vessel. Yan, Tan, and Su [40] used a convolutional neural network to classify aerial images of ice scenes into one of six different categories that defined the type of scene in the image. Hall, Hughes, and Wadhams [41] also analyzed ice images, though they used images taken from onboard a ship to determine the ice concentration ahead of the vessel based on the locations and sizes of ice floes in the scenes.

Due to the nature of this work, the exact method for collecting ice data is not discussed in depth. The state space, however, is applied in a manner that takes into account the type and accuracy of data that could practically be collected if desired.

2.4 Conclusions

A wide range of styles of aided navigation exist with different goals and drawbacks. Many of these are of little use when it comes to navigation through ice where hulls loads are a critical factor. Reinforcement learning can be applied to develop a model capable of solving this problem. In order to accomplish this, a simulator is needed for training; however, applying the learned model to the real world is limited by the simulation-reality gap. This issue is mitigated by ensuring the physics and state representation are as realistic as possible. This allows for a powerful style of machine learning that can result in a much stronger model than would otherwise be achievable.

3 Methodology

3.1 Introduction

Two separate simulation environments are used for this work. Each of the environmental setups are intended to address different challenges of ship navigation in ice. The two environments are treated independently with unique models being for each of the environments. The first environment, referred to as the "simple environment", is a simplistic level-ice environment. This environment is designed to test the ability of a model to adjust course to reach a specified destination while maintaining speeds that will avoid harmful hull loads. The second environment, referred to as the "GEM environment", uses a simulation environment called "GEM" to test the ability of a model to avoid sparse ice floes while travelling to the destination.

3.2 Experiment #1: Simple Environment

3.2.1 Introduction

A simplified test environment is used prior to implementing in GEM in order to work out some of the hurdles in an environment with fewer variables. In this version, only level ice is considered, with a requirement that the ship stay below a specific speed dictated by the ice thickness. The ice thickness is chosen at the start of each simulation and kept constant throughout the simulation. Although this does not account for an ice field with varying ice thickness, it suffices for this simulation and is representative of the best information that is typically available on-ship.

3.2.2 Objective

This work aims to demonstrate the capability of solving the same problems involved with a ship navigating through level ice. Specifically, this includes the ability of adjusting thrust and rudder angle over a significant number of time-steps to control the vessel and maneuver towards

the destination. Additionally, it is important to maintain a speed below that which would cause damage. Another goal of this section of the work is to ensure that the selected algorithms can solve a simplified version of the problem before implementing in GEM. Furthermore, many aspects of the state and action spaces applied in this environment can be carried forward into the more complex one. Determining the aspects of the state and action spaces that work best is easier in the simpler environment where there are fewer variables involved. Given the inherent complexities of determining the optimal state and action space representations, the ability to develop the initial machine learning framework up front is particularly beneficial.

3.2.3 Physical Representation

In order to properly represent the problem being considered, the method for controlling the vessel – adjusting thrust and rudder angle – needs to be maintained, along with the general way that adjustments will affect the vessel's location. The exact representation, however, is not critical since relevant conclusions can be drawn without perfect physics. As such, this simplified environment uses an abstraction of the real-world physics to capture the problem.

Specifically, all values are unitless and the ice representation is set as discrete intervals of 0.5 that define the minimum speed that results in damage as per

$$v_{damage} = 7 - (Ice\ Thickness)^2. \quad (3.1)$$

This equation is set up to require the ship to move substantially slower as the ice thickness is increased. The intent here is to ensure that the model functions for a range of restrictions on the vessel's speed. Furthermore,

$$v_1 = [v_0 + c_t\,(A_t - 1)] - (c_r\, v_0^2) \quad (3.2)$$

is used to define the vessel speed where v_0 and v_1 are the old and new speeds respectively, $c_t = 0.6$ is the thrust coefficient, cr = 0.01 resistance coefficient, and A_t is the acceleration that is directly controlled by the selected thrust action at each timestep.

3.2.4 Action Space

The action space is designed to provide the model rudder and thrust control for the vessel. In order for the control to be more realistic, the actions only make minor shifts to the thrust and rudder angle such that it takes several actions in a row to make large adjustments to thrust or rudder. This makes the problem more realistic by preventing sudden jolts in control parameters that would not be achievable in a real-world setting.

Action selection determines the adjustment made to both the thrust and rudder at each timestep. A total of five rudder values (zero to four) and four thrust values (zero to three) make up the action space. At any given timestep, both a rudder value and a thrust value are selected making for a total of twenty different actions that can be selected. At the extremes, the rudder action can cause a shift in rudder angle of up to 18-degrees in either direction. A value of zero equates to an 18-degree adjustment to starboard for the rudder while a value of four equates to an adjustment of 18-degrees to port. A value of two indicates that the rudder angle does not change for that timestep. The thrust actions are applied in a similar fashion. A thrust action of zero indicates a negative thrust while a thrust action of three is the largest thrust option resulting in the greatest acceleration for that timestep.

3.2.5 State Space

The state space representation for this environment is made up of three parameters: relative bearing, vessel speed, and ice thickness. With these three values, the model can make a sufficiently

informed decision as to the best set of actions to take. Relative bearing refers to the angle between the vessel heading and the direction of the destination. This is a useful parameter as it indicates whether adjustments to the heading – through modifications to the rudder angle – are required. Vessel speed is necessary for informed action selection for the thrust action; high thrust is desired at low speeds in order to bring the vessel up to speed, while low, or negative, thrust is desired at higher speeds as to ensure that the vessel does not reach speeds great enough as to cause damage to the hull. The ice thickness state parameter is a discrete value ranging from zero to two. The possible options are 0.0, 0.5, 1.0, 1.5, and 2.0 – equating to maximum allowable speeds of 9.75 (max achievable speed), 6.75, 6.00, 4.75, and 3.00, respectively. This value is selected randomly at the beginning of a scenario and then kept constant throughout the run.

3.2.6 Environment Representation

3.2.6.1 Gym

Gym is a reinforcement learning toolkit that provides a template for development of reinforcement learning environments. One advantage with formatting the environment using gym is that it allows for easy application of existing algorithms. Additionally, gym provides support for defining the state and action spaces to fit the problem. The primary methods for the environments are the step and reset methods. The step method, for the ship navigation model, takes the action value as a parameter and is used to inform the environment how to proceed. The reset method does not require any parameters and is called prior to taking the first action of any trial. Upon reaching a termination state, the reset method is called again to reinitialize the environment before starting a new trial.

3.2.6.2 Environment Initialization

For the ship navigation environment, the aim is to have the model be able to reach the destination regardless of the initial start or end point. Therefore, the reset method randomizes the location of the destination in terms of distance from the vessel as well as the relative bearing between the ship and the destination at the start of the run, with the vessel always being placed at the origin, (0, 0). Since the state space representation is relative to the ship, randomizing either of the ship or destination suffices for a fully random scenario. Both ship speed and rudder angle are initialized to zero. This is partially to ensure that the model learns how to bring the ship up to an appropriate speed from rest, as well as to make sure that the model does not have to deal with initial circumstances that make it impossible to avoid unacceptable ice collisions (e.g., a vessel speed above the allowable speed threshold). Finally, as discussed in Section 3.2.5, the ice thickness is set as a random value ranging between 0.0 and 2.0. The state parameters are then calculated and passed back to the model as the starting state for the run.

3.2.7 Model Training and Validation

The method for training and validation of the reinforcement learning model is outlined below. Temporal-difference learning with tile coding is applied because the method functions well when applied in a relatively simple continuous state space. An off-policy method is applied with epsilon-greedy being used for training the model and a pure greedy model being used during testing. Epsilon-greedy is applied during training for better exploration of the state space.

3.2.7.1 Temporal-Difference Learning with Tile Coding

As described in section 2.2.7.1, TD(λ) is a fundamental concept in reinforcement learning. Basic TD learning involves updating the model as per

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\,[R_{t+1} + max_a\, Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (3.3)$$

Here, $Q(s_t, a_t)$ represents the value for a state-action pair at a given timestep prior to an action being performed, α is the learning rate, R_{t+1} is the reward received based on the selected action, and $max_a\, Q(s_{t+1}, a)$ is the current greedy prediction for the new state. The greedy prediction is used as a metric of how good the new state is since it corresponds to the best possible outcome that the model currently believes is possible from that state. This method can be modified to implement alternative methods of gauging the value of the state space such as stochastic representations. In this case, a pure greedy evaluation is used since the model has full control over action selection and there is no randomization affecting the optimal actions. TD(λ) modifies the update by adding an eligibility trace parameter, e, based on the trace decay parameter, λ. The eligibility trace causes shorter term rewards to have a greater impact on a value's update than longer term rewards by decaying the importance of each consecutive reward after the action associated with the value being updated.

3.2.7.2 Hyperparameter Tuning

The hyperparameters for this environment are the learning rate, α, decay parameter, λ, and probability of random exploration, ϵ. The learning rate, α, is initially set to 1 and then decreased during training based on the number of times the respective state action pair has occurred. This creates a jump to the first value that is achieved for a given state-action pair but allows for increasingly minor shifts as the model becomes better informed. Decreasing the learning rate over time helps to properly converge on a solution. The lambda decay parameter is set to 0.8 and remains constant throughout training. Finally, the exploration parameter, ϵ, is set to 0.3 for the entirety of the training process, indicating a static 30 percent chance of a random action being selected.

3.2.7.3 Tile coding

Tile coding is a form of state aggregation for dealing with large or continuous state spaces. It transforms the state space into a discrete learning environment, thereby allowing for TD-learning to be applied. It is used here to allow a standard TD(λ) algorithm to be applied since both relative bearing and vessel speed are continuous state space parameters.

Provided below in Figure 3 is an example of 2D tile coding with four tilings. In this example, each tiling consists of a four-by-four grid of tiles. Each of these tiles represent a single aggregation of the continuous state space. Here, each of the tilings are offset at equal distances from each other in both dimensions; in practice, this is not ideal. Instead, a displacement vector is used to determine the amount of offset that should be used. Sutton and Barto [23] suggest that a good method for determining this vector is to use 1 for the first dimension, and then increase by 2 for every consecutive dimension. While tile coding can be applied without varying the offsets, applying this method leads to a more consistent state representation for more efficient learning.

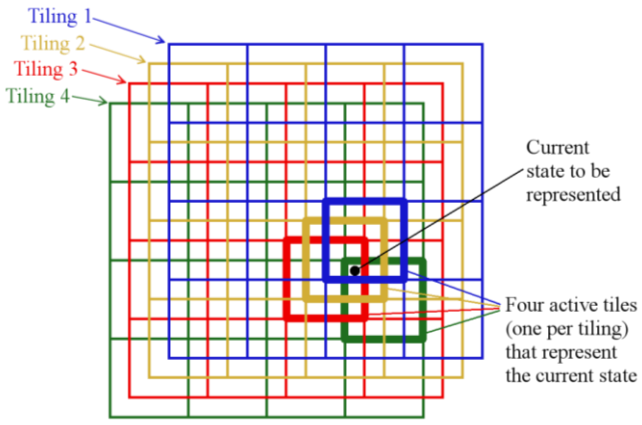


Figure 3: Visualization of 2-Dimensional Tile Coding [23, 42]

The number of tilings is determine based on equations:

$$n = 2^p \qquad (3.4)$$

$$n \geq 4k \quad (3.5)$$

where n is the number of tilings, k is the number of dimensions in the state space, and p is any positive integer. Both of these equations must be satisfied for tile coding to be applied as per the prescribed method.

3.2.8 Reward Function

In order to capture the desired multi-objective behavior, the reward function is made up of several parts. First of all, a reward (R) is achieved at every timestep based on the progress that the vessel makes towards the destination,

$$R = (-1 + d_0 - d_1)\,(10 - v_{damage}), \quad (3.6)$$

where d_0 and d_1 are the distances between the ship and the destination at the start and end of the timestep, respectively, and v_{damage} is the minimum speed that will cause hull damage for the current ice condition. A scale factor of $10 - v_{damage}$ is applied purely as a means for the reward to scale appropriately over all ice conditions. Without the scale factor, a very much smaller reward is achieved for good behavior in thicker ice than it is in thinner ice conditions. The most important aspect of the reward is $d_0 - d_1$. This portion of the reward makes it so that the more progress is made towards the destination in a given timestep, the greater the reward. By itself, this does not ensure that the greatest reward is achieved for reaching the destination as soon as possible – which is the desired outcome (setting aside ice collisions for the time being). In order to provide an incentive for the destination to be reached in as few timesteps as possible, a value of -1 is added to this reward. As a result, for this part of the reward, the net reward for any trial where the destination is reached is

$$R_{total} = (-n_{timesteps} + d_{total})(10 - v_{damage}) \quad (3.7)$$

where $n_{timesteps}$ is the duration of the trial in timesteps and d_{total} is the total distance between the vessel and the destination at the beginning of the trial.

In addition to reaching the destination, it is also critical that this is achieved without causing damage to the hull of the vessel. This is accomplished by providing a penalty (negative reward) for any timestep that corresponds with hull damage. This negative reward is set to be very large as to ensure that there are no situations where incurring hull damage to reach the destination faster receives a greater overall reward. Large penalties can lead to overly conservative models in some problems, however, since hull damage is strictly unacceptable, it is better to be conservative here than it is to incur occasional instances of damage. The modification to the previous reward, received for any timestep in which the ship is travelling too quickly for the ice condition, is defined by

$$R \leftarrow R - 1000\,(|v| - v_{damage}). \tag{3.8}$$

3.2.9 Validation Criteria

Validation criteria – determined prior to creating the model – are used for the purpose of determining the efficacy of the model. These criteria aim to encapsulate the acceptable level of the desired behavior within a reasonable margin. It is worth noting that this is not designed to reach the optimal behavior. Instead, the validation criteria consider reasonable levels of risk and define suitable requirements accordingly.

Four validation criteria are tested as per Table 1. It is required that the destination is reached a minimum of 95% of the time. A small allowance is provided here to account for outliers and instances where generally good behavior is displayed, but the vessel does not quite reach the destination. The maximum number of allowable damage instances is set to 10 over the course of

the entire validation phase for each of the five scenario types. As this is the net sum rather than an average, this is fairly low with only allowing an average of one collision every 10 runs. This is set low intentionally as it is an important metric for the proof of concept. Finally, while harmful collisions are intended to be avoided entirely, reaching the destination in a timely manner is also important. The final two validation criteria relate to the minimum speed of the model. This is set to ensure that the vessel proceeds at reasonable speeds rather than simply travelling at extremely slow speeds towards the destination in order to avoid harmful ice collisions. A minimum overall average speed is set as 50% of maximum allowable speed for each scenario, with the average of the maximum speeds of each trial being set as a minimum of 70% of the maximum allowable speed.

Table 1: Validation Criteria

Validation Criteria	Ice Condition				
	0.0	0.5	1.0	1.5	2.0
Percent Successful Completions	>95%	>95%	>95%	>95%	>95%
Total Damage Instances	≤10	≤10	≤10	≤10	≤10
Overall Average Speed	>4.88	>3.38	>3.00	>2.38	>1.50
Average of Maximum Speeds	>6.83	>4.73	>4.20	>3.33	>2.10

Once the model is trained, it is tested by running through each of the five separate ice thicknesses 100 times each with random start and end points. The data collected during these trials is compared against these criteria.

3.3 Experiment #2: GEM Environment

3.3.1 GEM

GEM is a program designed for simulated ship transit in ice. The program is capable of calculating the hull load on a vessel from interactions with ice. It also allows for easy set up of test environments with the ability to randomly generate scenarios. Also relevant to this work is the

control parameters for navigation of the vessel. In GEM, at any given time-step, the rudder angle and propeller thrust can be set. The simulation can be moved forward one step at a time with measurable values being studied for each step individually. This is used to provide state information to the learning algorithm [43]. Current and wind can also be implemented using GEM, however, these are considered out of scope for this project for simplicity purposes..

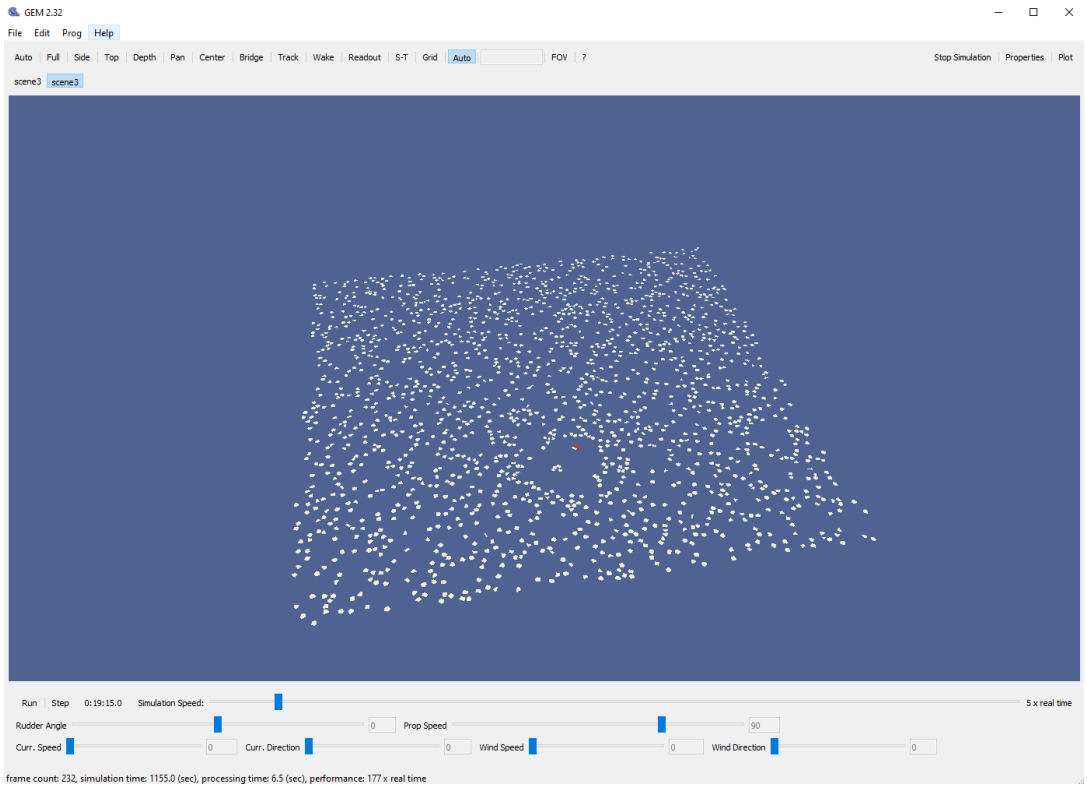


Figure 4: GEM GUI

3.3.1.1 Gym Wrapper

Within Python, the reinforcement learning model is trained through interaction with the gym environment. The gym environment is essentially a wrapper on the simulator so that the algorithm can easily interact with the simulator as expected. The actual simulator – GEM – is implemented with Qt (using C++). Python passes information to GEM as JSON messages using a

socket. Qt contains the server socket that receives the incoming connections and specifies the appropriate actions to be performed in the simulator.

3.3.2 Objective

This section of work aims to demonstrate the ability of a model trained using reinforcement learning to navigate around ice in sparse ice floe conditions in order to reach a destination in a reasonable manner. Here, any collision with an ice floe is considered unacceptable.

3.3.2.1 Scenarios

Multiple types of scenarios are used for training the model to ensure that final model is able to generalize to different situations. The scenarios are intended to reflect situations that could be encountered by a ship travelling through ice. This is not meant to be a comprehensive set of training scenarios but, rather, a set of scenarios designed to represent a narrow range of potential situations.

The overall area containing ice is a 4000m by 4000m square. This is considered to be the bounds of the environment and the model is penalized if the ship travels outside of this region. Ice thickness remains constant at 1m across all scenarios. Ice breaking is not considered within this work as it is not fully implemented in GEM. Similarly, sheet ice is not included at this point.

The first of four scenario types is a sparse ice condition with only 2 percent ice coverage using large floes. The size of the floes range from 140m to 170m in diameter and are irregularly shaped with anywhere from four to seven sides. Here it is reasonably easy for the model to maneuver around the large ice floes and it provides an example of scenarios where all ice can be reasonably avoided. The second type of scenario is the same as the first except with 5 percent ice coverage instead. For the third scenario type, the ice coverage is increased 10 percent while the

floe size is reduced to a range of 100m to 120m in diameter. This creates scenes with much more densely packed ice making it much harder to avoid collisions. Finally, the fourth scenario keeps the ice coverage at 10 percent, but reduces the size of the floes to a diameter range of only 40m to 60m. For the more densely packed scenarios, the intent is to check to see if the model is able to navigate through the field with minimal collisions, if any. The full scenario details are displayed below in

Table 2: GEM Environment Scenario Details

Scene Type	Ice Thickness	Ice Coverage	Floe Size (min)	Floe Size (max)	Shape (min sides)	Shape (max sides)	Floe Gap (min)
	[m]	[%]	[-]	[-]	[-]	[-]	[m]
2% Coverage	1	2	140	170	4	7	0.1
5% Coverage	1	5	100	120	4	7	0.1
10% Coverage (large floes)	1	10	100	120	4	7	0.1
10% Coverage (small floes)	1	10	40	60	4	7	0.1

Once the model is trained, it is tested by running through 100 scenarios for each of the conditions being considered. The full details for the scenes used during testing are shown in Appendix A. These scenarios are set up such that there are no issues involving initial collisions between the vessel and the ice. For this, each of the starting locations for the vessels are determined at random and then slightly adjusted as necessary. The same process is followed for the destination location to ensure that there is no overlap between the destination and an existing ice flow.

3.3.3 Action Space

The action space for this environment is similar to that of the simple environment (detailed in section 3.2.4). Here, the action space is continuous with the rudder and propulsion action values being selected independently as float values between negative one and positive one. A continuous

action space is applied due to the additional control that it provides to the model. Furthermore, there are no problems with using a continuous action space here given that the PPO algorithm is the only algorithm being applied at this stage and it is capable of selecting actions from among those presented by continuous action spaces.

3.3.4 State Space

The state space is made up of nine parameters; four of which are for providing ice thickness information for the ice within the designated field of view – discussed in depth in section 3.3.4.1. The non-ice related parameters are relative bearing to the destination, distance to the destination, ship speed, propeller speed, and rudder angle.

All of the simulations use Cartesian coordinates for locations of ice floes and the ship. For the state space, polar coordinates are used for the distance and direction to the destination instead. This is due to it being a better representation of the state space from a simplicity of learning perspective. By using a combination of distance and relative bearing to destination, the model gets a good representation of what needs to be done in order to reach the destination with only two values. Relative bearing is enough for the model to know if the ship is heading in the correct direction or, alternatively, the degree to which the ship is off course. Combining this information with the range to the destination allows the model to make decisions pertaining to how fast adjustments need to be made. While this does not fully inform the model of the exact state space, the unknown information is not relevant to decision making. For example, the model would consider the two different state spaces in Figure 5 to be the same.

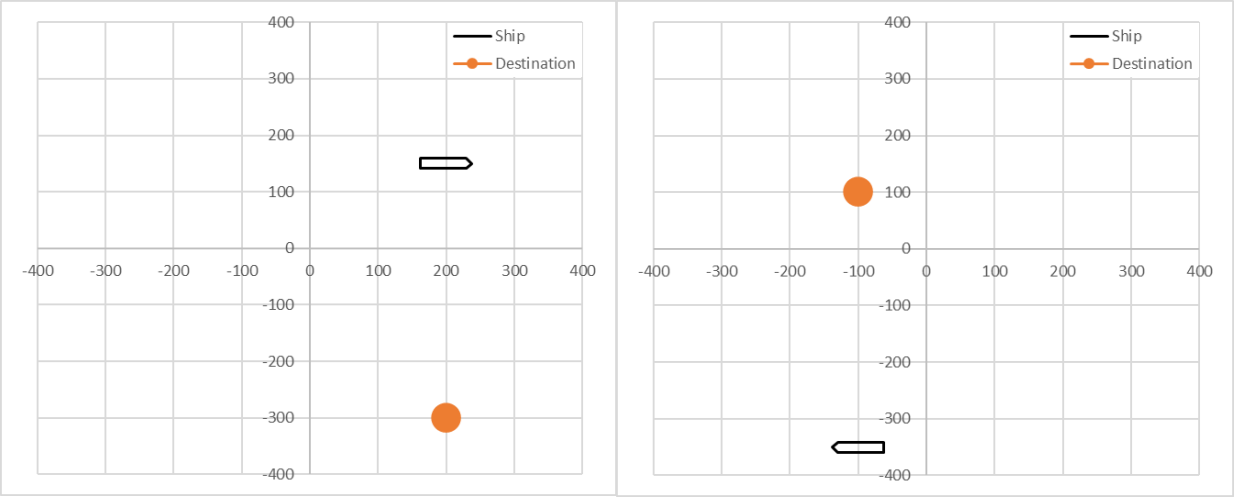


Figure 5: Similar State Space Representation Comparison

In the left instance the ship is at (200, 150) with a heading of 90 degrees and the destination is at (200, -300). In the right image, the ship is at (-100, -350) with a heading of 270 degrees and the destination is at (-100, 100). Even though these two states are different, the state space representation for both will be the same with a relative bearing of 90 degrees to the destination at a distance of 450. While instinctually this might seem problematic, it is actually desirable. This is because the actions that provide the optimal result are identical in these two cases. In both situations the ship needs to turn the exact same amount to starboard and travel an identical distance to reach the destination.

3.3.4.1 Ice Information Representation

Ice information could be represented in the state space in countless different ways. For example, the state representation could have the distance, size, and thickness of every single ice floe in the simulation. There are a few reasons why this would be a poor choice of representation. First of all, this would result in an extremely large state space making proper learning much more complicated. Secondly, this would have a non-constant size state space since the number of ice floes is not always the same. This too would complicate the learning process. Finally, as previously

discussed in section 2.3, the state information should be representative of information that could be obtained for its intended use. While it is possible to get limited ice information surrounding a vessel, detailed information of ice far away from the vessel is unrealistic. Due to this, even if the model successfully learns using the complicated state space, the resulting model would have limited practical applications.

A simpler and more realistic representation of the ice is needed. This is achieved through a form of discretization of the observation space. One method for accomplishing this is to split the 4000m square of relevant space into a 10-by-10 grid and pass ice values for each grid section – such as maximum thickness for each section. Doing this would provide a constant size state space with a more reasonable number of parameters. This can be further simplified by only looking at the ice nearby the ship, and defining the relevant sections, or zones, relative to the ship's location.

Below are three example zone layouts. At every time-step, ice information is collected for each zone and included in the state space. Specifically, the model is informed as to whether or not an ice flow is present in each zone. This is intentionally very simple information to make it both easier to learn and more real-world applicable at the same time. The total number of state parameters needed to represent the ice is equal to the number of zones, so only having a single value per zone greatly reduces the number of state space parameters. In the future, this could be expanded to include information such as maximum ice thickness, average ice thickness, ice concentration, and so on.

Figure 6 shows one potential zone layout that uses overlapping squares of increasing size. All six squares start near the bow of the ship on the centerline and provide information on either the port or starboard side of the vessel. This is beneficial as the ship can use this information to

decide whether or not it can maneuver to one side or the other in order to avoid larger ice floes. The smaller squares provide information for the very near future while the larger squares provide a broader idea of whether or not the ice flows are generally more hazardous to port or starboard ahead of the vessel. This provides an acceptable level of information, but there are more appropriate ways to relay such information.

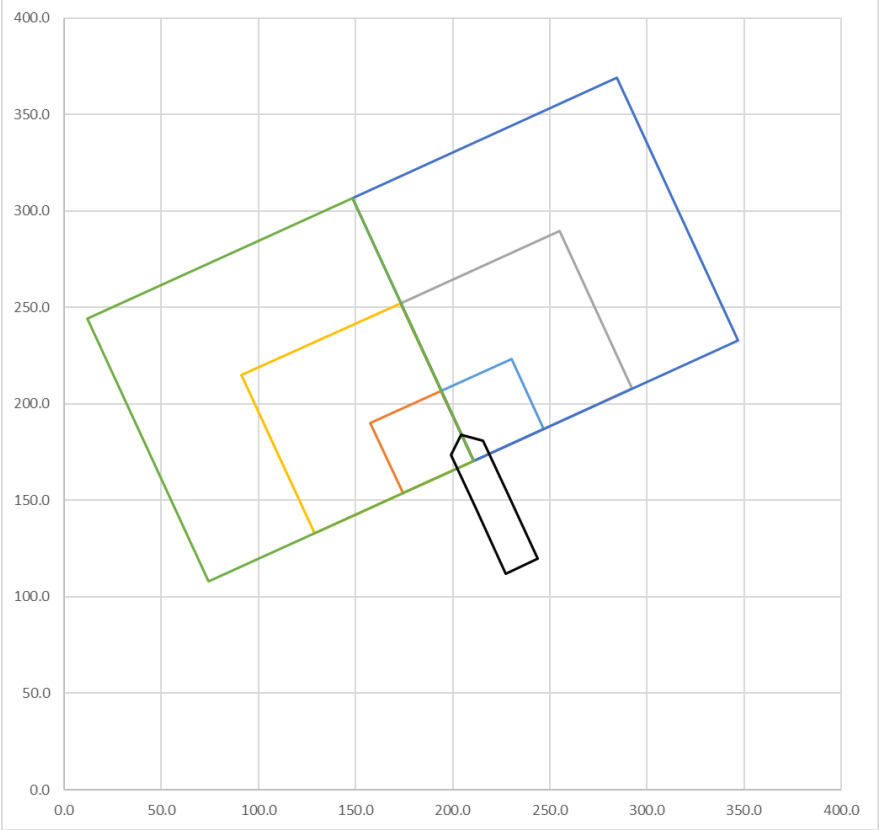


Figure 6: First iteration of zones

In Figure 7, the zones are rearranged into a radial-style pattern. This informs the model as to whether there is more ice to port or starboard in a way that is better suited to facilitate informed decisions for rudder control. One downside with this layout, as well as the previous layout, is that it provides very little information directly port, starboard, or aft of the ship. This isn't problematic when the vessel is heading directly towards the destination, but it can be a bigger problem if the ship needs to make a sharp turn at any point during its travel. This could be mitigated by either

adding additional zones or widening existing zones, however, this layout also has the downside of using a total of eight zones for a total of 13 state parameters.

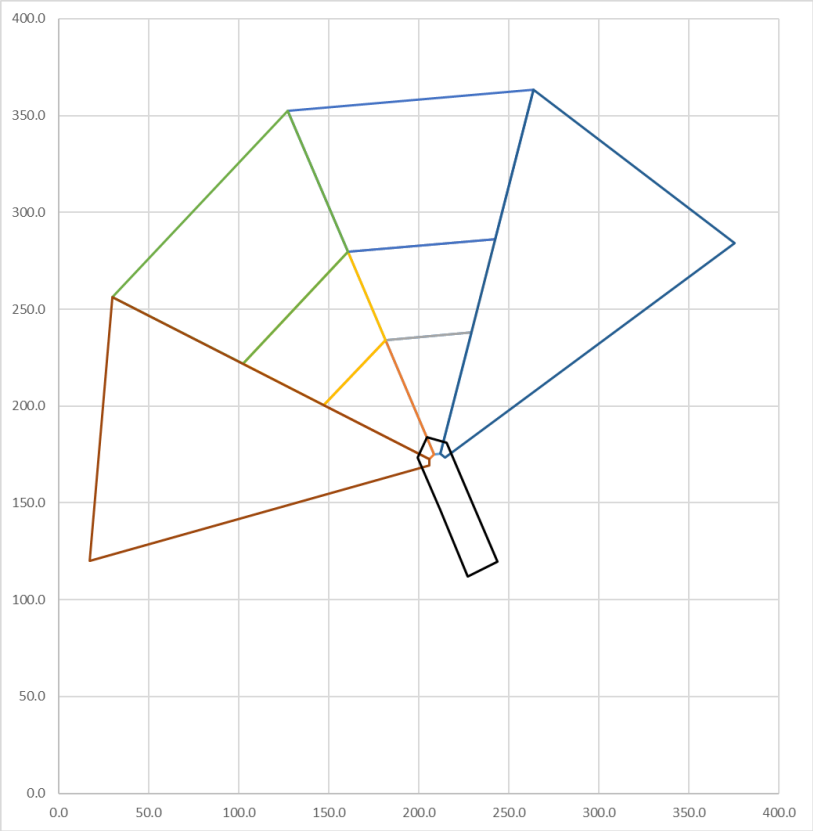


Figure 7: Second iteration of zones

In Figure 8, only four zones are used. This simplifies the trends that need to be learned by the model. Another benefit of this layout is that it discretizes the data more than the above examples. That is, there are no small zone areas in which the vessel is being provided fairly precise information. While this has the disadvantage of providing less information for decision making, it makes the model less susceptible to imprecise data collection. This is because the decisions made by the model would be based on the general state of ice within the various zones rather than the precise size and location of ice floes. As discussed in section 2.3, having a state representation that could be obtained in the intended setting, rather than just in the simulator, is crucial when it comes to applying the trained model to the real world. The section also discusses the limitations regarding

active data collection of ice on board a vessel. While these issues still need to be overcome for this zone configuration, the imprecise nature of the larger zones lends itself to being transferrable with less precise data collection.

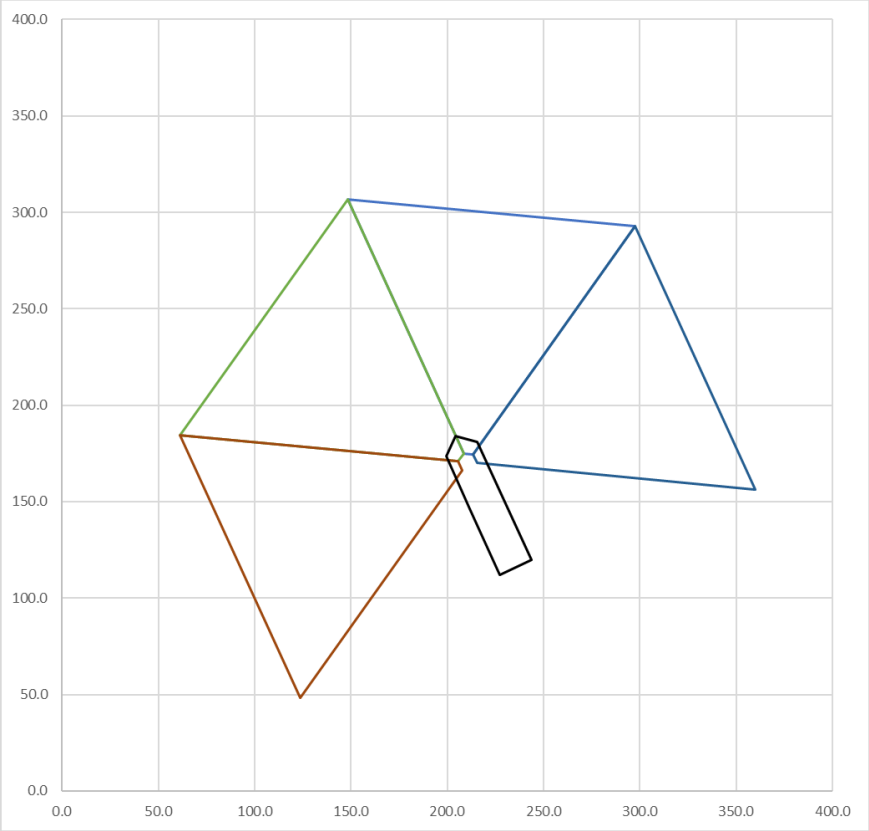


Figure 8: Selected application of zones

Ultimately, there are infinite possible layouts that could be considered; these three designs demonstrate some of the considerations of layout selection. In this work, layout 3 (shown in Figure 8) is applied for the learning model. The downsides relating to less precise information for the learning algorithm are considered worthwhile given the probable much faster and easier learning process in addition to facilitating more realistic application in the real-world equivalent.

3.3.5 Model Training

There are a multitude of different reinforcement learning algorithms that can be used to train a model. This work does not intend to further develop any existing algorithms. Furthermore, since it is a proof of concept, determining the perfect algorithm is also not a key concern. Here, the focus is simply on selecting an appropriate algorithm for the problem that can demonstrate the ability to find an acceptable solution to the problem within a reasonable timeframe. As such, only two learning algorithms are discussed due to the benefits they provide when being applied in a continuous state space: TD learning with tile coding and proximal policy optimization (PPO).

3.3.5.1 TD(λ) with Tile-Coding

Temporal-difference learning with tile coding (described in depth in section 3.2.7) is initially applied here but deemed sub-optimal due to its poor training time requirement in comparison to PPO. While TD learning is shown to be effective in the simpler learning environment (see section 3.2), the updates become computationally expensive when applying it to a larger state space with higher dimensionality. As such, it is excluded in favor of PPO for this section of work.

3.3.5.2 Proximal Policy Optimization (PPO)

Proximal policy optimization is a policy gradient method that uses an advantage function. The advantage is a value that denotes whether a performed action is better or worse than expected along with the size of the difference. It is calculated by taking the difference between the weighted sum of the rewards and a baseline estimate. The baseline estimate is the model's prediction of the sum of the rewards that will be acquired over the remainder of the trial. This can alternatively be viewed as the estimate of how good it is to be in the current state. Multiplying this advantage function, as a gradient, by the current policy – or the log of the policy in the case of PPO – gives

an update to the policy towards a better policy based on the achieved results. This direct update of the policy is why PPO is considered to be a policy gradient method [26]. A key feature of PPO is the surrogate objective function. This is a function applied to the policy updates to maintain regular updates by keeping the updates within a close region of the current policy using data from the previous iteration of the policy [44]. This concept is not unique to PPO; in fact, PPO is an expansion of Trust Region Policy Optimization (TRPO) algorithms that also apply this concept [45, 46]. PPO is applied in this work since it performs well with complex state spaces with delayed rewards and is easy to implement [26, 47].

3.3.5.2.1 Hyperparameter Tuning

Reinforcement learning algorithms use parameters that can be adjusted when initializing the model. These parameters affect the updates made by the model for adjusting to new information. The values assigned to the hyperparameters can affect the model's ability to properly converge to a solution as well as the amount of training required for that convergence to occur. For PPO, the most relevant hyperparameters that are tuned for this work are gamma, lambda, entropy coefficient (ent_coef), number of steps (n_steps), and learning rate (learning_rate). The size of the neural network is another parameter that can be adjusted. The suggested default values are initially applied and adjusted as necessary to improve the training process.

3.3.6 Reward Function

The reward function reflects the combination of factors that affect whether or not a behavior is considered good or bad. This problem has a similar overall objective to that of the simple environment: reaching the destination quickly. For the simplified environment, this is complicated by the addition of constant thickness ice for each scenario. In the GEM environment, sparse ice is used instead with the intent being to avoid the ice completely. One of the challenges

involved with developing a reward function for this model is that the penalty for hitting ice needs to be large enough that the model knows to avoid ice at all costs. On the other hand, if the penalty is made too large, then the model will have a hard time learning to reach the destination at all since the chance of hitting ice is not worth the penalty incurred.

One way to encourage the model to reach the destination despite the possibility of collision is to improve the exploratory properties of the model. This can be done through either random exploration, or by more methodically rewarding behaviors that have not been tried before. One concern in this case is that the model would still struggle to take the correct path enough times to learn the required complex trends. Instead, a tiered learning approach is applied in order to ensure the model's ability to deal with the conflicting rewards. As discussed in section 2.2.5, phased learning takes place in many environments. Here, this concept is used in order to specifically train the model one phase at a time. The problem is broken down into the primary tasks that need to be accomplished and then trained cumulatively. The model is trained to reach the destination using one reward function before modifying the reward function to include ice avoidance and training the model further. It is important to consider the problem before attempting such a method as the method can greatly affect the way that the model learns, and, therefore, the final outcome of the training. Here, it is considered critical that the model reaches the destination with it being impossible to have a successful run without reaching the destination. On top of that, the vessel is supposed to avoid ice entirely, presented to the model by a large penalty any time a collision occurs. Attempting to train both of these objectives at once makes it easy for a local optimum to be reached where the vessel never reaches the destination. This is particularly true here due to how difficult it is for a naïve model to reach the destination while it is very easy to have multiple collisions with ice floes.

The first training uses a reward function that ignores the presence of ice entirely – though it is still present in the state space – and focuses entirely on progress made towards the destination as per the following

$$R = (-10 + d_0 - d_1)/10. \quad (3.9)$$

This reward structure is similar to the one used for the simplified environment with the reward being based on the distance between the vessel and the destination before and after the results of the action. A value of -10 is applied since the maximum progress that can be made in any given timestep is a little less than 10 and the overall reward is intended to remain negative so that it is better to reach the destination as quickly as possible. A significant additional reward,

$$R \leftarrow R + 100, \quad (3.10)$$

is provided if the destination is reached to ensure that the model makes reaching the destination a priority. The second set of training uses the same reward structure as the first set but adds a substantial negative reward of

$$R \leftarrow R - 10 \quad (3.11)$$

to any timestep that involves a hull load from collision with an ice floe so that the model learns to avoid ice floes on the way to the destination.

3.3.7 Time Tradeoffs

Certain tradeoffs are implemented such that the model is able to learn within a reasonable amount of time. At every step, calculations are performed based on the number of ice floes. Due to this, overall ice field is kept as a 4000m square for each scenario with an overall ice

concentration never being more than 2 percent during training. Additionally, individual ice floes are never less than 140m^2 and the ice floes do not break apart due to impact.

3.3.8 Model Validation

Similar to the validation criteria outlined in section 3.2.9, a set of criteria are used to gauge the success of the model. These criteria aim to encapsulate the requirements mentioned in section 3.3.2. The number one priority is that the vessel reaches the destination. This is a critical requirement so it is deemed that the destination must be reached 100% of the time. While there could be instances of decent behaviors where the destination is not reached, enough time has been allotted that it is perfectly reasonable for the destination to be consistently reached and, therefore, any occurrences of failure for this criterion are indicative of an issue with the model. For the GEM environment, the ice conditions are more complex than they are for the simple environment that used constant ice thicknesses. Therefore, success criteria that deal with the ice are also more complex. Here, the aim is to have the vessel avoid the ice entirely on its path to reach the destination. Given some of the types of test scenes, occasional collisions are deemed acceptable. This is, in part, because the intent is for the vessel to make minor adjustments to the heading in order to navigate around the floes. In many instances, multiple floes generate a barrier that cannot be avoided with minor course adjustments alone. These instances are not currently a priority as these situations are typically navigable without serious hull loads by passing between the floes and only incurring glancing blows to the hull. Another consideration for setting the validation criteria for ice collisions is that a greater number of ice collisions are permissible over longer distances. In other words, the number of allowable collisions should be proportional to the distance between the starting location of the vessel and the destination. A similar proportionality is considered for the ice density as the density directly corresponds to areas that need to be avoided. As such, the

criterion for this is set as the ice density percentage times the distance to destination of kilometers (rounded down) plus one.

$$\textit{Number of collisions} \leq \textit{floor(Ice Density [\%]*|Distance to destination [km]| + 1)}. \quad (3.12)$$

A more subtle criteria checked during validation is that the vessel should take a fairly direct route to reach the destination with only minor adjustments made to avoid ice. This is a visual check that ensures that the model has not learned strange behavior such as taking unnecessarily wide arcs to reach the destination or weaving back and forth. These or similar behaviors could potentially perform relatively well while being unacceptable from a practical standpoint. In addition to this being checked visually, with a minimum average speed of 2.2 m/s and an average of maximum speeds of at least 2.5 m/s. This can only be reasonably be met by taking a reasonably direct route at moderate speeds.

4 Results

4.1 Experiment #1: Simple Environment

As can be seen in Figure 9, the vessel took a logical route to the destination. The path is not completely direct as some curvature exists over most of the path (i.e. the path is not perfectly straight). However, the model does manage to adjust the vessel heading towards the destination and then proceed to follow a fairly direct path. The curvature is also minor enough that it has minimal effect on the overall time taken to reach the destination.

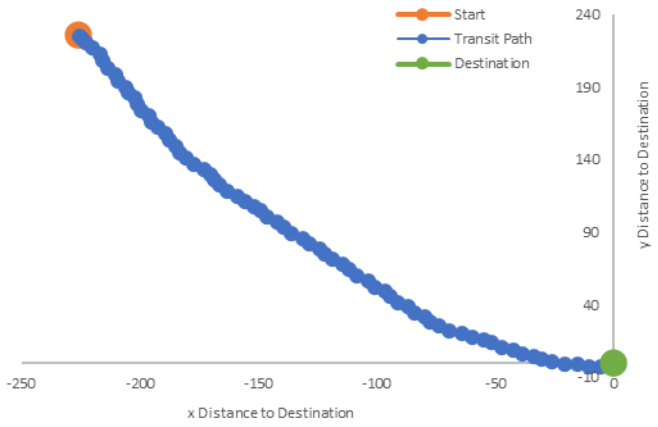


Figure 9: Example vessel path (0.5 ice condition)

This trial is in the 0.5 ice condition for which the maximum allowable speed is 6.75 (Table 3). The model manages to keep the ship's speed below that threshold throughout the entire trial. This behavior provides an initial indication that the model is functioning effectively.

Table 3: Example trial data (0.5 ice condition)

Max Allowable Speed	6.75
Average Speed	5.15
Maximum Speed	6.52
Number of Damage Instances	0
Ice Thickness	0.5

Table 4, Table 5, Table 6, Table 7, and Table 8 show the results of the trials for each of the different ice conditions. As a whole, it can be seen that the criteria are successfully met with a single notable exception in the 1.5 ice condition – discussed in detail below. Across all of the conditions, the destination is reached 100% of the time. In the 0.0 ice condition (see Table 4), the ship quickly accelerates and proceeds to the destination at speeds above those that are required by the criteria. There are obviously no damage instances during these trials as it is impossible to incur damage without ice. The 0.5 condition shows similar results with no damage instances and speeds cleanly above the minimum requirements for both the overall average speed and average of maximum speeds across the 100 trials. Here, there are also no damage instances as the model manages to keep the ship travelling below speeds that would incur damage throughout the entire trial for all 100 runs.

Table 6 shows the results for the trials in the 1.0 ice condition. Here, all of the criteria are passed, though not quite as cleanly as in the 0.5 ice condition as four damage instances occur throughout the runs. This is still well within the acceptable criteria for the proof of concept, but it is worth keeping in mind as it indicates room for improvement. The 1.5 ice condition, shown in Table 7, once again passes all of the criteria with speeds above the minimum threshold and no damage instances across all trials. Finally, the 2.0 ice condition is the only ice condition where one of the criteria is not met. As can be seen in Table 8, the ship reaches the destination while travelling at appropriate speeds for most of the trials. However, there are a total of 77 timesteps where the ship is travelling too quickly and, therefore, obtains hull damage. This remains a small portion of the overall timesteps, with many of the damage instances occurring in quick succession as the ship remains above the acceptable speed for consecutive timesteps. That said, it is well above the

acceptable number of allowable damage instances for this criterion, indicating that the model is not fully capable of controlling the speed of the vessel within the appropriate range.

Table 4: Summary data for 100 trials (0.0 condition)

Objective	Requirement	Actual	Pass (Y/N)
Percent Successful Completions	>95%	100%	Y
Total Damage Instances	≤10	0	Y
Overall Average Speed	>4.88	6.65	Y
Average of Maximum Speeds	>6.83	7.67	Y

Table 5: Summary data for 100 trials (0.5 condition)

Objective	Requirement	Actual	Pass (Y/N)
Percent Successful Completions	>95%	100%	Y
Total Damage Instances	≤10	0	Y
Overall Average Speed	>3.38	5.18	Y
Average of Maximum Speeds	>4.73	5.97	Y

Table 6: Summary data for 100 trials (1.0 condition)

Objective	Requirement	Actual	Pass (Y/N)
Percent Successful Completions	>95%	100%	Y
Total Damage Instances	≤10	4	Y
Overall Average Speed	>3.00	4.21	Y
Average of Maximum Speeds	>4.20	5.1	Y

Table 7: Summary data for 100 trials (1.5 ice condition)

Objective	Requirement	Actual	Pass (Y/N)
Percent Successful Completions	>95%	100%	Y
Total Damage Instances	≤10	0	Y
Overall Average Speed	>2.38	3.08	Y
Average of Maximum Speeds	>3.33	4.24	Y

Table 8: Summary data for 100 trials (2.0 ice condition)

Objective	Requirement	Actual	Pass (Y/N)
Percent Successful Completions	>95%	100%	Y
Total Damage Instances	≤10	77	N
Overall Average Speed	>1.50	1.82	Y
Average of Maximum Speeds	>2.10	2.81	Y

4.2 Experiment #2: GEM Environment

The results demonstrate that the model is able to control the vessel to consistently reach the destination through a fairly direct route. At the start of the scenario, a sharp turn is taken to adjust the relative bearing to be roughly zero – in other words, if the vessel is not initially pointing towards the destination, an adjustment is made over the first series of steps to bring the vessel around such that it is heading in the correct direction. After that, the thrust is increased to reach the destination quickly with minor adjustments being made to allow the vessel to narrowly avoid hitting the ice floes.

Figure 10, Figure 11, Figure 12, and Figure 13 show sample paths taken by the vessel for each type of scenario. A total of five example routes for each type of scenario are presented in Appendix I. While the routes may not be perfect, they are close to direct routes that avoid crashing into ice. In some cases, a very small gap between the vessel and the ice flows is present – with the vessel incurring minor collisions in some instances. This is largely due to the coarseness of the state parameters – specifically the ice zones. Since the model knows which zones ice is in rather than the exact locations of the ice floes, it is incapable of perfect avoidance.

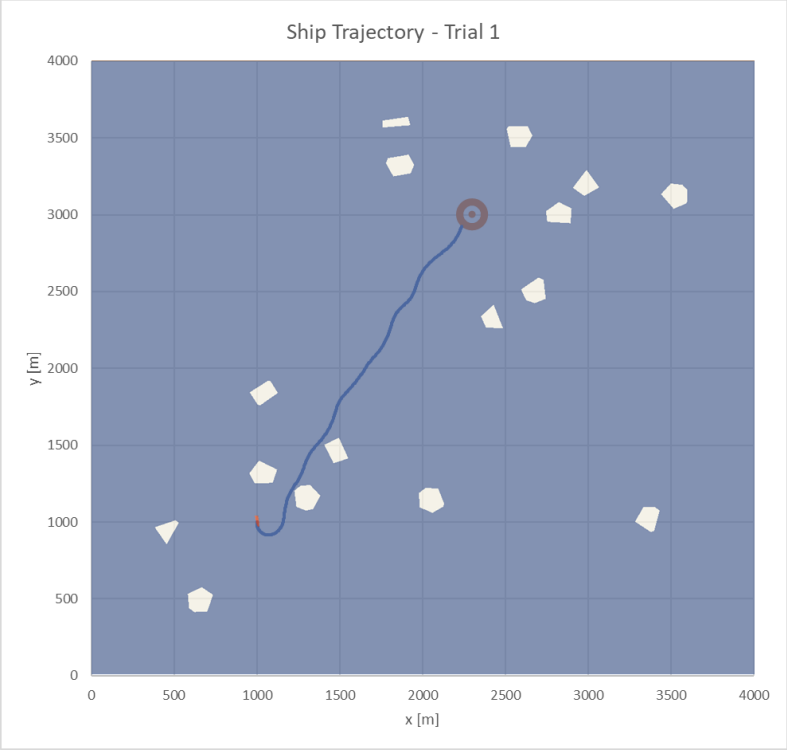


Figure 10: Example Ship Path – 2% Coverage (Trial 1)

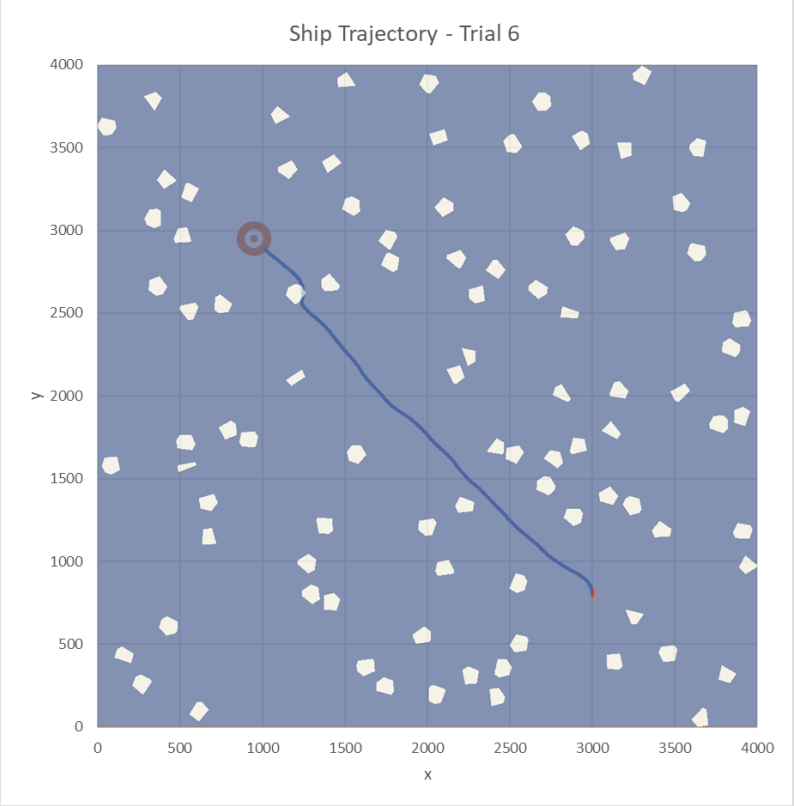


Figure 11: Example Ship Path – 5% Coverage (Trial 6)

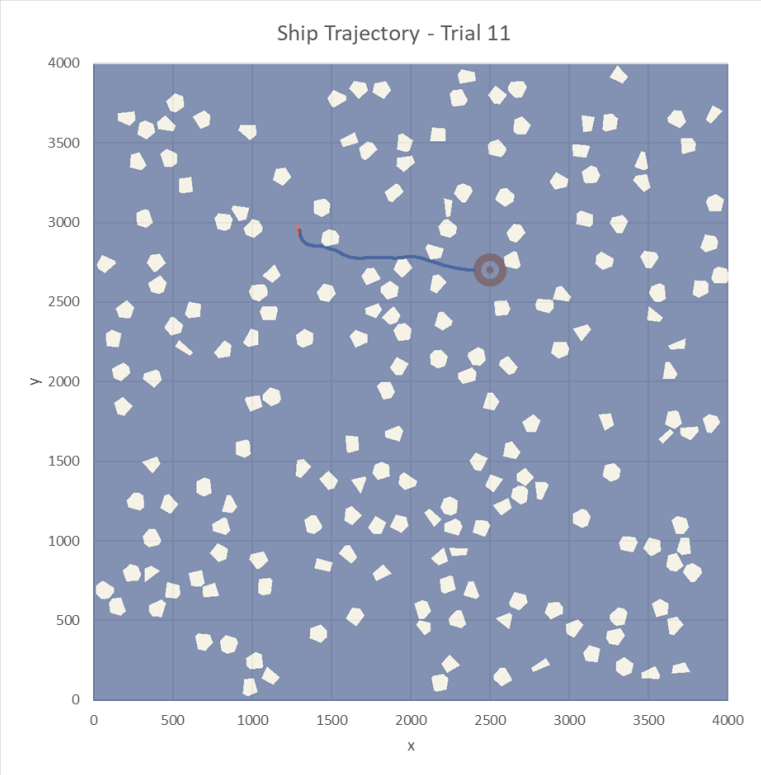


Figure 12: Example Ship Path – 10% Coverage with large floes (Trial 11)

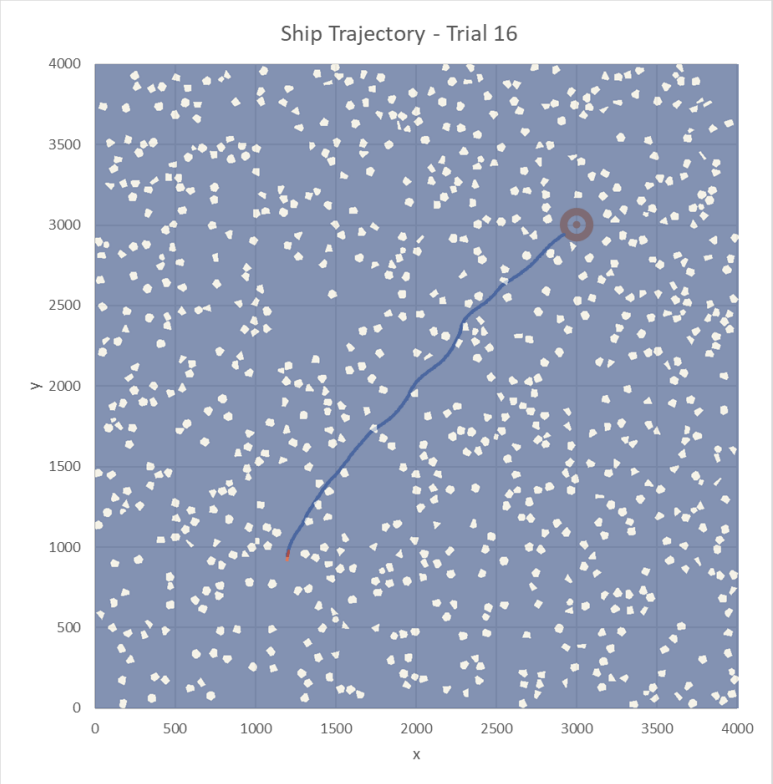


Figure 13: Example Ship Path – 10% coverage with small floes (Trial 16)

In many instances, the vessel narrowly misses an ice floe, or even barely glances off a floe in a fashion that does not trigger GEM to treat the impact as a collision. Similar collisions occurred during the training phases as well. Since these did not incur a force on the hull, they are not considered problematic for the reinforcement learning algorithm. That said, a ship glancing off of an ice flow at high speeds is, obviously, a concern. This can be fixed by adjusting the calculation in GEM to ensure that such instances result in a hull load being generated. For this work, a load not being generated is not directly problematic for the validation of the method. Any time such a glancing blow occurs, the model would not receive a penalty and no collision is recorded. As long as the results also do not treat this as a collision, then the environment remains consistent with the way that it is being applied. The intent is for the model to learn that traveling very near ice is dangerous and, therefore, allowing for a bit of space between the vessel and the ice is a good idea in order to consistently obtain a good overall reward. Having it so that the ice is only usually a negative factor makes this more complex. However, the general trend remains the same, and the vessel should still be steered away from ice to avoid collision. The ideal behavior remains almost unchanged – especially given the imprecise state space representation for the ice floes – and, as a result, the validation of the method through application within GEM is still valid.

The model fully passed the validation criteria of reaching the destination 100 percent of the time with no failures in reaching the destination in the 100 test runs for any of the four test scenario types (Table 9). While this does not demonstrate the ability of the model to navigate ice, it does illustrate a reasonable level of consistent control of the vessel that is necessary for navigation. The destination is also reached adequately fast in all cases once again passing the criteria.

Table 9: Non-Collision Based Result Summary

Scene Type [-]	Destination Reached [-]	Average Trial Distance [km]	Average of Max Speed [m/s]	Average of Average Speeds [m/s]
2%	100	1.997	2.89	2.70
5%	100	2.123	2.89	2.67
10% (large floes)	100	2.072	2.89	2.60
10% (small floes)	100	2.113	2.89	2.62

The collision results are not as simple. It is required that the number of collisions are kept to a minimum based on the overall distance between the start and end point for each trial. For the 2% scenario, the average distance travelled per trial is 1.997km. This equates to an average of four allowable collisions per trial. As shown in Table 10, the average number of collisions per trial are well below this value at only 1.09; however, eight individual trials still exceeded the allowable threshold for that particular trial. This is primarily due to individual collisions with ice floes resulting in collisions being triggered across several consecutive time steps. This could be further refined by adjusting the way that ice collisions are recorded or by penalizing the model differently to further discourage such collisions. As the ice density is increased, the number of collisions also grow. This is expected since there are an increased number of floes to avoid on the way to the destination. Specifically, the overall average number of allowable collisions per trial are 11, 21, and 22 for the 5%, 10% (large floes), and 10% (small floes) scenarios respectively. These average values are cleanly met for each scenario type. Similar to the 2% trial, however, there are a number of individual trials that surpass the criteria. The 10% (large floes) scenario is particularly problematic in this area with a total of 19 trials – almost 5 percent of the total number of trials – failing to meet this criterion.

Table 10: Collision Based Result Summary

Scene Type [-]	Total Number of Collisions [-]	Average Number of Collision Per Trial [-]	Average Number of Collisions per km [collisions/km]	Number of trials with Excessive Collisions [-]
2%	109	1.09	0.55	8
5%	380	3.8	1.79	9
10% (large floes)	1176	11.76	5.68	19
10% (small floes)	751	7.51	3.55	3

5 Discussion

5.1 Experiment #1: Simple Environment

Ideally, the model would be capable of passing all criteria for every ice condition being considered; however, the level of success presented above acceptably demonstrates the efficacy of the model for the proof of concept of the solution. As described in section 3.2.2, this setup intends to demonstrate the ability of a reinforcement based model to overcome a few of the major hurdles with navigation in level ice. The results show that the model was able to effectively reach the destination consistently without incurring loads that are considered harmful for the vast majority of the trials. There is a single failed criteria that occurs for the 2.0 ice condition. It is understandable that the issue occurred for the thickest ice condition since it provided the most sensitive thrust action decisions. The water resistance on the vessel is relative to the speed of the vessel. As such, minor changes in the thrust have a greater impact on the ship's speed when travelling at slower speeds. Furthermore, there are only a total of four thrust actions to choose from and only two of these are forward thrust actions. Combining these factors together results in a fairly precise thrust-action requirement when travelling at slower speeds – a necessity when dealing with thicker ice. If the model selects only the slower of the two thrust actions as the only thrust action, the vessel will still reach a speed of 7.15. This is over double the maximum allowable speed for the 2.0 ice condition. This could be dealt with by redesigning the action space with more thrust actions, thereby providing better control of the ships speed; however, A larger number of possible actions would make the problem more complex for which more training would be required.

5.2 Experiment #2: GEM Environment

For this environment, the tests are performed with the intent of avoiding ice on the way to the destination in sparse ice conditions. This is demonstrated to be an achievable goal with the vessel consistently reaching the destination through fairly direct means with minimal or no collisions. As discussed in section 4.2, there are occasional instances of collisions between the ship and the ice. While this indicates that the model is imperfect, the overall intended trends are, nonetheless, achieved. The destination is reached consistently and efficiently across all of the tested scenarios. Ice collisions demonstrate room for improvement, but the intention of the criteria is met with minimal or no collisions taking place during the majority of the trials. The primary criteria of concern is the number of trials that exceed the allowable number of collisions. Even though the overall average number of collisions was substantially lower than the overall averaged criteria – by roughly a factor of three – a notable number of scenarios still failed individually. This is due to a large gap in collision count between the scenarios that passed the criteria and the ones that failed. This gap is a result of consecutive collisions with ice in many of the trials. There are a number of trials where the ship incurred glancing collisions with ice resulting in only one, or at least very few, collisions being recorded. By contrast, more direct collisions between the vessel and ice resulted in a very large number of consecutive collisions being recorded. While this means that the trials that had a large number of collisions only truly collided with one or two floes on the way to the destination, it also indicates that these collisions are major collisions that are considered to be significant failures on the part of the model.

6 Conclusion

The problem of automated ship navigation in ice involves a range of hurdles that need to be overcome. Ship navigation in general is relatively complex given that it takes several seconds after taking an action before the effect of the action is truly seen. This form of delayed response creates a complex controls problem. In addition to the basic navigation problem, interactions with ice further complicates the scenarios. The scope for this works is limited to two separate scenarios: transit through constant ice and avoidance of ice flows in sparse ice conditions. For both environments the desired outcome is the same; reach a specified destination without incurring hull damage. The applied reinforcement learning based solutions demonstrate the ability to learn from simulated environments well enough to come to a reasonable solution.

This work is intended to act as a proof of concept of the ability for reinforcement learning algorithms to be applied to the ship navigation in ice problem. As such, the models derived here are not intended to be the final product. Specific requirements for each environment are specified as to validate that the trained models are sufficiently capable as to consider the method validated. For both the simple environment and the GEM environment, the intent is to reach the destination as quickly as possible without having any incidents of hull damage; however, the exact purpose of the two environments varies. The simple environment is used to demonstrate the ability of a reinforcement learning based model to control a ship by adjusting thrust and rudder angle to get from a start point to an end point. This is performed with the added restriction of maintaining a speed below a threshold that represents damage to the hull. For the GEM environment, sparse ice floes are used instead of constant thickness ice with the purpose of developing a model capable of avoiding ice collisions entirely. While neither problem is perfectly solved, the trained models perform sufficiently well as to consider this work to be a successful proof of concept.

6.1 Future Work

There are numerous directions that are viable for progression of this work. A logical first step is to combine the two types of models developed within this work. One of the models is developments to manage speeds for sheet ice conditions while the other is aimed at avoiding ice floes. A follow up to this would be to train a model capable of navigating through ice floes where avoidance is possible, but not always necessary. This would be a combination of the concepts discussed in this work using sparse ice floes of varying ice thickness such that the vessel's speed dictates safe collisions based on ice thickness.

Another method for development of this work would be to expand upon the types of scenarios being considered. Expanding the range of scenarios to include varying ice thicknesses would demonstrate wider capability. This work deals with consistent ice thicknesses throughout the individual runs. As such, the model can safely assume that the ice encountered in the future will be comparable to whatever is present in any given state. Having a range of ice thicknesses within the scenarios would result in a more generalizable model capable of adapting to more situations – a must for proper application. Similar expansions could be made in areas such as the range of ice thicknesses considered or the range of ice densities for the sparse ice conditions.

Additionally, more complicated zone layout would allow for more detailed information to be passed to the learning algorithm for more informed decisions; however, as discussed in section 3.3.4.1, the zone layout should be representative of the information that can be achieved. At this stage, feasibility of real-time state space representation in the real world is considered, but not studied in extensive depth. Delving deeper into the exact sensor capabilities – potentially including a real-world study – would help to determine the most appropriate zone representation.

Finally, an alternative area where improvements could be made is in the capabilities of the simulator. There are limitations with the methods applied in this work due to the restrictions of the programs being used, such as icebreaking not being possible. As discussed, the realism of the simulator governs the applicability of the trained model to real world settings. Improving the realism of the simulators increases the useability of the research and models involved.

7 References

[1] Q. Jiangyuan, "Simulation of auto obstacle avoidance based on Unity machine learning," *Journal of Physics: Conference Series,* vol. 1883, no. 1, 2021.

[2] J.-M. Choi, S.-J. Lee and M. Won, "Self-learning navigation algorithm for vision-based mobile robots using machine learning algorithms," *Journal of Mechanical Science and Technology,* vol. 25, no. 1, pp. 247-254, 2011.

[3] E. Thorn, S. C. Kimmel and M. Chaka, "A framework for automated driving system testable cases and scenarios," National Highway Traffic Safety Administration, Washington, DC, 2018.

[4] J. D. Lee, A. R. Pritchett and E. M. Roth, "Perspectives on Automotive Automation," *Journal of cognitive engineering and decision making,* vol. 12, no. 1, pp. 53-57, 2018.

[5] Norwegian Forum for Autonomous Ships, "Definitions for Autonomous Merchant Ships," 2017.

[6] Y. Wang, S. Chai and H. D. Nguyen, "Unscented Kalman Filter trained neural network control design for ship autopilot with experimental and numerical approaches," *Applied Ocean Research,* vol. 85, pp. 162-172, 2019.

[7] T. Lauvdal and T. I. B. M. Fossen, "Robust adaptive ship autopilot with wave filter and integral action," *International Journal of Adaptive Control and Signal Processing,* vol. 12, no. 8, pp. 605-622, 1998.

[8] A. J. Koshkoue, Y. Law and K. J. Burnham, "Sliding mode and pid controllers for ship roll stabilisation: a comparative simulation study," *IFAC Proceedings Volumes,* vol. 38, no. 1, pp. 7-12, 2005.

[9] S. Bhattacharya, S. Verma and S. Mukhopadhyay, "Comparative study of variants of Kalman filter aided by artificial neural networks for improved kinematic state estimation of air-borne vehicles," in *India Annual Conference*, Kharagpur, 2004.

[10] A. Nadzilah, D. M. Gandana, J. Muliadi and Y. Daryanto, "Application of Kalman filter to track ship maneuver," in *5th International Conference on Cyber and IT Service Management (CITSM)*, Denpasar, 2017.

[11] J. Velagic, Z. Vukic and E. Omerdic, "Adaptive fuzzy ship autopilot for track-keeping," *Control Engineering Practice,* vol. 11, no. 4, p. 433–443, 2003.

[12] G. W. Timco and K. R. Croasdale, "How well can we predict ice loads," in *Proceedings of the 18th IAHR International Symposium on Ice*, 2006.

[13] M. Kotilainen, J. Vanhatalo, M. Suominen and P. Kujala, "Predicting ice-induced load amplitudes on ship bow conditional on ice thickness and ship speed in the Baltic Sea," *Cold Regions Science and Technology,* vol. 135, pp. 116-126, 2017.

[14] T. Lundamo, B. Bonnemaire, A. Jensen and O. T. Gudmestad, "Back-Calculation of the Ice Load Applying on a Moored Vessel," in *19th IAHR International Symposium on Ice*, Vancouver, British Columbia, 2008.

[15] J. Liu, M. Lau and F. M. Williams, "Numerical Implementation and Benchmark of Ice-Hull Interaction Model for Ship Manoeuvring Simulations," in *19th IAHR International Symposium on Ice*, Vancouver, British Columbia, 2008.

[16] R. S. Sutton and A. G. Barto, "Introduction," in *Reinforcement Learning: an introduction*, second ed., Cambridge, MA, The MIT Press, 2018, pp. 1-13.

[17] R. S. Sutton and A. G. Barto, in *Reinforcement Learning: an introduction*, second ed., Cambridge, MA, The MIT Press, 2018.

[18] W. D. Smart and L. P. Kaelbling, "Practical Reinforcement Learning in Continuous Spaces," in *ICML*, 2000.

[19] R. S. Sutton and A. G. Barto, "Planning and Learning with Tabular Methods," in *Reinforcement Learning: An Introduction*, second ed., Cambridge, MA, The MIT Press, 2018, pp. 159-194.

[20] C. Szepesvári, "Algorithms for large state spaces," in *Algorithms for Reinforcement Learning*, Morgan & Claypool, 2010, pp. 18-36.

[21] S. Gu, T. Lillicrap, I. Sutskever and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*, 2016.

[22] J. Fan, Z. Wang, Y. Xie and Z. Yang, "A theoretical analysis of deep Q-learning," *arXiv preprint arXiv:1901.00137v3,* 2020.

[23] R. S. Sutton and A. G. Barto, "Tile Coding," in *Reinforcement Learning: An Introduction*, second ed., Cambridge, MA, The MIT Press, 2018, pp. 217-220.

[24] H. v. Hasselt and M. A. Wiering, "Reinforcement Learning in Continuous Action Spaces," in *2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007.

[25] H. van Hasselt, "Reinforcement learning in continuous state and action spaces," in *Reinforcement Learning: State-of-the-Art*, vol. 12, Berlin, Heidelberg, Springer, 2012, pp. 207-251.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347,* 2017.

[27] M. Tokic, "Adaptive ε-Greedy Exploration in Reinforcement Learning Based on Value Differences," in *Annual Conference on Artificial Intelligence*, Berlin, Heidelberg, 2010.

[28] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew and I. Mordatch, "Emergent Tool Use from Multi-Agent Autocurricula," in *International Conference on Learning Representations*, 2020.

[29] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu and C.-Y. Lee, "Diversity-Driven Exploration Strategy for Deep Reinforcement Learning," in *Conference on Neural Information Processing Systems*, Montréal, Canada, 2018.

[30] O. Nachum, M. Norouzi, K. Xu and D. Schuurmans, "Bridging the Gap Between Value and Policy Based Reinforcement Learning," in *Conference on Neural Information Processing Systems*, CA, USA, 2017.

[31] P. Dangeti, "Reinforcement learning basics," in *Statistics for Machine Learning*, Birmingham, Mumbai, Packt Publishing Ltd, 2017, pp. 361-368.

[32] J. Peters, "Policy gradient methods," *Scholarpedia,* vol. 5, no. 11, p. 3698, 2010.

[33] H. Kim, O. I. Haugen, B. Rokseth and M. A. Lundteigen, "Safety Verification for Autonomous Ships," in *MATEC Web of Conferences*, 2019.

[34] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu and R. Hadsell, "Sim-to-Real Robot Learning from Pixels with Progressive Nets," *Conference on Robot Learning,* pp. 262-270, 2017.

[35] R. Birch, D. Fissel, H. Melling, K. Vaudrey, K. Schaudt, J. Heideman and W. Lamb, "Ice-profiling sonar," *Sea Technology,* vol. 41, no. 8, pp. 48-54, 2000.

[36] J. M. Killen and J. S. Gulliver, "The Use of SONAR to Measure Ice Thickness," 1991.

[37] D. Fissel, R. Chave and J. Buermans, "Real-Time Measurement of Sea Ice Thickness, Keel Sizes and Distributions and Ice Velocities Using Upward Looking Sonar Instruments," *OCEANS,* pp. 1-6, 2009.

[38] S. M. Pershin, V. N. Lednev, V. K. Klinkov, R. N. Yulmetov and A. F. Bunkin, "Ice thickness measurements by Raman scattering," *Optics Letters,* vol. 39, no. 9, pp. 2573-2575, 2014.

[39] J. C. Landy, A. S. Komarov and D. G. Barber, "Numerical and experimental evaluation of terrestrial LiDAR for parameterizing centimeter-scale sea ice surface roughness," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 53, no. 9, pp. 4887-4898, 2015.

[40] Y. Yan, Z. Tan and N. Su, "Sea-ice scene classification using aerial images in arctic based on transfer learning," in *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018.

[41] R. J. Hall, N. Hughes and P. Wadhams, "A systematic method of obtaining ice concentration measurements from ship-based observations," *Cold Regions Science and Technology,* vol. 34, no. 2, pp. 97-102, 2002.

[42] M. Dawson, M. Musharraf, D. Peters, S. Power, B. Veitch and C. Daley, "Temporal-Difference Learning for Ship Navigation in Ice using Tile Coding," in *Newfoundland Electrical and Computer Engineering Conference (NECEC)*, St. John's, 2019.

[43] C. Daley, "User's Manual for the GEM Simulation Program," 2017.

[44] C. C.-Y. Hsu, C. Mendler-Dünner and M. Hardt, "Revisiting Design Choices in Proximal Policy Optimization," *arXiv preprint arXiv:2009.10897,* 2020.

[45] J. Schulman, S. Levine, P. Abbeel, M. Jordan and P. Moritz, "Trust Region Policy Optimization," in *32nd International Conference on Machine Learning*, 2015.

[46] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph and A. Madry, "Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO," *arXiv preprint arXiv:2005.12729,* 2020.

[47] N. Vanvuchelen, J. Gijsbrechts and R. Boute, "Use of proximal policy optimization for the joint replenishment problem," *Computers in Industry,* vol. 119, p. 103239, 2020.

[48] B. Myhre, A. Hellandsvik and S. Petersen, "A responsibility-centered approach to defining levels of automation," *Journal of Physics: Conference Series,* vol. 1357, no. 1, p. 012027, 14 11 2019.

[49] B. Rokseth, O. I. Haugen and I. B. Utne, "Safety Verification for Autonomous Ships," in *MATEC Web of Conferences*, 2019.

[50] M. S. Martis, "Validation of simulation based models: a theoretical outlook," *The electronic journal of business research methods,* vol. 4, no. 1, pp. 39-46, 2006.

[51] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. Pondé

de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski and S. Zhang, "Dota 2 with Large Scale Deep Reinforcement Learning," *arXiv preprint arXiv:1912.06680,* 2019.

[52] S. Idrissova, P. Kujala, R. Repin and F. Li, "The study of the Popov method for estimation of ice loads on ship's hull using full-scale data from the Antarctic sea," in *Proceedings of the 25th International conference on Port and Ocean Engineering under Arctic Conditions*, Delft, The Netherlands, 2019.

Appendix A – 100 Run Scenario Information (2% coverage)

Scene #	Ice Thickness [m]	Ice coverage [%]	Floe Size (min) [-]	Floe Size (max) [-]	Shape (min sides) [-]	Shape (max sides) [-]	Floe Gap (min) [m]	Innitial x-location [m]	Innitial y-location [m]	Innitial Heading [deg]	Destination x-location [m]	Destination y-location [m]
1	1	2	140	170	4	7	0.1	1400	3200	130	1850	1750
2	1	2	140	170	4	7	0.1	3700	3000	20	850	2350
3	1	2	140	170	4	7	0.1	700	3400	150	2200	3500
4	1	2	140	170	4	7	0.1	1500	3750	80	3600	2500
5	1	2	140	170	4	7	0.1	800	600	20	600	1950
6	1	2	140	170	4	7	0.1	3000	2800	10	2800	3750
7	1	2	140	170	4	7	0.1	2300	1750	220	800	3100
8	1	2	140	170	4	7	0.1	700	3800	130	3150	1250
9	1	2	140	170	4	7	0.1	650	2300	270	2200	2600
10	1	2	140	170	4	7	0.1	300	3000	40	1350	1800
11	1	2	140	170	4	7	0.1	750	200	330	3350	2250
12	1	2	140	170	4	7	0.1	3700	3450	10	1600	550
13	1	2	140	170	4	7	0.1	2050	3300	230	2250	2300
14	1	2	140	170	4	7	0.1	3700	3450	310	2350	1200
15	1	2	140	170	4	7	0.1	2300	2450	120	850	350
16	1	2	140	170	4	7	0.1	700	250	310	2000	950
17	1	2	140	170	4	7	0.1	2250	2950	180	2800	3250
18	1	2	140	170	4	7	0.1	3150	3200	330	2400	2950
19	1	2	140	170	4	7	0.1	1650	2350	90	3650	2750
20	1	2	140	170	4	7	0.1	1450	2350	120	1050	1000
21	1	2	140	170	4	7	0.1	1300	2850	130	2250	400
22	1	2	140	170	4	7	0.1	350	750	180	650	2200
23	1	2	140	170	4	7	0.1	1200	200	40	800	2200
24	1	2	140	170	4	7	0.1	3400	2800	10	2450	2850
25	1	2	140	170	4	7	0.1	1700	3500	340	2700	3750
26	1	2	140	170	4	7	0.1	350	1150	320	3550	2250
27	1	2	140	170	4	7	0.1	2350	3650	300	1200	1550
28	1	2	140	170	4	7	0.1	300	3350	190	2150	1400
29	1	2	140	170	4	7	0.1	1800	650	220	850	2750
30	1	2	140	170	4	7	0.1	3100	3800	340	3750	1250
31	1	2	140	170	4	7	0.1	450	3700	90	2150	500
32	1	2	140	170	4	7	0.1	3350	2400	160	3550	1250
33	1	2	140	170	4	7	0.1	950	3000	90	3800	500
34	1	2	140	170	4	7	0.1	1400	1000	150	3650	1950
35	1	2	140	170	4	7	0.1	2300	1800	90	2500	1300
36	1	2	140	170	4	7	0.1	3400	1950	200	1250	1100
37	1	2	140	170	4	7	0.1	550	3700	150	2800	1700
38	1	2	140	170	4	7	0.1	400	3150	350	1300	2800
39	1	2	140	170	4	7	0.1	550	1900	150	1300	3750
40	1	2	140	170	4	7	0.1	1100	2000	150	300	1100
41	1	2	140	170	4	7	0.1	1350	3800	0	1950	3250
42	1	2	140	170	4	7	0.1	900	1450	40	300	1850
43	1	2	140	170	4	7	0.1	3650	1250	150	600	400
44	1	2	140	170	4	7	0.1	1100	300	270	3250	1200
45	1	2	140	170	4	7	0.1	2150	2300	130	1350	1850
46	1	2	140	170	4	7	0.1	1100	3450	30	2700	1000
47	1	2	140	170	4	7	0.1	1500	1850	250	1650	700
48	1	2	140	170	4	7	0.1	2000	2300	40	800	2200
49	1	2	140	170	4	7	0.1	3700	3400	50	1500	3300
50	1	2	140	170	4	7	0.1	2200	3550	260	600	3600

51	1	2	140	170	4	7	0.1	1350	1100	310	300	2050
52	1	2	140	170	4	7	0.1	1750	2000	260	400	2050
53	1	2	140	170	4	7	0.1	2350	3450	20	2450	2700
54	1	2	140	170	4	7	0.1	1700	950	80	650	1200
55	1	2	140	170	4	7	0.1	3450	1800	230	2800	3250
56	1	2	140	170	4	7	0.1	2750	1450	30	2200	1250
57	1	2	140	170	4	7	0.1	2450	3500	20	3650	3150
58	1	2	140	170	4	7	0.1	2400	550	220	2950	3300
59	1	2	140	170	4	7	0.1	1500	700	230	1900	1000
60	1	2	140	170	4	7	0.1	700	850	100	2050	3100
61	1	2	140	170	4	7	0.1	800	1100	160	1700	2850
62	1	2	140	170	4	7	0.1	500	2850	70	750	450
63	1	2	140	170	4	7	0.1	3200	2400	170	1100	2400
64	1	2	140	170	4	7	0.1	1600	3450	210	950	2300
65	1	2	140	170	4	7	0.1	1550	3200	240	3550	2550
66	1	2	140	170	4	7	0.1	1800	300	120	1700	2950
67	1	2	140	170	4	7	0.1	1350	2650	60	3500	1150
68	1	2	140	170	4	7	0.1	300	600	190	3700	950
69	1	2	140	170	4	7	0.1	450	2100	310	3650	2050
70	1	2	140	170	4	7	0.1	950	1400	230	600	3500
71	1	2	140	170	4	7	0.1	400	3200	80	3450	1700
72	1	2	140	170	4	7	0.1	3600	1150	10	1950	2750
73	1	2	140	170	4	7	0.1	550	1350	160	3600	2900
74	1	2	140	170	4	7	0.1	3200	350	110	400	3600
75	1	2	140	170	4	7	0.1	3350	3600	10	250	850
76	1	2	140	170	4	7	0.1	2200	350	20	2050	3100
77	1	2	140	170	4	7	0.1	2650	1650	170	3050	2050
78	1	2	140	170	4	7	0.1	2150	3500	180	1500	2500
79	1	2	140	170	4	7	0.1	600	2300	120	450	800
80	1	2	140	170	4	7	0.1	2850	1450	320	3200	2000
81	1	2	140	170	4	7	0.1	2350	2950	310	550	2500
82	1	2	140	170	4	7	0.1	2250	1750	350	1500	1650
83	1	2	140	170	4	7	0.1	1050	3500	340	1800	3250
84	1	2	140	170	4	7	0.1	2250	850	180	2600	3400
85	1	2	140	170	4	7	0.1	1150	3650	140	850	1800
86	1	2	140	170	4	7	0.1	3600	3800	240	400	650
87	1	2	140	170	4	7	0.1	2250	2400	100	900	2800
88	1	2	140	170	4	7	0.1	800	3350	140	2950	3400
89	1	2	140	170	4	7	0.1	1650	1950	180	2950	2400
90	1	2	140	170	4	7	0.1	250	2100	0	1450	400
91	1	2	140	170	4	7	0.1	3150	3500	170	3300	1600
92	1	2	140	170	4	7	0.1	2600	1850	200	1250	950
93	1	2	140	170	4	7	0.1	250	2500	340	450	1300
94	1	2	140	170	4	7	0.1	2750	3100	290	350	1600
95	1	2	140	170	4	7	0.1	1450	1750	10	2100	1500
96	1	2	140	170	4	7	0.1	2350	500	220	2150	2750
97	1	2	140	170	4	7	0.1	3600	1000	200	2600	3500
98	1	2	140	170	4	7	0.1	1250	3000	0	2200	700
99	1	2	140	170	4	7	0.1	3700	500	340	1050	1400
100	1	2	140	170	4	7	0.1	2500	2950	350	2550	1750

Appendix B – 100 Run Scenario Information (5% coverage)

Scene #	Ice Thickness	Ice coverage	Floe Size (min)	Floe Size (max)	Shape (min sides)	Shape (max sides)	Floe Gap (min)	Innitial x-location	Innitial y-location	Innitial Heading	Destination x-location	Destination y-location
	[m]	[%]	[-]	[-]	[-]	[-]	[m]	[m]	[m]	[deg]	[m]	[m]
1	1	2	140	170	4	7	0.1	3100	2800	20	1300	2000
2	1	2	140	170	4	7	0.1	3550	3350	0	1650	2550
3	1	2	140	170	4	7	0.1	2150	3500	310	1900	900
4	1	2	140	170	4	7	0.1	650	2950	210	3300	3700
5	1	2	140	170	4	7	0.1	3000	900	250	450	2550
6	1	2	140	170	4	7	0.1	1650	1450	160	1000	250
7	1	2	140	170	4	7	0.1	1400	850	290	3650	1650
8	1	2	140	170	4	7	0.1	3700	3450	0	650	250
9	1	2	140	170	4	7	0.1	1400	1150	250	2000	750
10	1	2	140	170	4	7	0.1	550	3200	200	2600	500
11	1	2	140	170	4	7	0.1	2500	3650	220	2150	2800
12	1	2	140	170	4	7	0.1	600	1150	340	3700	2900
13	1	2	140	170	4	7	0.1	700	850	240	2300	1400
14	1	2	140	170	4	7	0.1	2650	1850	180	3250	2000
15	1	2	140	170	4	7	0.1	3200	350	50	3550	3150
16	1	2	140	170	4	7	0.1	750	2150	220	2050	700
17	1	2	140	170	4	7	0.1	2050	1250	70	2350	2300
18	1	2	140	170	4	7	0.1	3450	2050	120	450	2850
19	1	2	140	170	4	7	0.1	1750	2550	260	2000	3300
20	1	2	140	170	4	7	0.1	1500	2300	0	1900	700
21	1	2	140	170	4	7	0.1	2650	300	130	3300	450
22	1	2	140	170	4	7	0.1	1500	2950	70	3750	1200
23	1	2	140	170	4	7	0.1	1250	3300	120	3100	650
24	1	2	140	170	4	7	0.1	3700	2400	70	1700	600
25	1	2	140	170	4	7	0.1	3800	1800	0	2150	2500
26	1	2	140	170	4	7	0.1	3700	2100	330	3350	550
27	1	2	140	170	4	7	0.1	750	300	170	2800	3050
28	1	2	140	170	4	7	0.1	500	1950	140	1950	1700
29	1	2	140	170	4	7	0.1	950	3250	90	2750	400
30	1	2	140	170	4	7	0.1	900	700	210	1300	2400
31	1	2	140	170	4	7	0.1	1050	1000	170	3450	1950
32	1	2	140	170	4	7	0.1	1300	1700	10	300	3300
33	1	2	140	170	4	7	0.1	2150	2700	340	850	2100
34	1	2	140	170	4	7	0.1	1750	3800	330	450	850
35	1	2	140	170	4	7	0.1	1400	3750	230	350	2700
36	1	2	140	170	4	7	0.1	2450	950	40	750	1100
37	1	2	140	170	4	7	0.1	3600	2450	250	2700	1350
38	1	2	140	170	4	7	0.1	2150	1300	260	600	3250
39	1	2	140	170	4	7	0.1	3150	1950	210	1300	3800
40	1	2	140	170	4	7	0.1	2050	2050	130	600	1450
41	1	2	140	170	4	7	0.1	1600	3800	40	3800	3350
42	1	2	140	170	4	7	0.1	1600	3050	280	1850	950
43	1	2	140	170	4	7	0.1	950	2050	190	2000	3450
44	1	2	140	170	4	7	0.1	800	1800	40	1300	3100
45	1	2	140	170	4	7	0.1	2300	1500	210	400	1600
46	1	2	140	170	4	7	0.1	250	3800	300	650	1550
47	1	2	140	170	4	7	0.1	1000	3200	340	1800	1700
48	1	2	140	170	4	7	0.1	2700	1550	170	2050	1150
49	1	2	140	170	4	7	0.1	3000	3100	70	3200	400
50	1	2	140	170	4	7	0.1	2300	2800	30	2900	3450

51	1	2	140	170	4	7	0.1	2000	2450	190	2900	2400
52	1	2	140	170	4	7	0.1	3750	350	80	3250	3500
53	1	2	140	170	4	7	0.1	800	2700	170	2100	1900
54	1	2	140	170	4	7	0.1	2450	1350	150	1000	1350
55	1	2	140	170	4	7	0.1	2800	200	40	1100	600
56	1	2	140	170	4	7	0.1	600	1350	140	2950	1250
57	1	2	140	170	4	7	0.1	2550	2800	250	2550	1850
58	1	2	140	170	4	7	0.1	3050	2750	100	3350	700
59	1	2	140	170	4	7	0.1	2100	2000	190	950	2200
60	1	2	140	170	4	7	0.1	1000	750	50	3750	2900
61	1	2	140	170	4	7	0.1	2000	900	60	3200	2750
62	1	2	140	170	4	7	0.1	200	2300	0	3550	3500
63	1	2	140	170	4	7	0.1	600	800	130	3200	1450
64	1	2	140	170	4	7	0.1	3600	3500	210	3800	750
65	1	2	140	170	4	7	0.1	3450	3550	70	300	300
66	1	2	140	170	4	7	0.1	1550	1500	270	3400	800
67	1	2	140	170	4	7	0.1	1900	2300	240	1950	1800
68	1	2	140	170	4	7	0.1	2500	650	10	1850	650
69	1	2	140	170	4	7	0.1	1200	2250	140	2500	3300
70	1	2	140	170	4	7	0.1	3800	600	330	750	1050
71	1	2	140	170	4	7	0.1	1700	300	210	250	900
72	1	2	140	170	4	7	0.1	600	1100	20	1800	3150
73	1	2	140	170	4	7	0.1	1100	3450	240	1300	2800
74	1	2	140	170	4	7	0.1	3000	600	140	650	2950
75	1	2	140	170	4	7	0.1	2450	2750	300	3800	350
76	1	2	140	170	4	7	0.1	2900	3600	320	200	900
77	1	2	140	170	4	7	0.1	1750	2550	170	3300	2250
78	1	2	140	170	4	7	0.1	2250	1650	250	2900	3200
79	1	2	140	170	4	7	0.1	2850	1050	200	350	1100
80	1	2	140	170	4	7	0.1	2800	900	140	1100	3250
81	1	2	140	170	4	7	0.1	950	3450	230	3400	2400
82	1	2	140	170	4	7	0.1	1800	3150	50	300	1700
83	1	2	140	170	4	7	0.1	750	750	150	3800	1450
84	1	2	140	170	4	7	0.1	1300	3000	90	3300	2300
85	1	2	140	170	4	7	0.1	3350	3600	110	700	3350
86	1	2	140	170	4	7	0.1	3750	2500	60	3050	3750
87	1	2	140	170	4	7	0.1	200	2400	110	3000	1250
88	1	2	140	170	4	7	0.1	3800	3400	40	2450	3650
89	1	2	140	170	4	7	0.1	1950	500	270	2450	3400
90	1	2	140	170	4	7	0.1	750	1550	60	2800	1200
91	1	2	140	170	4	7	0.1	2750	1000	40	3550	3350
92	1	2	140	170	4	7	0.1	1950	2700	220	1050	800
93	1	2	140	170	4	7	0.1	350	3000	60	1200	2500
94	1	2	140	170	4	7	0.1	1550	3250	300	2950	350
95	1	2	140	170	4	7	0.1	2700	550	0	3750	3450
96	1	2	140	170	4	7	0.1	450	1500	190	1400	2750
97	1	2	140	170	4	7	0.1	2900	2150	50	2200	250
98	1	2	140	170	4	7	0.1	2050	2600	100	700	2800
99	1	2	140	170	4	7	0.1	1250	400	80	2650	650
100	1	2	140	170	4	7	0.1	1900	1100	280	900	1100

Appendix C – 100 Run Scenario Information (10% coverage, large floes)

Scene #	Ice Thickness	Ice coverage	Floe Size (min)	Floe Size (max)	Shape (min sides)	Shape (max sides)	Floe Gap (min)	Innitial x-location	Innitial y-location	Innitial Heading	Destination x-location	Destination y-location
	[m]	[%]	[-]	[-]	[-]	[-]	[m]	[m]	[m]	[deg]	[m]	[m]
1	1	10	100	120	4	7	0.1	1350	2850	60	1400	300
2	1	10	100	120	4	7	0.1	600	850	80	2700	300
3	1	10	100	120	4	7	0.1	500	3050	320	2900	2100
4	1	10	100	120	4	7	0.1	1450	1350	130	3000	1300
5	1	10	100	120	4	7	0.1	3400	1550	120	3450	3350
6	1	10	100	120	4	7	0.1	3150	2800	90	1750	2900
7	1	10	100	120	4	7	0.1	1700	2200	120	400	1650
8	1	10	100	120	4	7	0.1	2800	3450	60	3350	2550
9	1	10	100	120	4	7	0.1	2600	700	60	1100	950
10	1	10	100	120	4	7	0.1	2450	1950	80	2000	1700
11	1	10	100	120	4	7	0.1	3000	900	180	3700	500
12	1	10	100	120	4	7	0.1	550	1200	100	650	2150
13	1	10	100	120	4	7	0.1	1500	300	310	2850	300
14	1	10	100	120	4	7	0.1	2000	1450	210	2500	1550
15	1	10	100	120	4	7	0.1	2600	3150	50	2500	650
16	1	10	100	120	4	7	0.1	2800	2400	130	2000	2550
17	1	10	100	120	4	7	0.1	3600	2800	320	3750	200
18	1	10	100	120	4	7	0.1	2100	1000	270	1050	3100
19	1	10	100	120	4	7	0.1	2100	3550	320	2800	1200
20	1	10	100	120	4	7	0.1	700	1800	180	950	3200
21	1	10	100	120	4	7	0.1	2350	2950	320	3550	3650
22	1	10	100	120	4	7	0.1	300	2100	280	1900	2200
23	1	10	100	120	4	7	0.1	1950	1750	40	900	1500
24	1	10	100	120	4	7	0.1	3150	2200	250	850	250
25	1	10	100	120	4	7	0.1	2500	2400	340	3300	3500
26	1	10	100	120	4	7	0.1	2050	3500	260	2150	1000
27	1	10	100	120	4	7	0.1	2350	800	100	1850	1100
28	1	10	100	120	4	7	0.1	2050	2100	220	1850	2900
29	1	10	100	120	4	7	0.1	3000	2100	110	1750	2700
30	1	10	100	120	4	7	0.1	3450	2550	40	1700	850
31	1	10	100	120	4	7	0.1	1150	1400	150	3800	2450
32	1	10	100	120	4	7	0.1	900	550	110	3000	3700
33	1	10	100	120	4	7	0.1	3200	1900	350	1100	2900
34	1	10	100	120	4	7	0.1	3750	2600	300	450	1600
35	1	10	100	120	4	7	0.1	2050	1750	140	2700	400
36	1	10	100	120	4	7	0.1	3500	550	100	2850	3750
37	1	10	100	120	4	7	0.1	1350	2450	50	3650	1550
38	1	10	100	120	4	7	0.1	2500	650	70	3500	1850
39	1	10	100	120	4	7	0.1	750	2250	210	2150	3750
40	1	10	100	120	4	7	0.1	400	2150	300	1500	2250
41	1	10	100	120	4	7	0.1	600	3300	30	1300	950
42	1	10	100	120	4	7	0.1	3650	1600	20	1800	800
43	1	10	100	120	4	7	0.1	1200	350	140	400	3450
44	1	10	100	120	4	7	0.1	3550	3150	330	1400	200
45	1	10	100	120	4	7	0.1	1100	550	210	2100	300
46	1	10	100	120	4	7	0.1	1950	2500	80	700	1300
47	1	10	100	120	4	7	0.1	700	1800	100	3350	500
48	1	10	100	120	4	7	0.1	3450	300	10	1550	2850
49	1	10	100	120	4	7	0.1	1150	2600	150	1500	1300
50	1	10	100	120	4	7	0.1	2900	1950	100	1100	2000

51	1	10	100	120	4	7	0.1	750	3100	40	3750	2950
52	1	10	100	120	4	7	0.1	350	2250	160	1550	2400
53	1	10	100	120	4	7	0.1	850	2750	0	850	1100
54	1	10	100	120	4	7	0.1	850	3000	80	2550	3350
55	1	10	100	120	4	7	0.1	1100	3200	190	1900	1200
56	1	10	100	120	4	7	0.1	250	1900	240	1950	2400
57	1	10	100	120	4	7	0.1	700	2900	290	3400	250
58	1	10	100	120	4	7	0.1	3450	1950	330	650	3250
59	1	10	100	120	4	7	0.1	2300	900	50	2550	3200
60	1	10	100	120	4	7	0.1	1850	3650	310	2900	350
61	1	10	100	120	4	7	0.1	1250	3500	120	2500	550
62	1	10	100	120	4	7	0.1	2800	1500	140	2950	3250
63	1	10	100	120	4	7	0.1	1550	2600	340	2600	3800
64	1	10	100	120	4	7	0.1	3100	200	300	450	1700
65	1	10	100	120	4	7	0.1	3450	2100	20	700	2950
66	1	10	100	120	4	7	0.1	3050	1150	290	500	3150
67	1	10	100	120	4	7	0.1	2400	1250	80	2000	650
68	1	10	100	120	4	7	0.1	650	3000	350	3750	1300
69	1	10	100	120	4	7	0.1	3000	1050	320	1450	500
70	1	10	100	120	4	7	0.1	850	650	190	2100	2550
71	1	10	100	120	4	7	0.1	3150	400	10	1350	1950
72	1	10	100	120	4	7	0.1	2800	2750	140	600	3500
73	1	10	100	120	4	7	0.1	1750	1400	120	1000	2100
74	1	10	100	120	4	7	0.1	2350	450	310	1950	3500
75	1	10	100	120	4	7	0.1	750	2500	320	2850	300
76	1	10	100	120	4	7	0.1	950	200	300	850	2650
77	1	10	100	120	4	7	0.1	1500	200	50	1100	1000
78	1	10	100	120	4	7	0.1	450	1900	170	2950	1250
79	1	10	100	120	4	7	0.1	3550	3550	190	1150	2450
80	1	10	100	120	4	7	0.1	3100	2100	80	450	350
81	1	10	100	120	4	7	0.1	1100	2100	190	1650	1100
82	1	10	100	120	4	7	0.1	2000	2900	10	1200	2800
83	1	10	100	120	4	7	0.1	1100	700	10	3450	1250
84	1	10	100	120	4	7	0.1	1950	1100	120	2850	2200
85	1	10	100	120	4	7	0.1	200	3000	130	3100	1250
86	1	10	100	120	4	7	0.1	3100	3200	50	1050	700
87	1	10	100	120	4	7	0.1	3750	3450	320	2050	1500
88	1	10	100	120	4	7	0.1	2400	3750	190	200	1050
89	1	10	100	120	4	7	0.1	300	1150	160	500	3100
90	1	10	100	120	4	7	0.1	500	3450	350	2400	3500
91	1	10	100	120	4	7	0.1	300	550	110	1550	2350
92	1	10	100	120	4	7	0.1	2050	1100	320	1200	1700
93	1	10	100	120	4	7	0.1	950	450	240	2550	2050
94	1	10	100	120	4	7	0.1	3350	950	260	3200	200
95	1	10	100	120	4	7	0.1	1650	1200	270	3000	650
96	1	10	100	120	4	7	0.1	1000	2300	190	1000	1000
97	1	10	100	120	4	7	0.1	1550	400	20	1950	3500
98	1	10	100	120	4	7	0.1	3050	3400	170	3750	3800
99	1	10	100	120	4	7	0.1	1200	2300	340	3150	2500
100	1	10	100	120	4	7	0.1	2600	1300	320	1450	3050

Appendix D – 100 Run Scenario Information (10% coverage, small floes)

Scene #	Ice Thickness	Ice coverage	Floe Size (min)	Floe Size (max)	Shape (min sides)	Shape (max sides)	Floe Gap (min)	Innitial x-location	Innitial y-location	Innitial Heading	Destination x-location	Destination y-location
	[m]	[%]	[-]	[-]	[-]	[-]	[m]	[m]	[m]	[deg]	[m]	[m]
1	1	10	40	60	4	7	0.1	3700	600	160	3700	3700
2	1	10	40	60	4	7	0.1	2100	900	290	2200	2900
3	1	10	40	60	4	7	0.1	1200	1700	80	3100	1000
4	1	10	40	60	4	7	0.1	3700	3100	180	1300	700
5	1	10	40	60	4	7	0.1	2200	1200	170	3550	3000
6	1	10	40	60	4	7	0.1	900	1000	10	1300	1650
7	1	10	40	60	4	7	0.1	1500	2550	210	2950	3000
8	1	10	40	60	4	7	0.1	800	2500	100	2900	800
9	1	10	40	60	4	7	0.1	1600	1350	340	2400	1500
10	1	10	40	60	4	7	0.1	2850	3650	260	2450	200
11	1	10	40	60	4	7	0.1	2950	3150	240	400	800
12	1	10	40	60	4	7	0.1	2150	3400	290	550	3150
13	1	10	40	60	4	7	0.1	3550	3700	30	3350	1550
14	1	10	40	60	4	7	0.1	2800	1800	290	3550	2050
15	1	10	40	60	4	7	0.1	1100	1550	200	3000	1100
16	1	10	40	60	4	7	0.1	3300	250	170	1850	2200
17	1	10	40	60	4	7	0.1	3550	1450	230	550	600
18	1	10	40	60	4	7	0.1	2400	3400	240	3000	500
19	1	10	40	60	4	7	0.1	950	3400	280	1600	700
20	1	10	40	60	4	7	0.1	3200	2250	130	1900	2650
21	1	10	40	60	4	7	0.1	450	300	200	2400	2200
22	1	10	40	60	4	7	0.1	1300	1100	100	2200	3200
23	1	10	40	60	4	7	0.1	3500	2550	0	2350	1800
24	1	10	40	60	4	7	0.1	1900	3250	340	2600	950
25	1	10	40	60	4	7	0.1	2250	2600	120	300	1850
26	1	10	40	60	4	7	0.1	1150	700	130	400	1550
27	1	10	40	60	4	7	0.1	2450	350	210	3750	1050
28	1	10	40	60	4	7	0.1	900	1900	350	1450	250
29	1	10	40	60	4	7	0.1	1450	3200	180	1600	400
30	1	10	40	60	4	7	0.1	3000	3000	110	300	1500
31	1	10	40	60	4	7	0.1	2900	3000	10	3800	2500
32	1	10	40	60	4	7	0.1	2950	2150	260	3650	550
33	1	10	40	60	4	7	0.1	3050	1350	220	2500	3650
34	1	10	40	60	4	7	0.1	3250	1200	50	1850	3050
35	1	10	40	60	4	7	0.1	1200	650	200	1900	900
36	1	10	40	60	4	7	0.1	500	2450	350	3450	2400
37	1	10	40	60	4	7	0.1	2850	1250	220	2600	1950
38	1	10	40	60	4	7	0.1	500	600	80	1550	3400
39	1	10	40	60	4	7	0.1	1650	2300	290	2550	350
40	1	10	40	60	4	7	0.1	1200	600	100	3350	3750
41	1	10	40	60	4	7	0.1	3550	350	280	750	2550
42	1	10	40	60	4	7	0.1	400	1800	110	1500	2300
43	1	10	40	60	4	7	0.1	800	1500	150	2550	2500
44	1	10	40	60	4	7	0.1	300	2000	90	2250	2400
45	1	10	40	60	4	7	0.1	3000	300	350	2900	1550
46	1	10	40	60	4	7	0.1	2200	3300	70	350	1550
47	1	10	40	60	4	7	0.1	2650	800	240	2100	750
48	1	10	40	60	4	7	0.1	1900	2050	240	3550	2050
49	1	10	40	60	4	7	0.1	2050	800	330	2750	3250
50	1	10	40	60	4	7	0.1	2050	1250	0	3200	3500

51	1	10	40	60	4	7	0.1	2400	2700	250	2500	400
52	1	10	40	60	4	7	0.1	1800	3350	30	3450	1900
53	1	10	40	60	4	7	0.1	2700	2500	130	1200	3600
54	1	10	40	60	4	7	0.1	3350	1300	60	1450	1300
55	1	10	40	60	4	7	0.1	2900	200	90	250	2550
56	1	10	40	60	4	7	0.1	2750	2500	120	1950	1000
57	1	10	40	60	4	7	0.1	650	1700	90	3150	900
58	1	10	40	60	4	7	0.1	1400	1750	250	400	2550
59	1	10	40	60	4	7	0.1	2600	1650	60	1800	1050
60	1	10	40	60	4	7	0.1	350	250	100	3000	2250
61	1	10	40	60	4	7	0.1	3600	1250	170	1500	3000
62	1	10	40	60	4	7	0.1	3100	2800	30	200	3650
63	1	10	40	60	4	7	0.1	1850	2550	40	3200	250
64	1	10	40	60	4	7	0.1	3100	3550	40	2300	1950
65	1	10	40	60	4	7	0.1	1550	2500	340	3700	300
66	1	10	40	60	4	7	0.1	2400	3000	110	3800	1500
67	1	10	40	60	4	7	0.1	2150	1200	170	700	3300
68	1	10	40	60	4	7	0.1	2350	2500	110	2200	3300
69	1	10	40	60	4	7	0.1	1900	1200	160	3050	700
70	1	10	40	60	4	7	0.1	1600	1600	110	3200	2350
71	1	10	40	60	4	7	0.1	3200	3700	60	1800	3050
72	1	10	40	60	4	7	0.1	600	1000	100	2300	2000
73	1	10	40	60	4	7	0.1	3350	1100	120	1000	1300
74	1	10	40	60	4	7	0.1	1100	3550	20	2200	3650
75	1	10	40	60	4	7	0.1	3300	3050	330	2100	1500
76	1	10	40	60	4	7	0.1	1100	3400	180	2000	2700
77	1	10	40	60	4	7	0.1	400	3750	80	2800	3150
78	1	10	40	60	4	7	0.1	500	350	350	3150	600
79	1	10	40	60	4	7	0.1	3600	2600	240	1300	3650
80	1	10	40	60	4	7	0.1	1750	1700	290	950	1050
81	1	10	40	60	4	7	0.1	1450	1750	20	3250	1450
82	1	10	40	60	4	7	0.1	3050	1250	290	750	450
83	1	10	40	60	4	7	0.1	1650	2850	140	2800	2950
84	1	10	40	60	4	7	0.1	2600	1000	310	1450	3550
85	1	10	40	60	4	7	0.1	3600	2750	280	250	1550
86	1	10	40	60	4	7	0.1	600	2100	30	2900	1550
87	1	10	40	60	4	7	0.1	550	1200	220	2000	1850
88	1	10	40	60	4	7	0.1	300	3650	80	2600	1550
89	1	10	40	60	4	7	0.1	2600	1600	130	200	200
90	1	10	40	60	4	7	0.1	350	2250	140	2550	1250
91	1	10	40	60	4	7	0.1	3550	900	10	2650	1200
92	1	10	40	60	4	7	0.1	400	1000	200	3650	3450
93	1	10	40	60	4	7	0.1	850	1750	140	3200	2150
94	1	10	40	60	4	7	0.1	750	2550	350	3450	3100
95	1	10	40	60	4	7	0.1	2600	2000	230	2150	1800
96	1	10	40	60	4	7	0.1	2850	3750	220	1250	3150
97	1	10	40	60	4	7	0.1	3700	3300	300	2000	3300
98	1	10	40	60	4	7	0.1	1000	1850	30	3350	1450
99	1	10	40	60	4	7	0.1	1300	2500	190	1200	1600
100	1	10	40	60	4	7	0.1	3100	1800	300	2500	800

Appendix E – 100 Run Ship Transit Data (2% coverage)

Scene #	Trial Distance	Destination Reached	Total Step Count	Collision Count	Min Speed	Max Speed	Average Speed
1	1518	Yes	310	0	0	2.898	2.670
2	2923	Yes	570	1	0	2.896	2.745
3	1503	Yes	301	0	0	2.896	2.693
4	2444	Yes	452	0	0	2.895	2.762
5	1365	Yes	249	0	0	2.893	2.689
6	971	Yes	184	0	0	2.893	2.625
7	2018	Yes	363	0	0	2.895	2.766
8	3536	Yes	679	0	0	2.896	2.779
9	1579	Yes	299	0	0	2.897	2.712
10	1595	Yes	293	0	0	2.892	2.722
11	3311	Yes	611	0	0	2.895	2.718
12	3581	Yes	660	0	0	2.898	2.793
13	1020	Yes	180	0	0	2.885	2.665
14	2624	Yes	470	7	0	2.897	2.785
15	2552	Yes	453	0	0	2.891	2.786
16	1476	Yes	268	0	0	2.891	2.701
17	626	Yes	155	0	0	2.880	2.486
18	791	Yes	175	0	0	2.892	2.576
19	2040	Yes	367	0	0	2.896	2.761
20	1408	Yes	279	0	0	2.893	2.667
21	2628	Yes	508	0	0	2.896	2.741
22	1481	Yes	280	0	0	2.894	2.682
23	2040	Yes	367	3	0	2.897	2.734
24	951	Yes	224	8	0	2.883	2.533
25	1031	Yes	187	0	0	2.892	2.565
26	3384	Yes	598	0	0	2.896	2.809
27	2394	Yes	421	0	0	2.895	2.789
28	2688	Yes	501	0	0	2.896	2.760
29	2305	Yes	422	0	0	2.896	2.765
30	2632	Yes	462	0	0	2.897	2.799
31	3624	Yes	697	18	0	2.897	2.766
32	1167	Yes	234	0	0	2.897	2.648
33	3791	Yes	698	0	0	2.895	2.802
34	2442	Yes	459	0	0	2.895	2.765
35	539	Yes	142	0	0	2.886	2.445
36	2312	Yes	412	1	0	2.894	2.752
37	3010	Yes	578	0	0	2.899	2.755
38	966	Yes	162	0	0	2.895	2.705
39	1996	Yes	360	0	0	2.896	2.760
40	1204	Yes	220	0	0	2.894	2.673
41	814	Yes	163	10	0	2.890	2.462
42	721	Yes	148	0	0	2.894	2.515
43	3166	Yes	559	0	0	2.897	2.801
44	2331	Yes	433	0	0	2.898	2.753
45	918	Yes	169	0	0	2.896	2.621

46	2926	Yes	525	0	0	2.895	2.799
47	1160	Yes	198	0	0	2.895	2.724
48	1204	Yes	259	1	0	2.896	2.569
49	2202	Yes	421	0	0	2.895	2.732
50	1601	Yes	292	1	0	2.898	2.697
51	1416	Yes	303	0	0	2.892	2.665
52	1351	Yes	245	0	0	2.894	2.709
53	757	Yes	152	0	0	2.897	2.527
54	1079	Yes	201	0	0	2.897	2.653
55	1589	Yes	300	0	0	2.896	2.712
56	585	Yes	156	0	0	2.892	2.445
57	1250	Yes	218	0	0	2.891	2.726
58	2804	Yes	533	4	0	2.896	2.766
59	500	Yes	153	1	0	2.883	2.275
60	2624	Yes	459	13	0	2.897	2.797
61	1968	Yes	359	0	0	2.896	2.753
62	2413	Yes	506	1	0	2.897	2.551
63	2100	Yes	362	0	0	2.895	2.788
64	1321	Yes	244	0	0	2.895	2.634
65	2103	Yes	390	0	0	2.894	2.748
66	2652	Yes	471	0	0	2.894	2.769
67	2622	Yes	479	0	0	2.896	2.761
68	3418	Yes	658	7	0	2.896	2.758
69	3200	Yes	567	0	0	2.896	2.805
70	2129	Yes	407	0	0	2.894	2.726
71	3399	Yes	617	0	0	2.895	2.797
72	2298	Yes	435	0	0	2.895	2.751
73	3421	Yes	666	2	0	2.896	2.684
74	4290	Yes	783	0	0	2.896	2.777
75	4144	Yes	774	6	0	2.896	2.790
76	2754	Yes	489	0	0	2.895	2.786
77	566	Yes	137	0	0	2.879	2.462
78	1193	Yes	211	0	0	2.892	2.696
79	1507	Yes	299	1	0	2.894	2.658
80	652	Yes	133	0	0	2.883	2.500
81	1855	Yes	347	0	0	2.892	2.733
82	757	Yes	187	0	0	2.892	2.534
83	791	Yes	144	0	0	2.892	2.470
84	2574	Yes	489	0	0	2.893	2.704
85	1874	Yes	354	0	0	2.895	2.726
86	4490	Yes	805	7	0	2.897	2.805
87	1408	Yes	255	3	0	2.894	2.680
88	2151	Yes	420	4	0	2.897	2.712
89	1376	Yes	289	0	0	2.893	2.668
90	2081	Yes	365	0	0	2.895	2.781
91	1906	Yes	355	0	0	2.889	2.729
92	1622	Yes	277	0	0	2.898	2.774
93	1217	Yes	219	0	0	2.897	2.690
94	2830	Yes	504	0	0	2.895	2.791
95	696	Yes	117	0	0	2.890	2.617
96	2259	Yes	424	0	0	2.894	2.753
97	2693	Yes	498	1	0	2.895	2.721
98	2488	Yes	459	1	0	2.897	2.719
99	2799	Yes	554	8	0	2.896	2.759
100	1201	Yes	220	0	0	2.895	2.703

Appendix F – 100 Run Ship Transit Data (5% coverage)

Scene #	Trial Distance	Destination Reached	Total Step Count	Collision Count	Min Speed	Max Speed	Average Speed
1	1970	Yes	376	11	0	2.896	2.738
2	2062	Yes	438	1	0	2.896	2.551
3	2612	Yes	464	2	0	2.897	2.770
4	2754	Yes	580	5	0	2.897	2.544
5	3037	Yes	620	47	0	2.892	2.530
6	1365	Yes	267	1	0	2.893	2.548
7	2388	Yes	461	0	0	2.897	2.618
8	4421	Yes	861	11	0	2.895	2.685
9	721	Yes	135	0	0	2.893	2.563
10	3390	Yes	649	0	0	2.894	2.658
11	919	Yes	162	1	0	2.886	2.640
12	3560	Yes	641	3	0	2.898	2.779
13	1692	Yes	349	7	0	2.897	2.699
14	618	Yes	174	1	0	2.894	2.311
15	2822	Yes	511	1	0	2.896	2.625
16	1947	Yes	372	1	0	2.895	2.640
17	1092	Yes	198	2	0	2.892	2.616
18	3105	Yes	546	0	0	2.897	2.804
19	791	Yes	194	0	0	2.886	2.526
20	1649	Yes	318	3	0	2.895	2.630
21	667	Yes	144	0	0	2.896	2.502
22	2850	Yes	521	1	0	2.898	2.689
23	3232	Yes	662	9	0	2.897	2.678
24	2691	Yes	475	0	0	2.896	2.774
25	1792	Yes	377	1	0	2.895	2.604
26	1589	Yes	297	17	0	2.894	2.640
27	3430	Yes	662	11	0	2.896	2.688
28	1471	Yes	299	1	0	2.891	2.625
29	3371	Yes	679	3	0	2.897	2.654
30	1746	Yes	342	1	0	2.896	2.698
31	2581	Yes	497	0	0	2.891	2.750
32	1887	Yes	364	5	0	2.896	2.682
33	1432	Yes	308	2	0	2.894	2.528
34	3224	Yes	585	2	0	2.893	2.780
35	1485	Yes	320	0	0	2.893	2.482
36	1707	Yes	336	0	0	2.896	2.708
37	1421	Yes	242	0	0	2.897	2.763
38	2491	Yes	539	31	0	2.895	2.431
39	2616	Yes	487	22	0	2.897	2.701
40	1569	Yes	282	0	0	2.891	2.721
41	2246	Yes	393	0	0	2.896	2.792
42	2115	Yes	378	1	0	2.894	2.724
43	1750	Yes	359	8	0	2.890	2.641
44	1393	Yes	241	1	0	2.897	2.728
45	1903	Yes	337	0	0	2.895	2.732

46	2285	Yes	409	0	0	2.898	2.746
47	1700	Yes	298	0	0	2.891	2.758
48	763	Yes	134	0	0	2.885	2.605
49	2707	Yes	521	0	0	2.898	2.761
50	885	Yes	182	14	0	2.894	2.591
51	901	Yes	208	0	0	2.895	2.552
52	3189	Yes	579	1	0	2.894	2.754
53	1526	Yes	338	7	0	2.886	2.508
54	1450	Yes	251	0	0	2.894	2.745
55	1746	Yes	336	0	0	2.894	2.713
56	2352	Yes	464	21	0	2.896	2.690
57	950	Yes	161	0	0	2.891	2.692
58	2072	Yes	426	2	0	2.895	2.638
59	1167	Yes	200	0	0	2.894	2.729
60	3491	Yes	613	3	0	2.896	2.812
61	2205	Yes	399	5	0	2.892	2.721
62	3558	Yes	619	1	0	2.898	2.826
63	2680	Yes	494	1	0	2.897	2.750
64	2757	Yes	503	1	0	2.896	2.719
65	4526	Yes	904	15	0	2.897	2.656
66	1978	Yes	382	1	0	2.894	2.620
67	502	Yes	98	1	0	2.887	2.216
68	650	Yes	175	0	0	2.883	2.512
69	1671	Yes	311	1	0	2.896	2.708
70	3083	Yes	636	14	0	2.897	2.634
71	1569	Yes	306	1	0	2.893	2.532
72	2375	Yes	407	0	0	2.895	2.785
73	680	Yes	126	1	0	2.886	2.521
74	3323	Yes	593	1	0	2.898	2.751
75	2754	Yes	477	0	0	2.895	2.811
76	3818	Yes	705	15	0	2.897	2.767
77	1579	Yes	353	7	0	2.895	2.603
78	1681	Yes	349	0	0	2.895	2.671
79	2500	Yes	441	2	0	2.897	2.772
80	2900	Yes	508	0	0	2.896	2.802
81	2666	Yes	454	0	0	2.897	2.764
82	2086	Yes	432	2	0	2.895	2.630
83	3129	Yes	593	0	0	2.896	2.744
84	2119	Yes	423	0	0	2.896	2.559
85	2662	Yes	480	8	0	2.897	2.754
86	1433	Yes	245	1	0	2.897	2.726
87	3027	Yes	575	7	0	2.897	2.747
88	1373	Yes	273	0	0	2.895	2.666
89	2943	Yes	600	5	0	2.897	2.677
90	2080	Yes	377	1	0	2.897	2.715
91	2482	Yes	434	0	0	2.896	2.798
92	2102	Yes	377	0	0	2.894	2.708
93	986	Yes	190	1	0	2.890	2.592
94	3220	Yes	601	1	0	2.894	2.744
95	3084	Yes	594	17	0	2.894	2.663
96	1570	Yes	309	0	0	2.894	2.696
97	2025	Yes	400	0	0	2.896	2.728
98	1365	Yes	256	9	0	2.894	2.646
99	1422	Yes	252	0	0	2.896	2.736
100	1000	Yes	158	0	0	2.886	2.569

Appendix G – 100 Run Ship Transit Data (10% coverage, large floes)

Scene #	Trial Distance	Destination Reached	Total Step Count	Collision Count	Min Speed	Max Speed	Average Speed
1	2550	Yes	533	4	0	2.896	2.552
2	2171	Yes	444	68	0	2.898	2.552
3	2581	Yes	497	22	0	2.892	2.599
4	1551	Yes	313	18	0	2.897	2.638
5	1801	Yes	327	2	0	2.893	2.675
6	1404	Yes	287	63	0	2.890	2.503
7	1412	Yes	271	2	0	2.887	2.606
8	1055	Yes	244	24	0	2.891	2.322
9	1521	Yes	295	2	0	2.895	2.651
10	515	Yes	128	2	0	2.873	2.351
11	806	Yes	202	26	0	2.894	2.352
12	955	Yes	163	0	0	2.893	2.689
13	1350	Yes	247	15	0	2.893	2.660
14	510	Yes	155	1	0	2.878	2.211
15	2502	Yes	507	20	0	2.897	2.607
16	814	Yes	144	0	0	2.892	2.602
17	2604	Yes	518	58	0	2.895	2.551
18	2348	Yes	476	8	0	2.896	2.633
19	2452	Yes	447	3	0	2.897	2.704
20	1422	Yes	293	15	0	2.894	2.457
21	1389	Yes	323	2	0	2.882	2.239
22	1603	Yes	319	1	0	2.894	2.560
23	1079	Yes	249	0	0	2.896	2.415
24	3015	Yes	535	2	0	2.895	2.799
25	1360	Yes	256	1	0	2.897	2.608
26	2502	Yes	440	1	0	2.897	2.769
27	583	Yes	105	3	0	2.866	2.516
28	825	Yes	179	1	0	2.895	2.467
29	1387	Yes	246	1	0	2.895	2.690
30	2440	Yes	485	7	0	2.896	2.706
31	2850	Yes	628	6	0	2.894	2.413
32	3786	Yes	726	23	0	2.896	2.688
33	2326	Yes	498	3	0	2.896	2.538
34	3448	Yes	661	21	0	2.897	2.696
35	1498	Yes	315	1	0	2.896	2.620
36	3265	Yes	625	30	0	2.895	2.615
37	2470	Yes	447	6	0	2.894	2.753
38	1562	Yes	289	0	0	2.888	2.653
39	2052	Yes	423	6	0	2.894	2.682
40	1105	Yes	235	29	0	2.888	2.316
41	2452	Yes	479	29	0	2.893	2.647
42	2016	Yes	431	6	0	2.895	2.609
43	3202	Yes	614	4	0	2.895	2.638
44	3650	Yes	690	4	0	2.893	2.688
45	1031	Yes	220	1	0	2.889	2.588

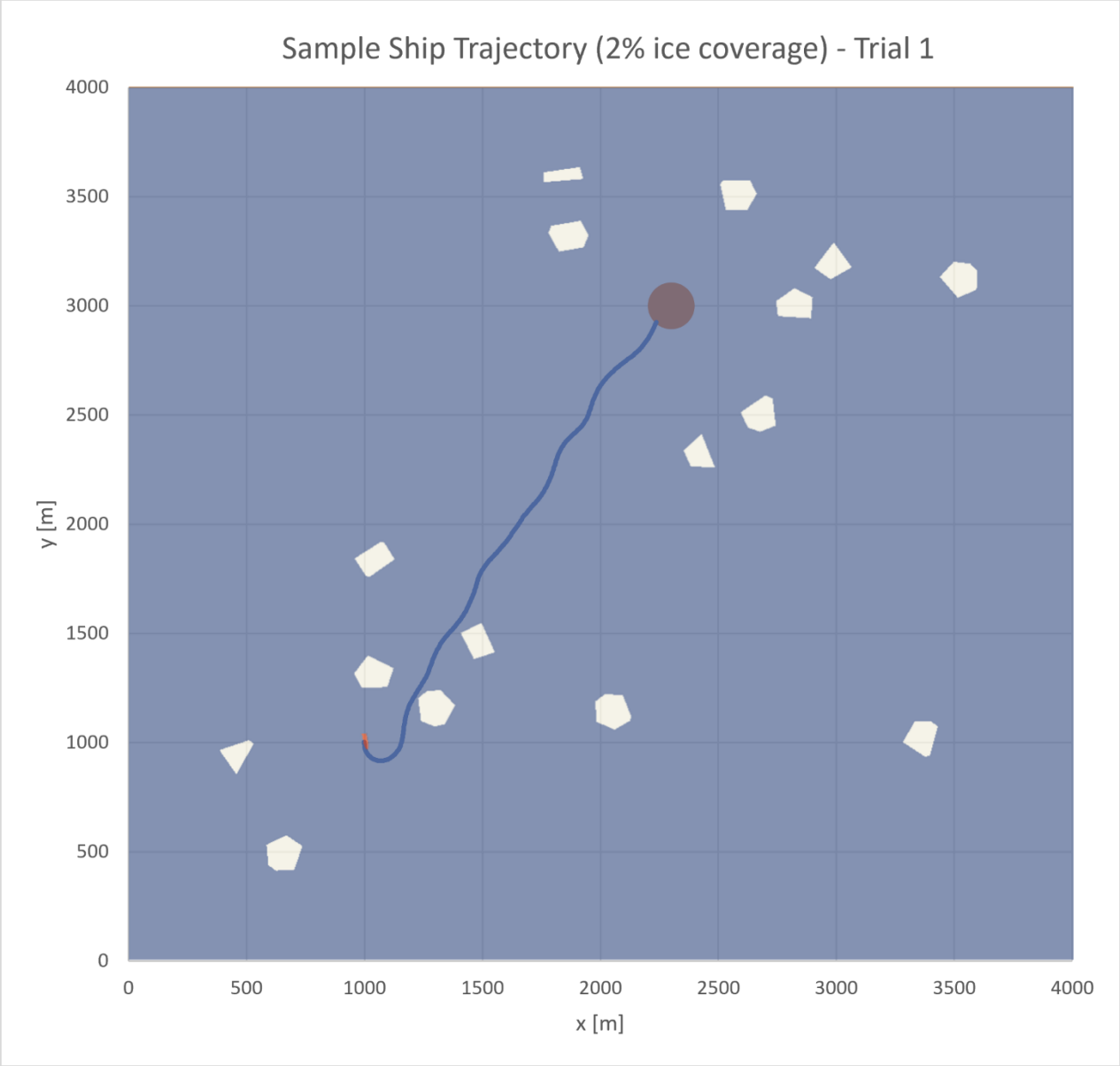
46	1733	Yes	348	13	0	2.894	2.661
47	2952	Yes	642	6	0	2.894	2.445
48	3180	Yes	623	14	0	2.897	2.652
49	1346	Yes	270	26	0	2.893	2.608
50	1801	Yes	331	3	0	2.894	2.692
51	3004	Yes	556	9	0	2.893	2.715
52	1209	Yes	278	15	0	2.896	2.438
53	1650	Yes	322	11	0	2.896	2.576
54	1736	Yes	327	1	0	2.890	2.680
55	2154	Yes	400	1	0	2.895	2.729
56	1772	Yes	357	5	0	2.892	2.651
57	3783	Yes	732	19	0	2.896	2.622
58	3087	Yes	627	3	0	2.895	2.645
59	2314	Yes	444	12	0	2.896	2.626
60	3463	Yes	653	18	0	2.895	2.674
61	3204	Yes	667	3	0	2.893	2.586
62	1756	Yes	325	1	0	2.897	2.665
63	1595	Yes	295	1	0	2.891	2.688
64	3045	Yes	611	28	0	2.895	2.659
65	2878	Yes	571	18	0	2.897	2.667
66	3241	Yes	685	24	0	2.896	2.556
67	721	Yes	214	46	0	2.885	2.198
68	3536	Yes	696	41	0	2.896	2.526
69	1645	Yes	354	19	0	2.893	2.468
70	2274	Yes	437	5	0	2.896	2.710
71	2375	Yes	463	10	0	2.894	2.685
72	2324	Yes	422	4	0	2.892	2.719
73	1026	Yes	182	2	0	2.888	2.632
74	3076	Yes	641	3	0	2.897	2.545
75	3041	Yes	568	1	0	2.895	2.661
76	2452	Yes	527	50	0	2.894	2.614
77	894	Yes	161	9	0	2.887	2.614
78	2583	Yes	560	16	0	2.896	2.508
79	2640	Yes	487	4	0	2.894	2.701
80	3176	Yes	616	4	0	2.893	2.674
81	1141	Yes	229	2	0	2.893	2.570
82	806	Yes	200	0	0	2.894	2.516
83	2414	Yes	440	12	0	2.893	2.690
84	1421	Yes	262	12	0	2.895	2.676
85	3387	Yes	674	16	0	2.896	2.688
86	3233	Yes	662	14	0	2.896	2.664
87	2587	Yes	503	2	0	2.895	2.592
88	3483	Yes	717	69	0	2.893	2.466
89	1960	Yes	365	8	0	2.895	2.703
90	1901	Yes	355	3	0	2.893	2.642
91	2191	Yes	398	15	0	2.891	2.720
92	1040	Yes	244	4	0	2.893	2.562
93	2263	Yes	493	2	0	2.889	2.512
94	765	Yes	127	7	0	2.895	2.648
95	1458	Yes	292	16	0	2.891	2.425
96	1300	Yes	243	1	0	2.892	2.656
97	3126	Yes	581	3	0	2.894	2.706
98	806	Yes	184	0	0	2.889	2.546
99	1960	Yes	348	3	0	2.892	2.747
100	2094	Yes	447	1	0	2.896	2.610

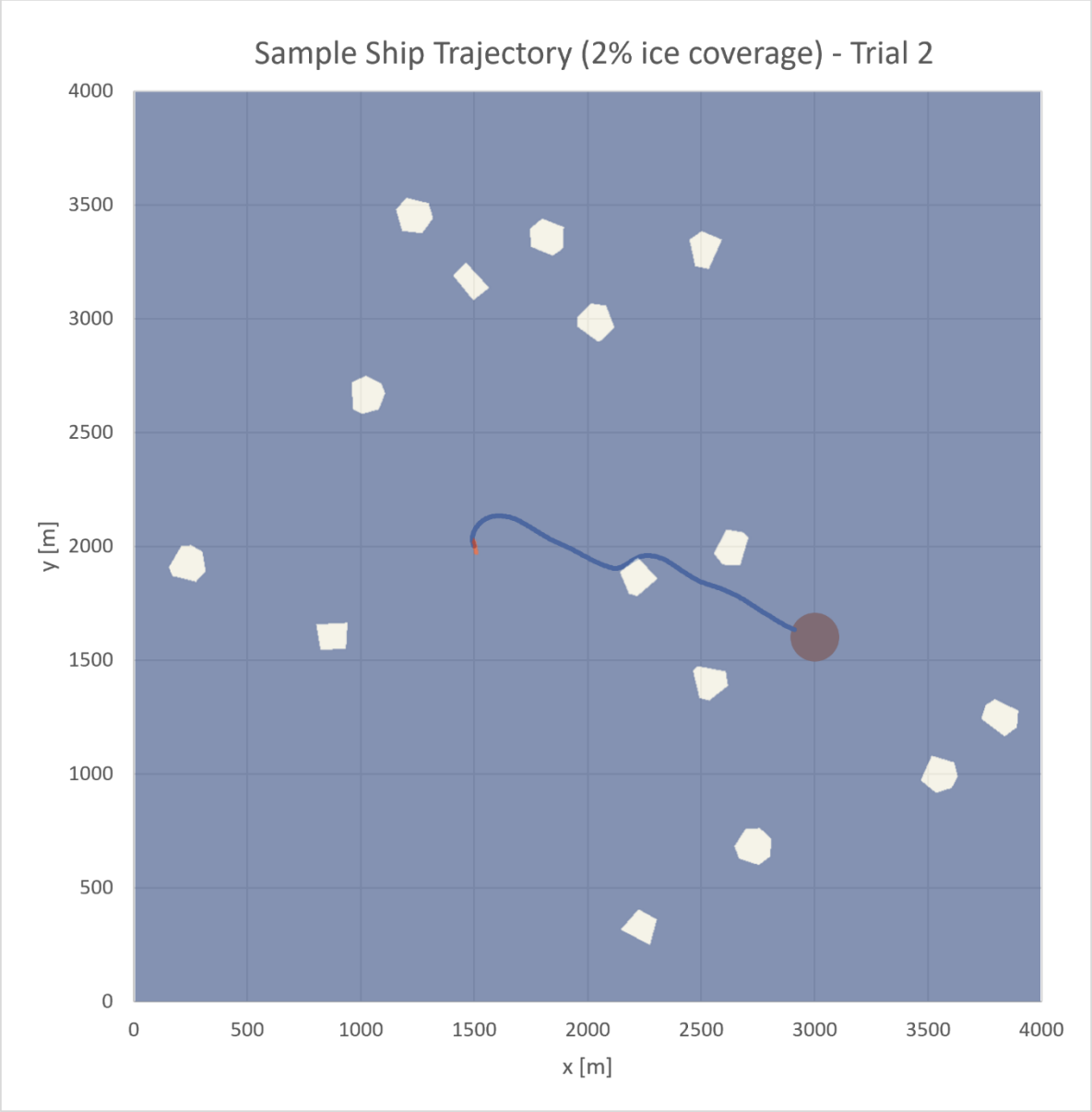
Appendix H – 100 Run Ship Transit Data (10% coverage, small floes)

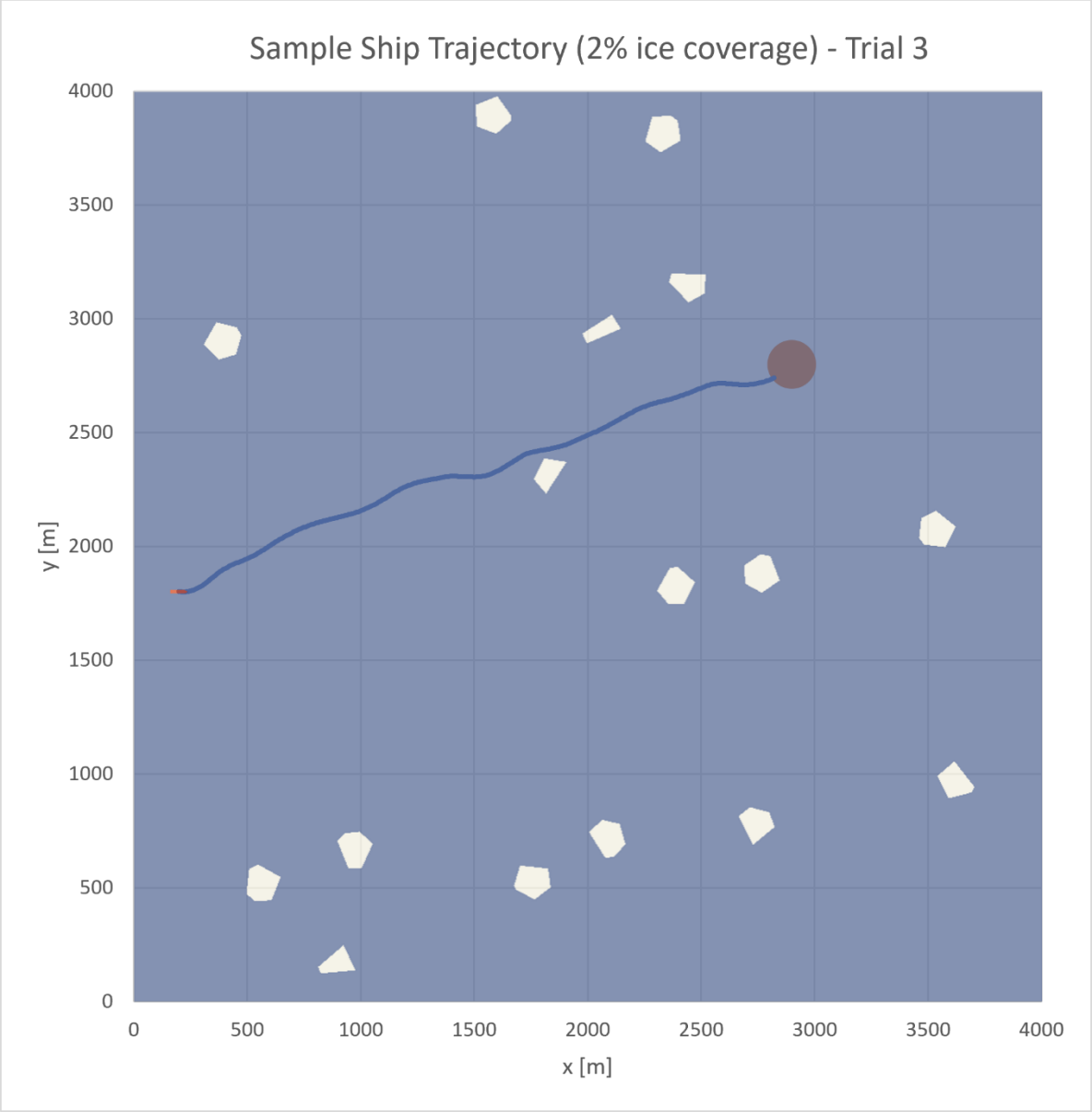
Scene #	Trial Distance	Destination Reached	Total Step Count	Collision Count	Min Speed	Max Speed	Average Speed
1	3100	Yes	571	17	0	2.894	2.713
2	2002	Yes	401	2	0	2.894	2.688
3	2025	Yes	402	17	0	2.889	2.570
4	3394	Yes	647	11	0	2.897	2.626
5	2250	Yes	440	8	0	2.894	2.625
6	763	Yes	152	3	0	2.885	2.319
7	1518	Yes	327	6	0	2.897	2.577
8	2702	Yes	531	14	0	2.895	2.681
9	814	Yes	145	2	0	2.895	2.532
10	3473	Yes	662	9	0	2.893	2.612
11	3468	Yes	639	11	0	2.893	2.702
12	1619	Yes	315	10	0	2.892	2.633
13	2159	Yes	425	7	0	2.895	2.649
14	791	Yes	167	1	0	2.887	2.325
15	1953	Yes	397	15	0	2.892	2.638
16	2430	Yes	451	13	0	2.894	2.661
17	3118	Yes	575	5	0	2.896	2.693
18	2961	Yes	563	3	0	2.899	2.627
19	2777	Yes	491	3	0	2.896	2.747
20	1360	Yes	241	3	0	2.889	2.683
21	2723	Yes	570	18	0	2.894	2.586
22	2285	Yes	412	4	0	2.891	2.713
23	1373	Yes	281	1	0	2.895	2.654
24	2404	Yes	435	8	0	2.896	2.717
25	2089	Yes	388	2	0	2.897	2.681
26	1134	Yes	206	2	0	2.897	2.568
27	1476	Yes	333	7	0	2.895	2.580
28	1739	Yes	329	5	0	2.891	2.587
29	2804	Yes	540	4	0	2.895	2.629
30	3089	Yes	608	15	0	2.894	2.580
31	1030	Yes	189	2	0	2.891	2.532
32	1746	Yes	320	9	0	2.896	2.627
33	2365	Yes	487	24	0	2.891	2.523
34	2320	Yes	439	15	0	2.887	2.653
35	743	Yes	200	3	0	2.881	2.343
36	2950	Yes	531	10	0	2.896	2.720
37	743	Yes	156	2	0	2.893	2.495
38	2990	Yes	560	8	0	2.895	2.649
39	2148	Yes	405	12	0	2.895	2.626
40	3814	Yes	684	3	0	2.896	2.775
41	3561	Yes	699	12	0	2.896	2.683
42	1208	Yes	227	1	0	2.890	2.631
43	2016	Yes	392	1	0	2.893	2.673
44	1991	Yes	377	11	0	2.892	2.625
45	1254	Yes	255	19	0	2.886	2.557

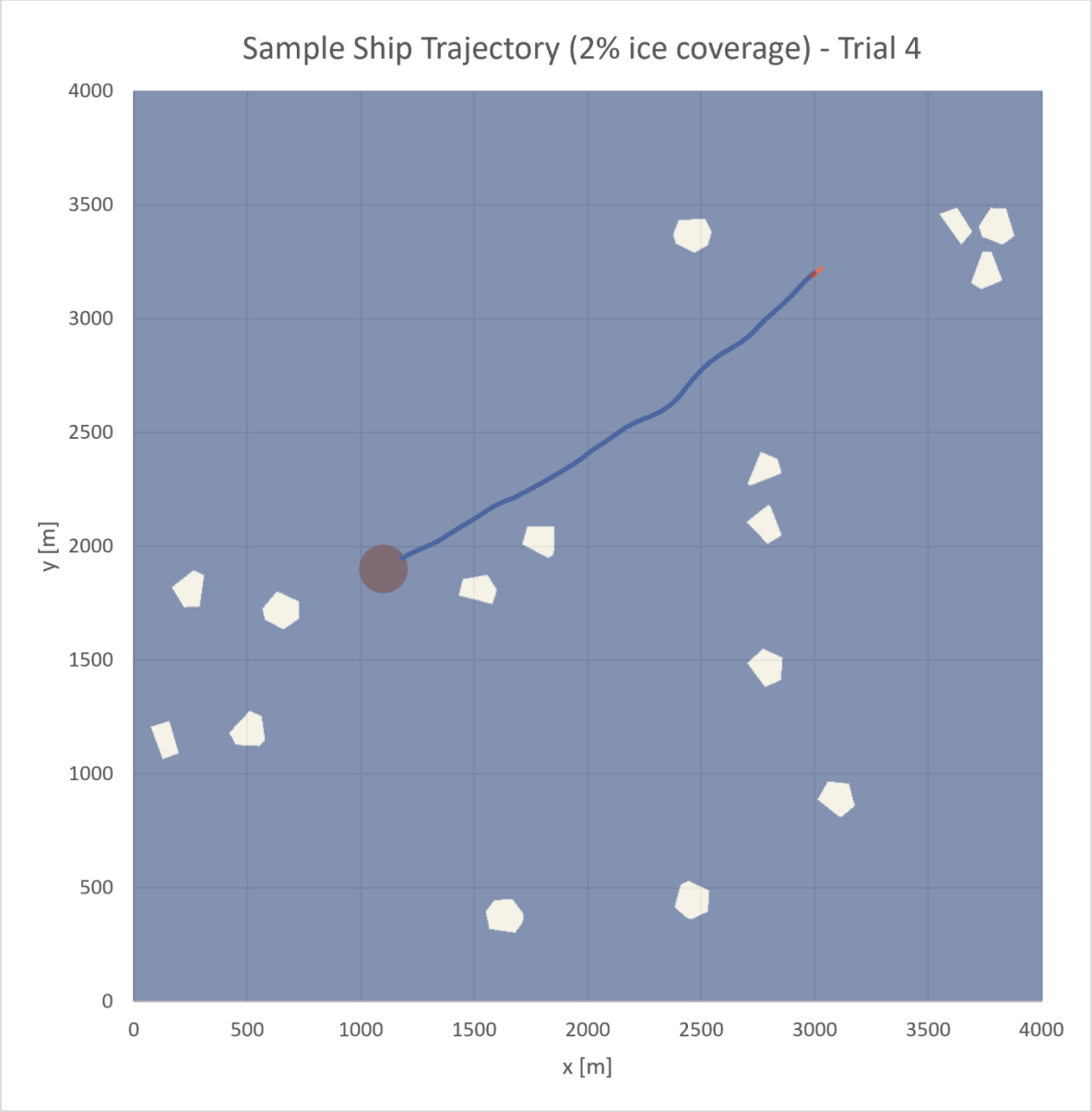
46	2547	Yes	500	4	0	2.896	2.708
47	552	Yes	104	2	0	2.895	2.376
48	1650	Yes	338	5	0	2.893	2.566
49	2548	Yes	508	7	0	2.897	2.560
50	2527	Yes	468	5	0	2.896	2.681
51	2302	Yes	426	11	0	2.893	2.640
52	2197	Yes	426	6	0	2.895	2.563
53	1860	Yes	322	0	0	2.895	2.768
54	1900	Yes	374	6	0	2.895	2.661
55	3542	Yes	666	19	0	2.897	2.655
56	1700	Yes	350	4	0	2.892	2.555
57	2625	Yes	506	9	0	2.893	2.636
58	1281	Yes	265	6	0	2.889	2.538
59	1000	Yes	221	1	0	2.893	2.576
60	3320	Yes	611	10	0	2.897	2.724
61	2734	Yes	514	10	0	2.894	2.644
62	3022	Yes	584	6	0	2.895	2.673
63	2667	Yes	518	14	0	2.896	2.600
64	1789	Yes	375	8	0	2.898	2.613
65	3076	Yes	557	4	0	2.896	2.728
66	2052	Yes	430	15	0	2.894	2.597
67	2552	Yes	469	7	0	2.893	2.682
68	814	Yes	146	12	0	2.893	2.516
69	1254	Yes	279	2	0	2.894	2.578
70	1767	Yes	325	4	0	2.896	2.695
71	1544	Yes	325	4	0	2.897	2.568
72	1972	Yes	365	3	0	2.888	2.693
73	2358	Yes	434	2	0	2.897	2.684
74	1105	Yes	186	1	0	2.895	2.729
75	1960	Yes	376	3	0	2.895	2.681
76	1140	Yes	257	4	0	2.894	2.461
77	2474	Yes	460	10	0	2.896	2.622
78	2662	Yes	504	11	0	2.896	2.714
79	2528	Yes	476	7	0	2.892	2.672
80	1031	Yes	201	1	0	2.885	2.460
81	1825	Yes	332	8	0	2.893	2.644
82	2435	Yes	467	7	0	2.893	2.628
83	1154	Yes	241	3	0	2.892	2.578
84	2797	Yes	562	2	0	2.894	2.646
85	3558	Yes	685	21	0	2.894	2.614
86	2365	Yes	474	14	0	2.894	2.470
87	1589	Yes	338	18	0	2.893	2.574
88	3114	Yes	594	9	0	2.897	2.726
89	2778	Yes	529	17	0	2.895	2.659
90	2417	Yes	522	8	0	2.893	2.509
91	949	Yes	227	3	0	2.873	2.415
92	4070	Yes	802	16	0	2.895	2.705
93	2384	Yes	462	12	0	2.894	2.704
94	2755	Yes	494	4	0	2.891	2.735
95	492	Yes	89	3	0	2.880	2.334
96	1709	Yes	303	2	0	2.895	2.711
97	1700	Yes	332	12	0	2.895	2.669
98	2384	Yes	426	5	0	2.893	2.736
99	906	Yes	178	4	0	2.893	2.458
100	1166	Yes	223	2	0	2.896	2.495

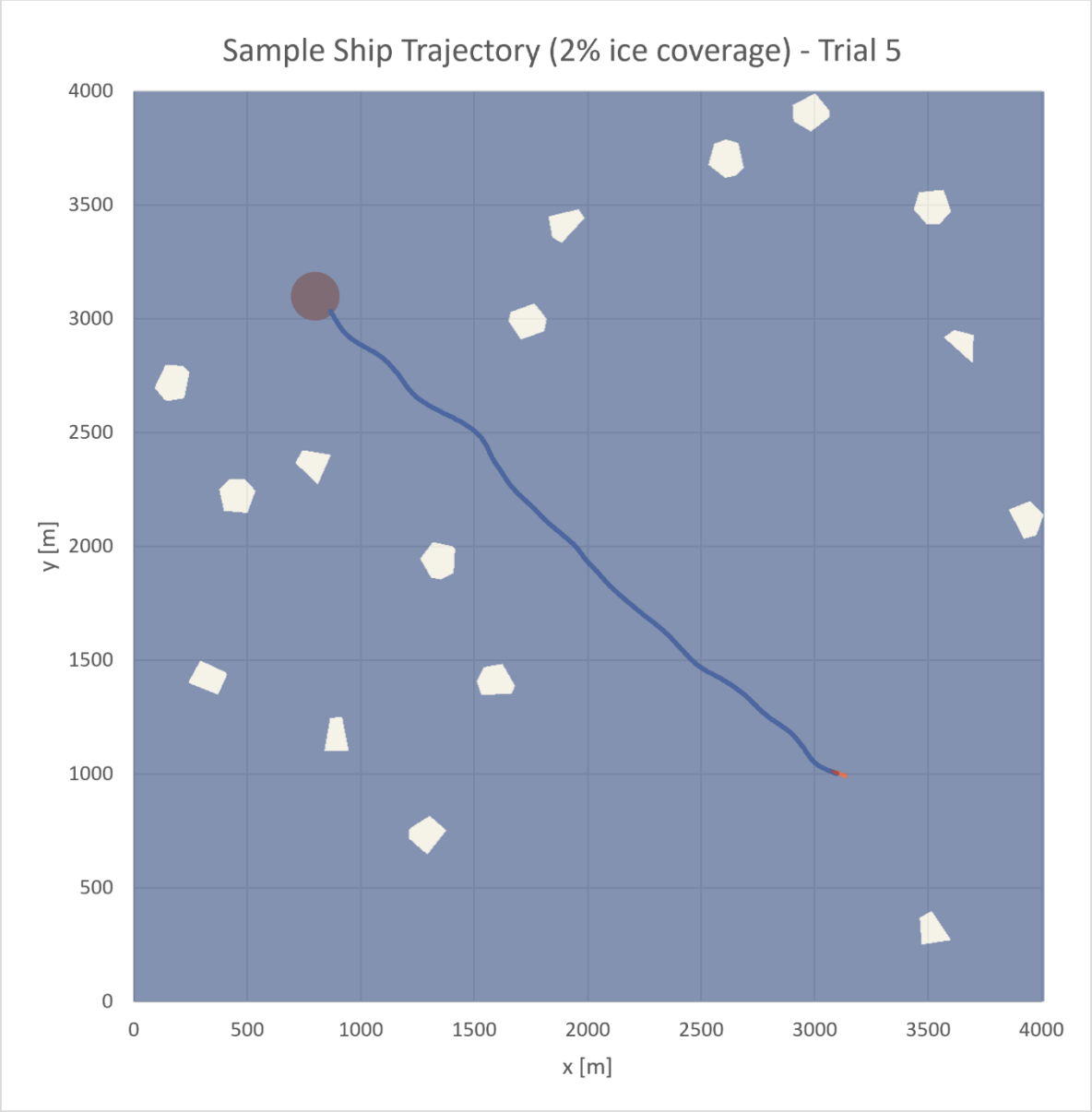
Appendix I – Ship Trajectory Graphs

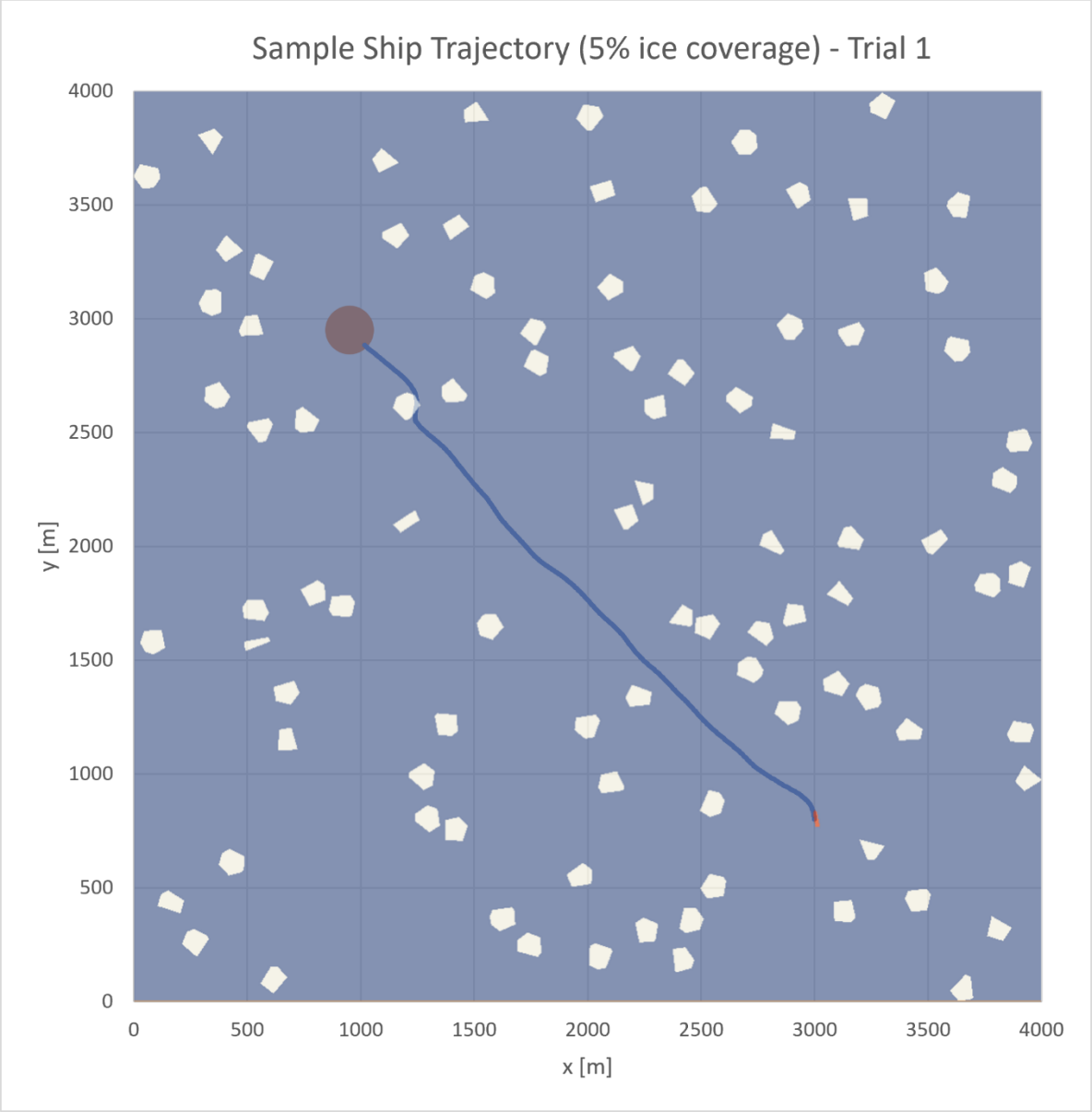


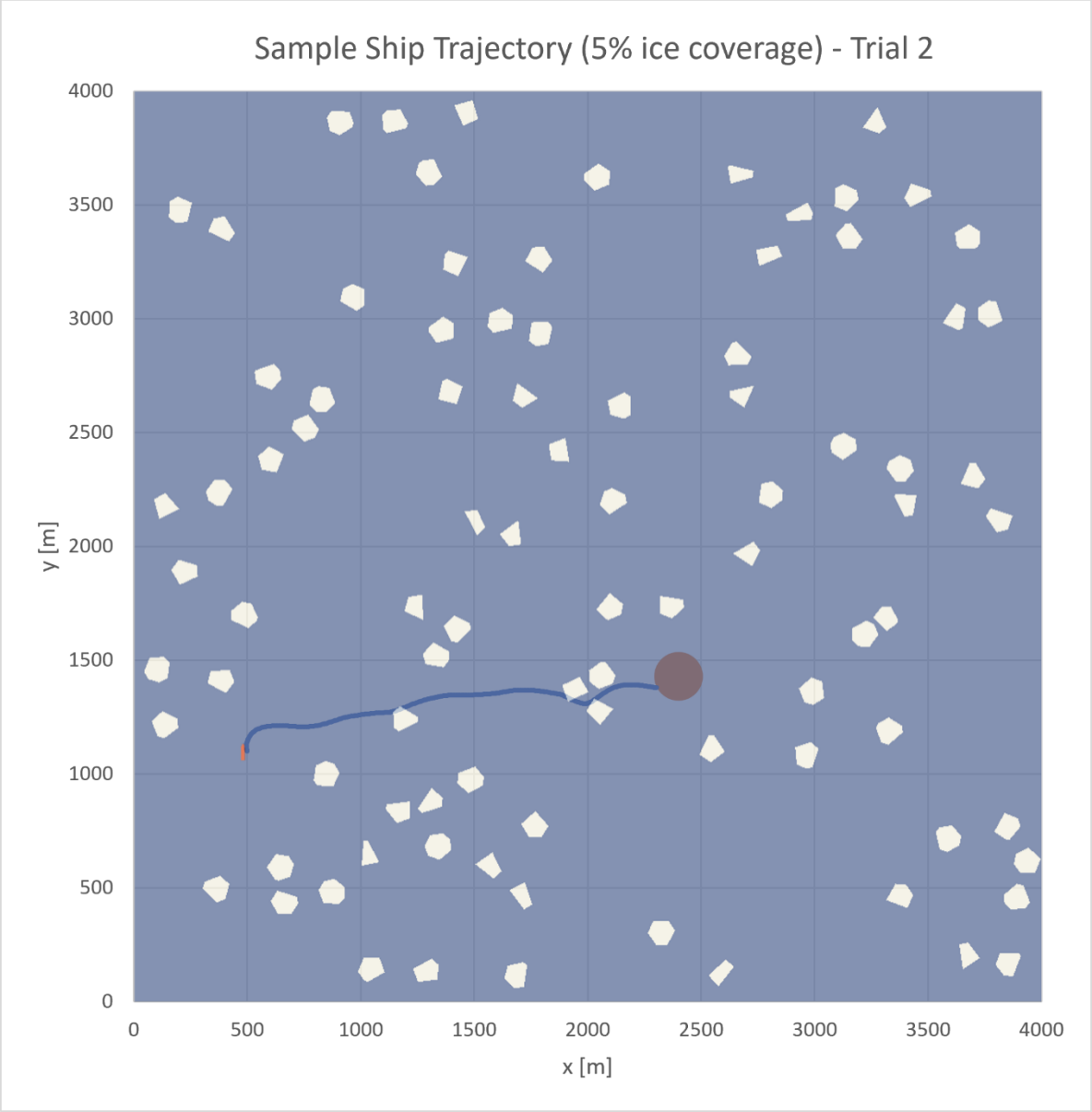

Sample Ship Trajectory (2% ice coverage) - Trial 2
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

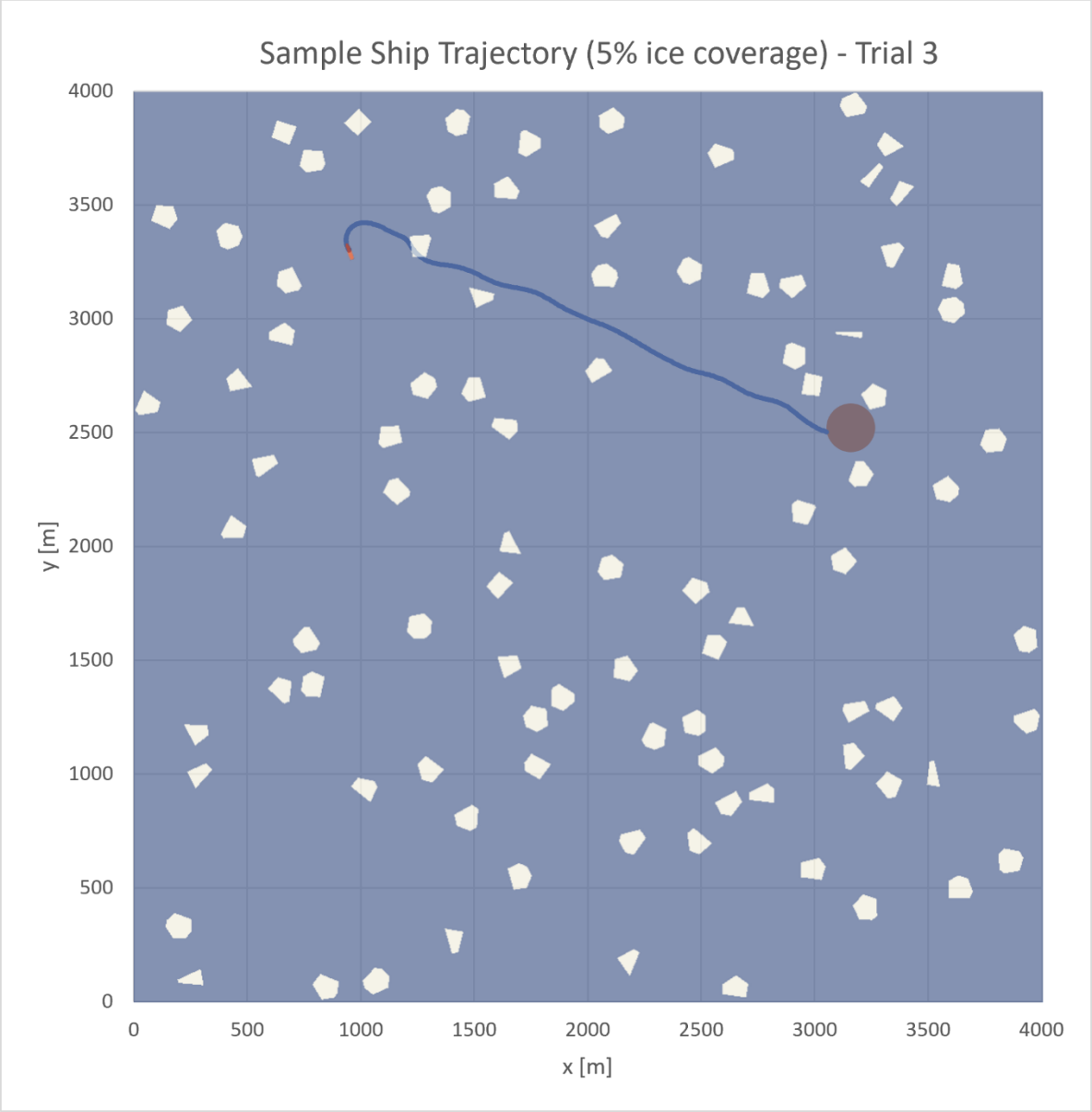

Sample Ship Trajectory (2% ice coverage) - Trial 3
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

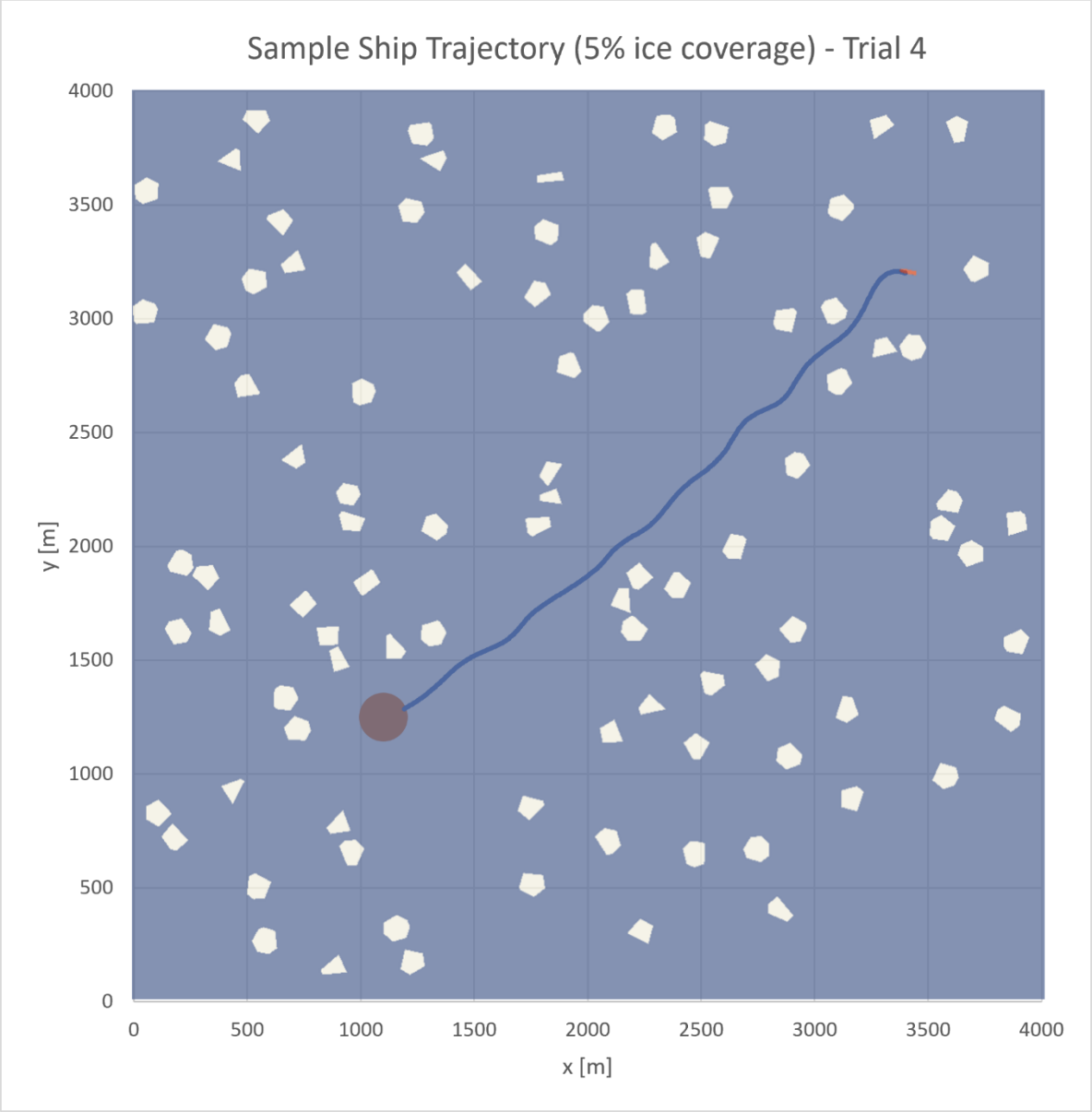

Sample Ship Trajectory (2% ice coverage) - Trial 4
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

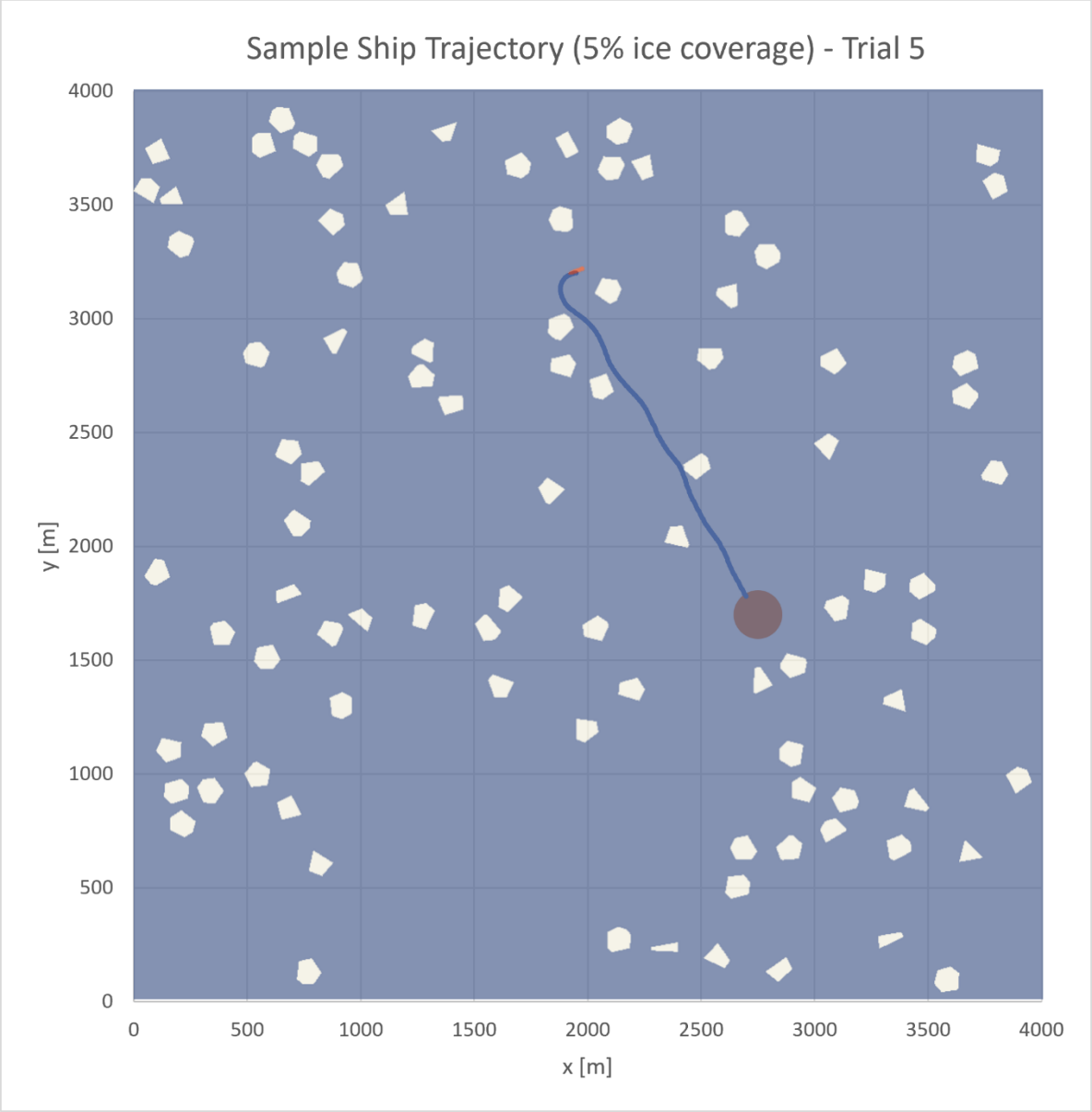

Sample Ship Trajectory (2% ice coverage) - Trial 5
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

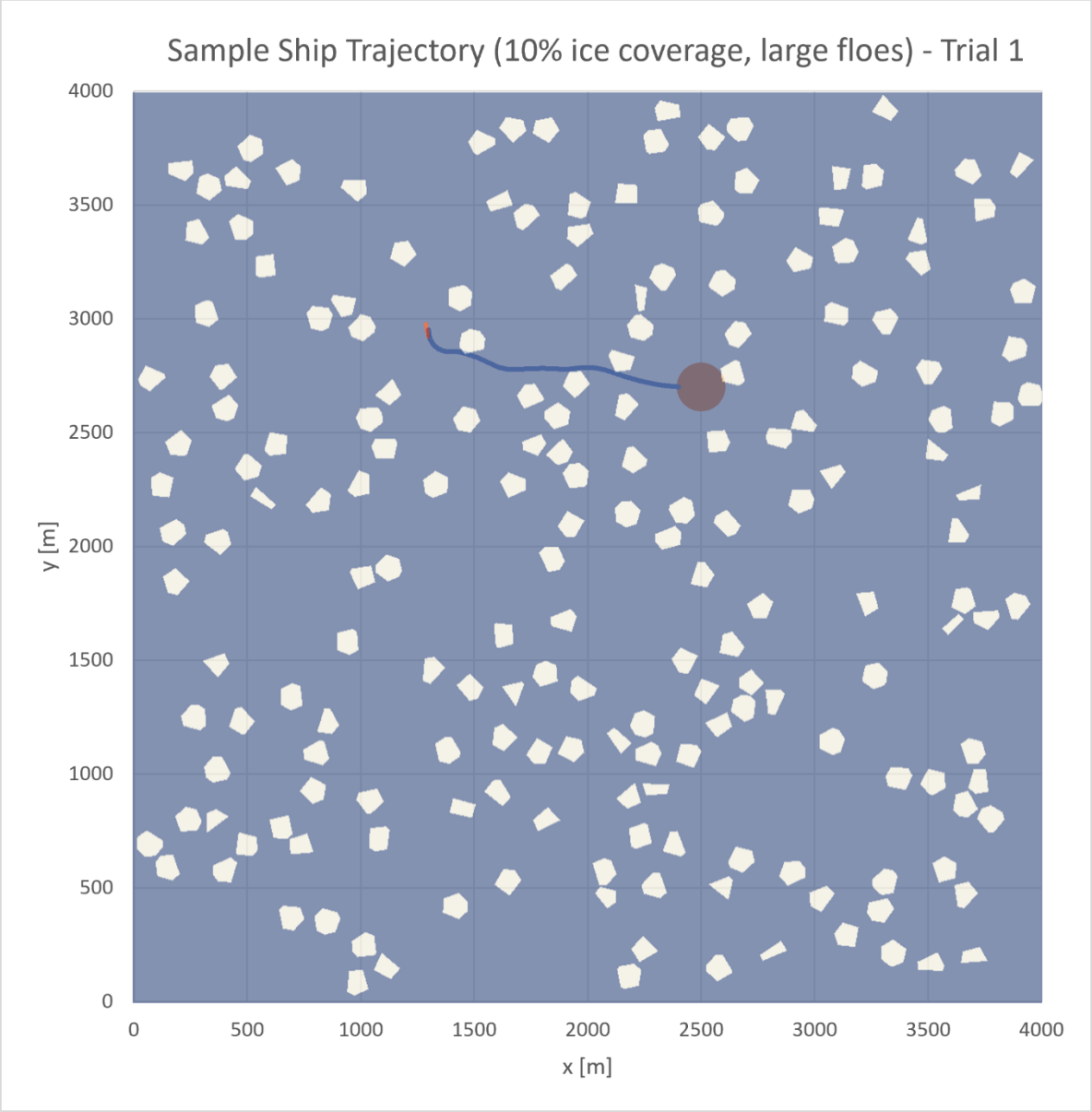

Sample Ship Trajectory (5% ice coverage) - Trial 1
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

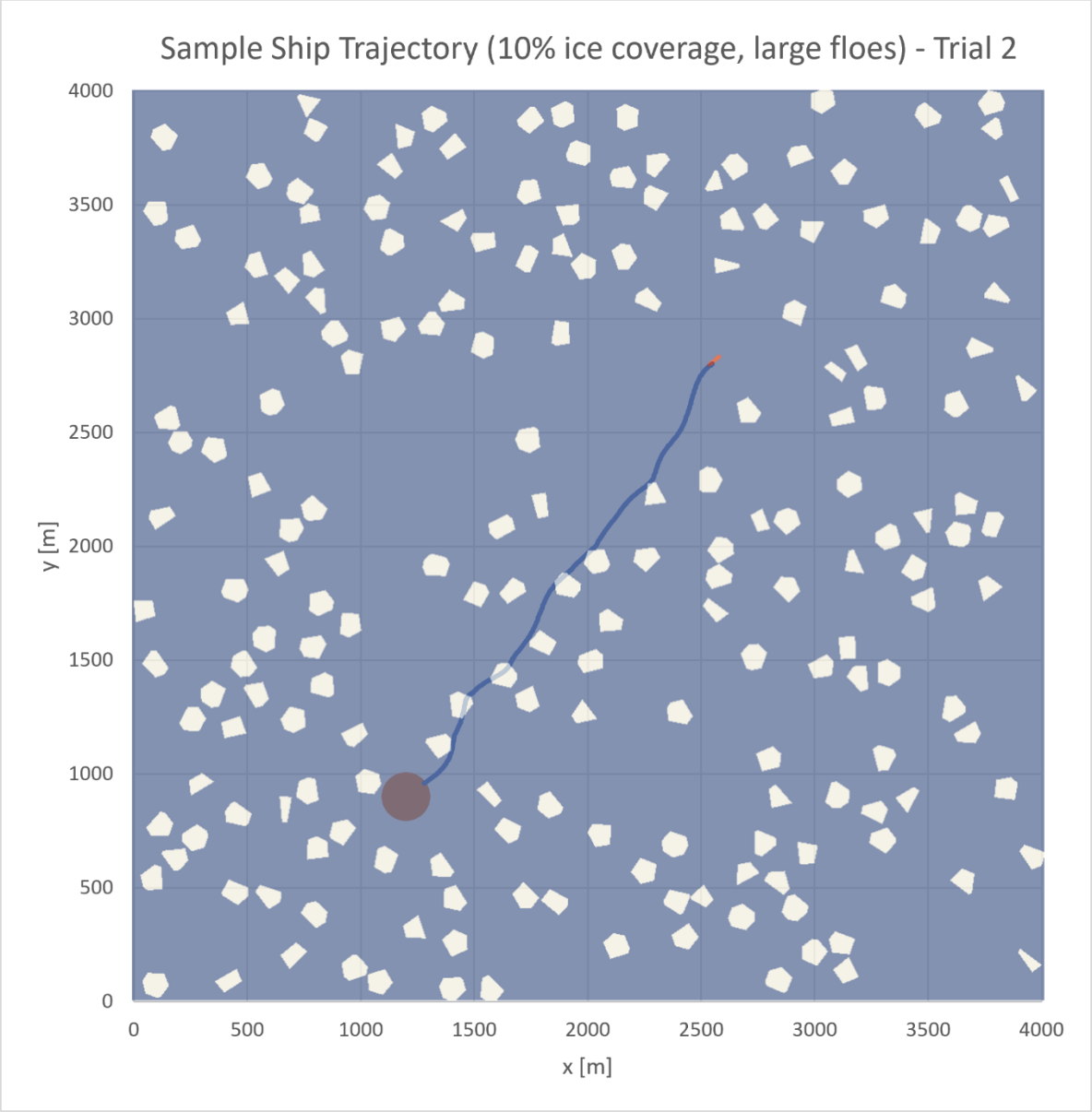

Sample Ship Trajectory (5% ice coverage) - Trial 2
y [m]
x [m]
0
500
1000
1500
2000
2500
3000
3500
4000

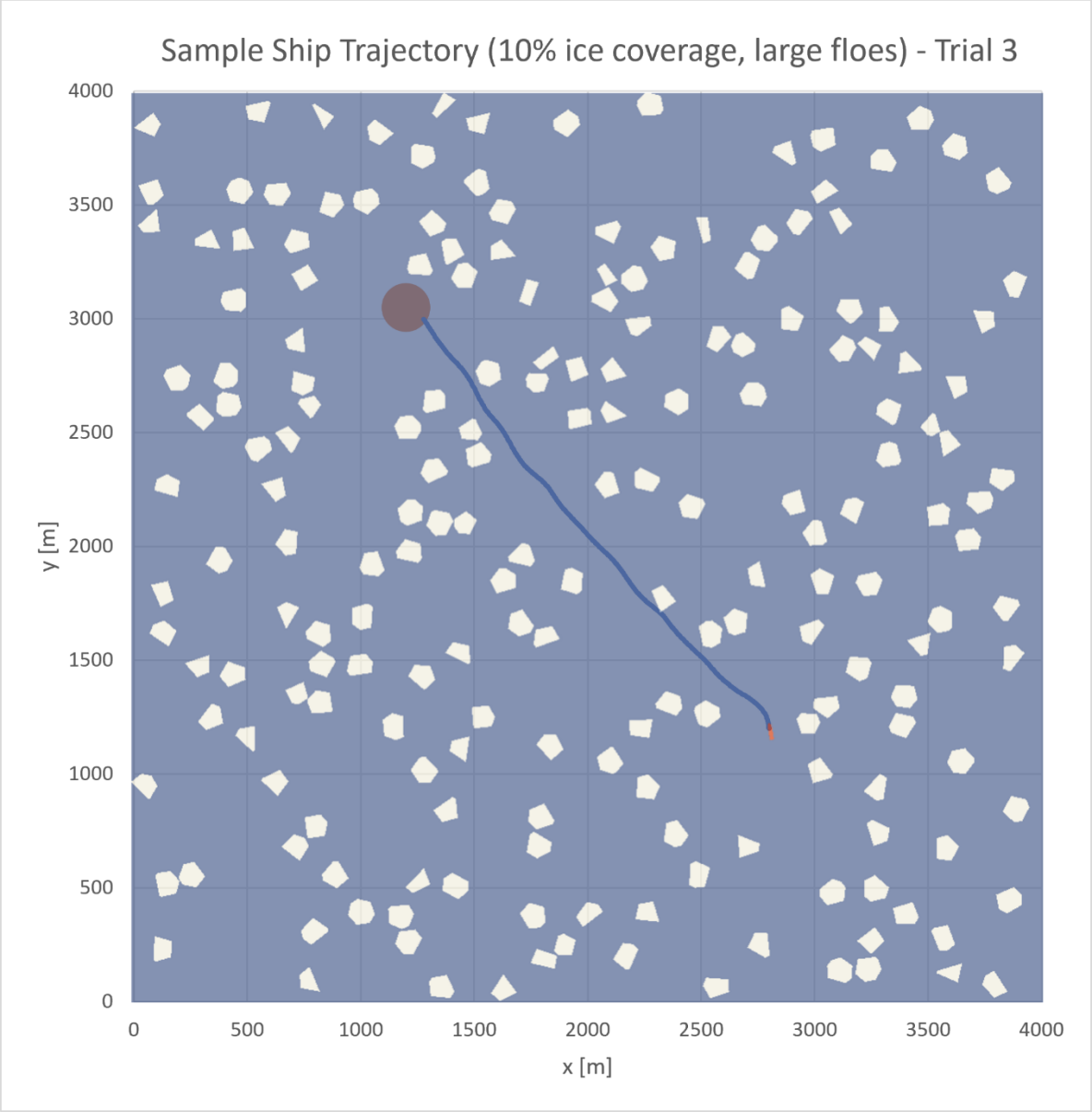

Sample Ship Trajectory (5% ice coverage) - Trial 3
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

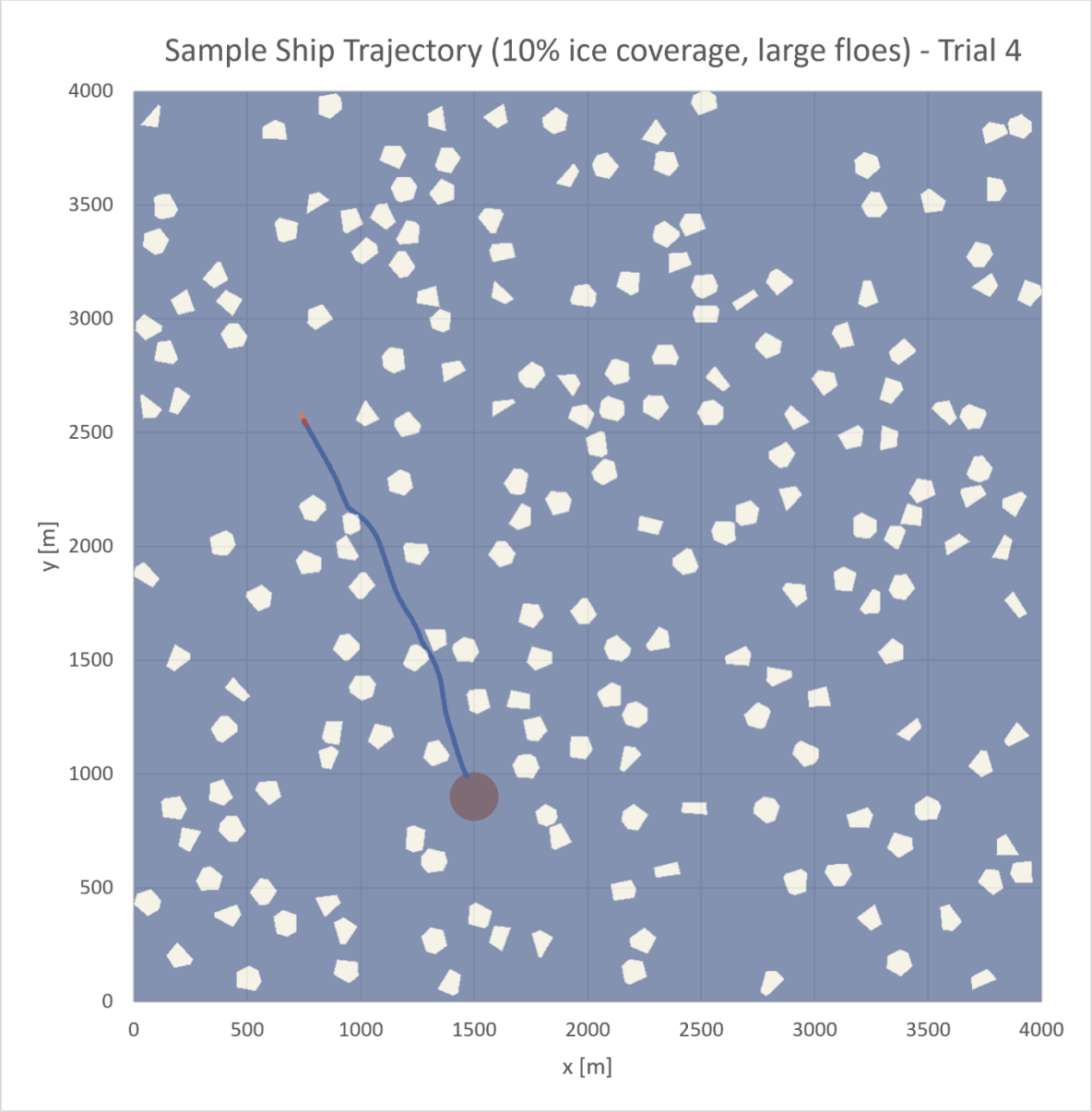

Sample Ship Trajectory (5% ice coverage) - Trial 4
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

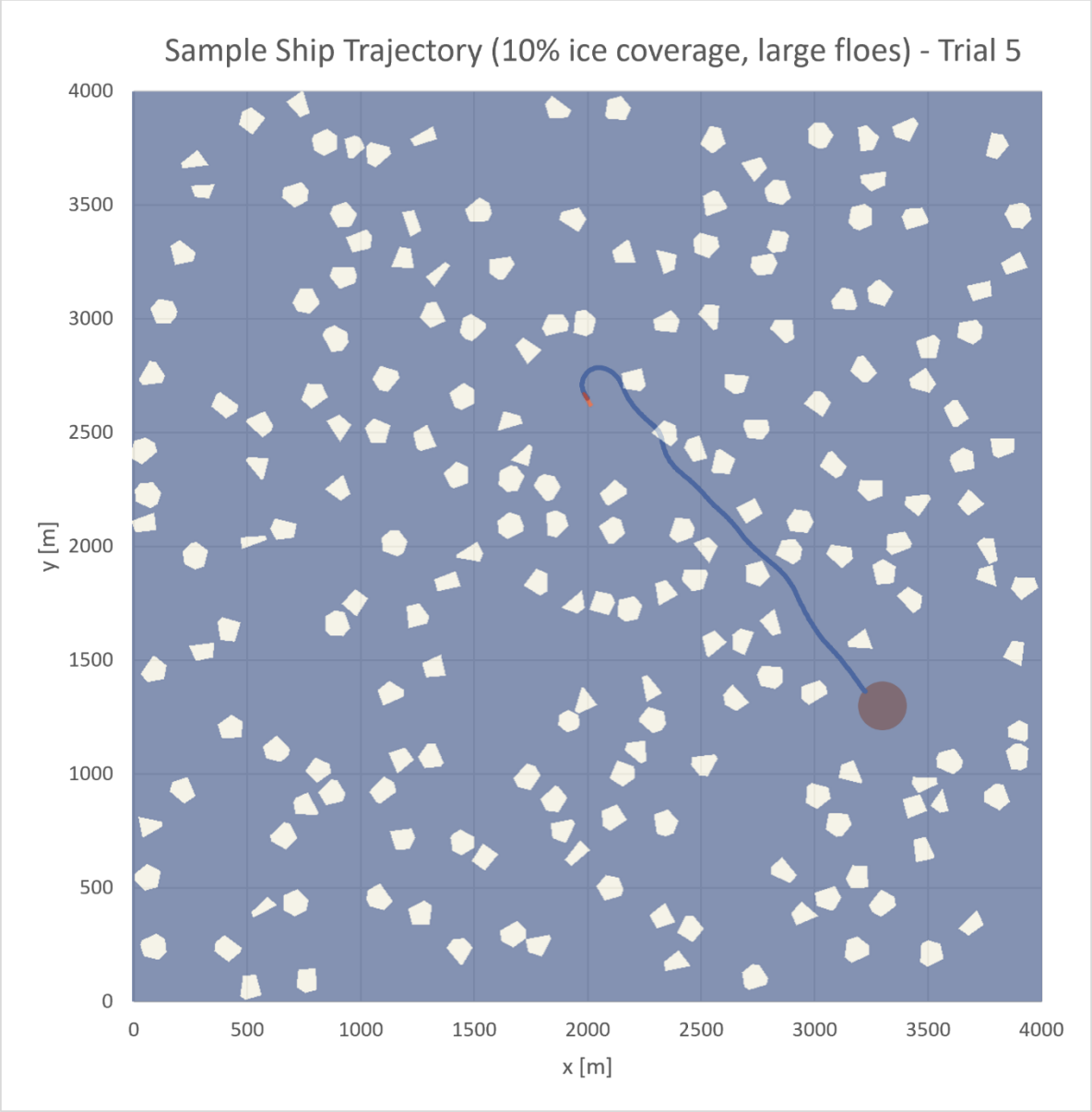

Sample Ship Trajectory (5% ice coverage) - Trial 5
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

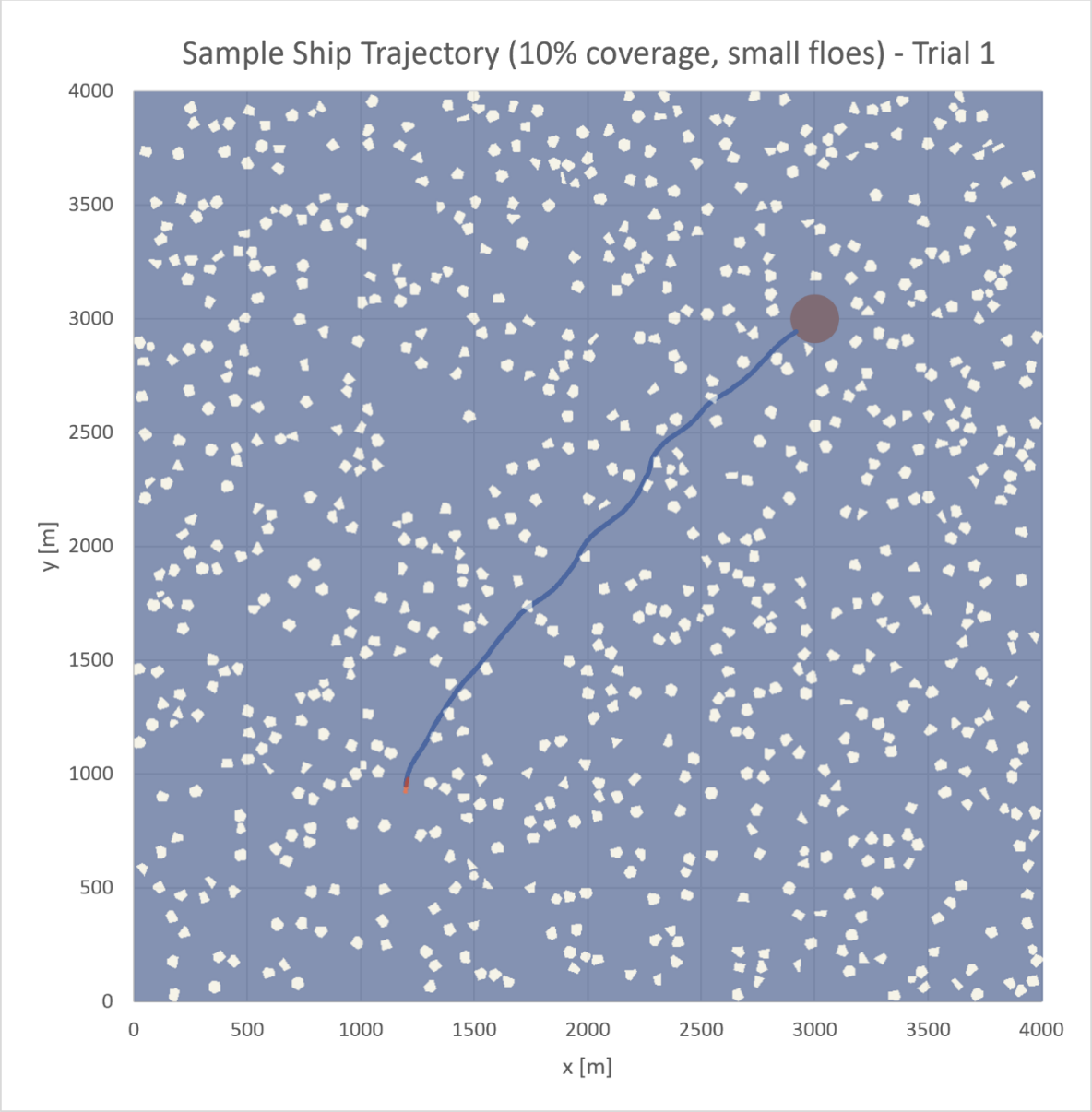

Sample Ship Trajectory (10% ice coverage, large floes) - Trial 1
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

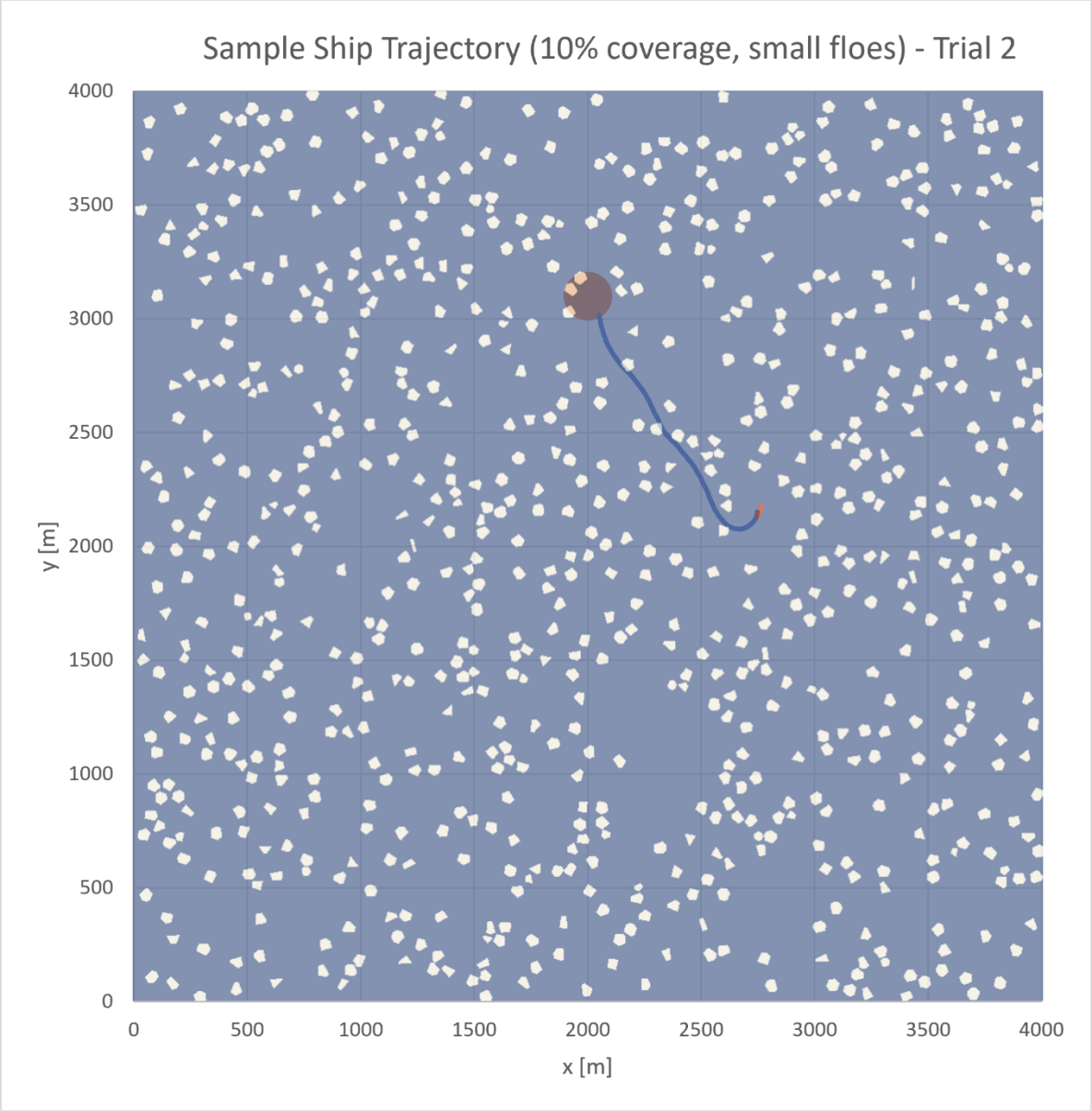

Sample Ship Trajectory (10% ice coverage, large floes) - Trial 2
y [m]
x [m]
0
500
1000
1500
2000
2500
3000
3500
4000

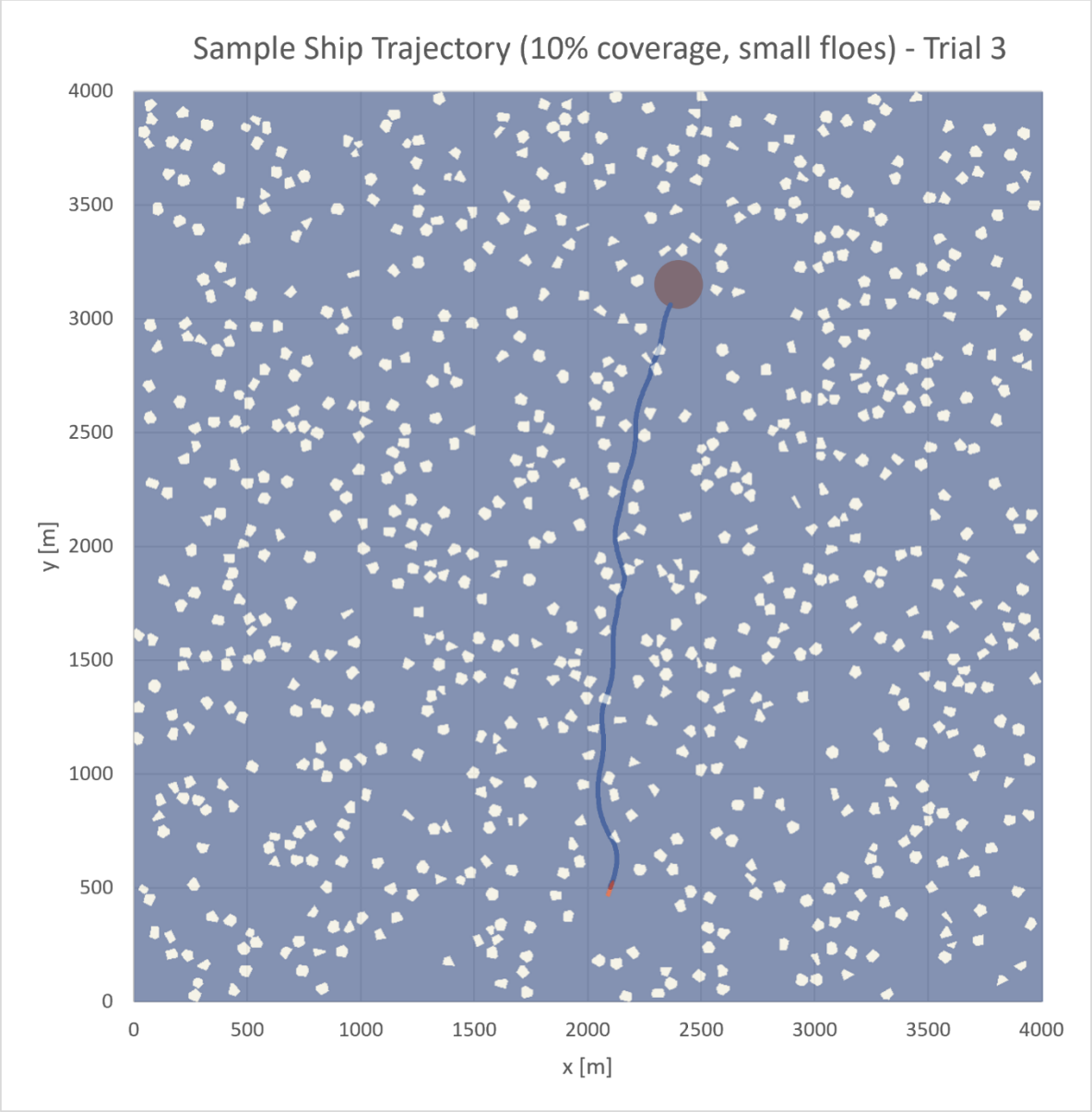

Sample Ship Trajectory (10% ice coverage, large floes) - Trial 3
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]

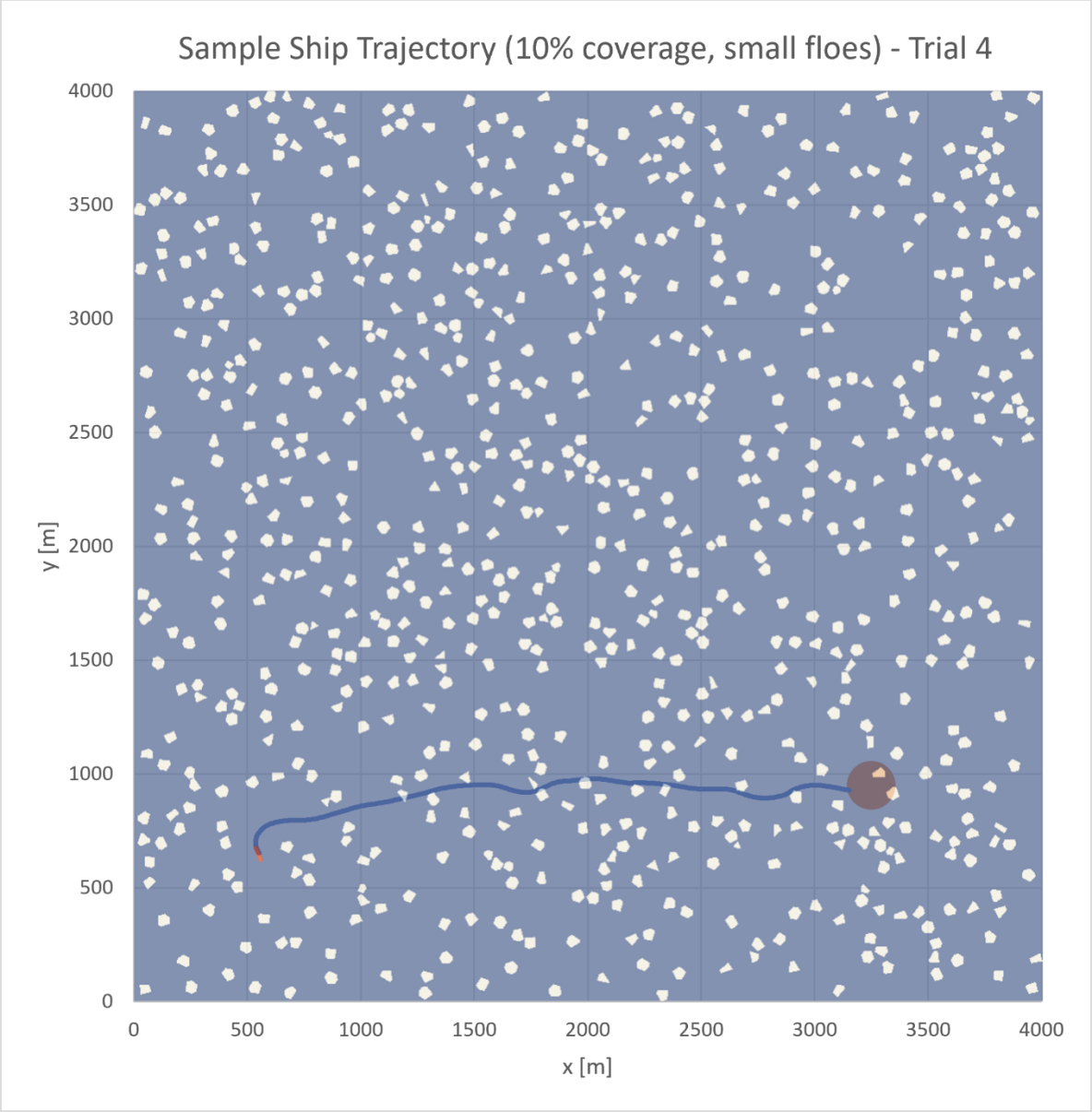

Sample Ship Trajectory (10% ice coverage, large floes) - Trial 4
y [m]
x [m]
0
500
1000
1500
2000
2500
3000
3500
4000

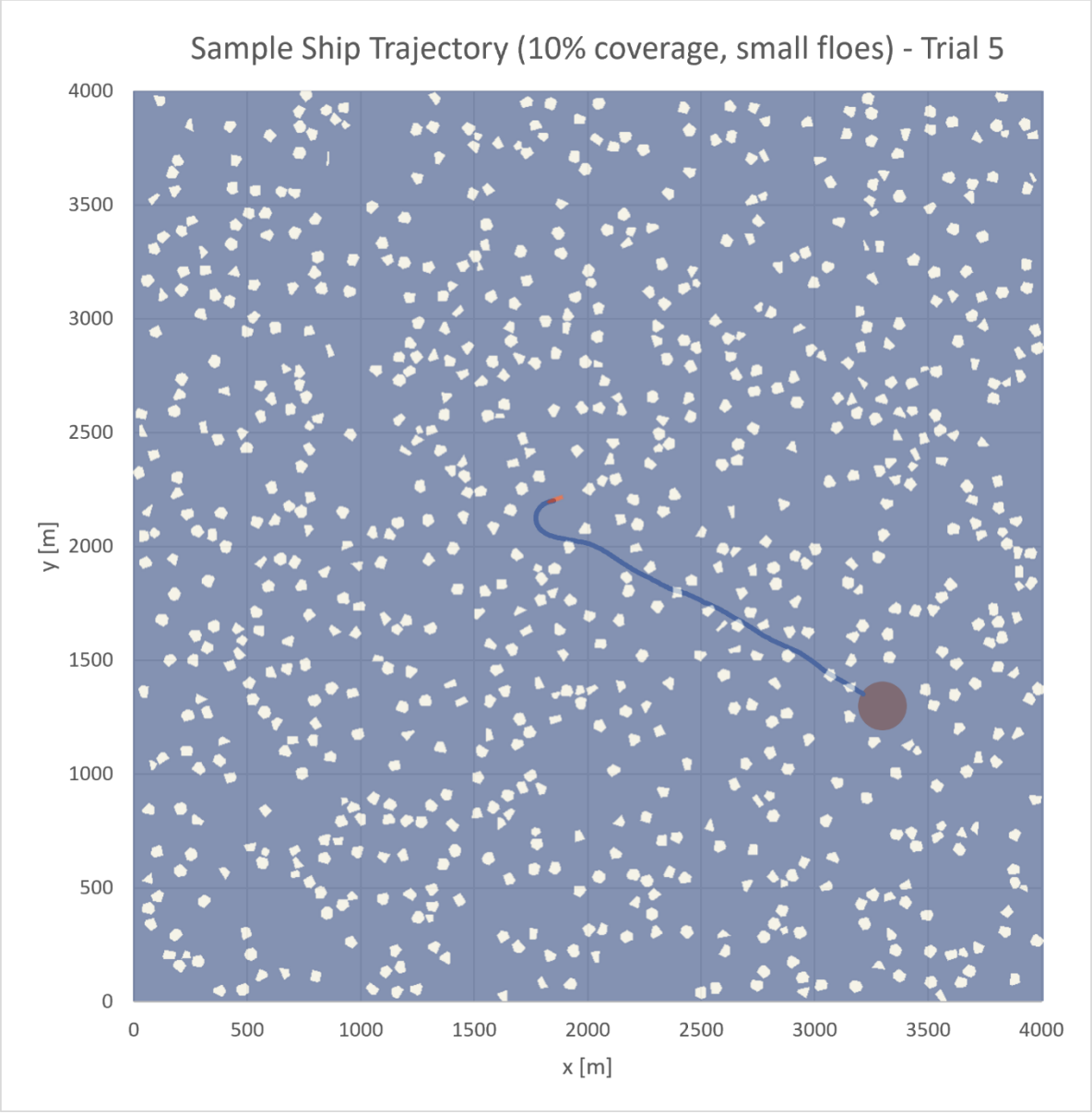

Sample Ship Trajectory (10% ice coverage, large floes) - Trial 5
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]


Sample Ship Trajectory (10% coverage, small floes) - Trial 1
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]


Sample Ship Trajectory (10% coverage, small floes) - Trial 2
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]


Sample Ship Trajectory (10% coverage, small floes) - Trial 3
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]


Sample Ship Trajectory (10% coverage, small floes) - Trial 4
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]


Sample Ship Trajectory (10% coverage, small floes) - Trial 5
4000
3500
3000
2500
2000
1500
1000
500
0
y [m]
0
500
1000
1500
2000
2500
3000
3500
4000
x [m]