Combining Supervised Learning and Digital Twin for Path-planning with Dynamic Obstacle-avoidance

Submitted by

Chanjei Vasanthan

Supervised by

Dong Trong Nguyen

NTNU

Department of Marine Technology
Norwegian University of Science and technology
Trondheim
24. June 2019

Preface

This master thesis is a continuation of my project thesis on reinforcement learning for collision-avoidance on vessels. This was written during the spring of 2020 at the Department of Marine Technology, NTNU, and concludes my master studies in Marine technology. The subject of the thesis is based on my curiosity of self-governing systems and programming in the intersection with control theory. Initially the idea was to develop a deep learning-based solution, but academic discussions, as well as conversations with the industry changed my direction. Especially, considering the huge gap between developing a deep learning-solution and actually deploying it on the actual problem due to the uncertain factors related to deep learning. Therefore, my motivation was to look into a solution approach based on the industry perspective on essential requirements.

Acknowledgment

First and foremost I would like to express my sincere thanks to supervisor Dong Trong Nguyen. Throughout the thesis, he has offered me valuable feedback, guidance and always been flexible in terms of supervision. I would also like to thank my fellow students for interesting academic discussions and enthusiasm, while working with the thesis. Additionally, I would like to thank the DNV GL office at Trondheim for assisting me on software guidance. Finally, I would like to thank my friends and parents for valuable support throughout the years.

Chanjei Vasanthan
24. June 2020, Trondheim

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MSC THESIS DESCRIPTION SHEET

Name of the candidate:	Chanjei Vasanthan
Field of study:	Marine control engineering
Thesis title (Norwegian):	**Kombinering av veiledet læring og digital tvilling for autonomt baneplanelgging med dynamisk kollisjonsunngåelse**
Thesis title (English):	**Combining Supervised Learning and Digital Twin for Autonomous Path-planning with Dynamic obstacle-avoidance**

Background

The goal of this thesis is to develop a model capable of generating a path from A to B using Bézier curves, while avoiding both static and dynamic obstacles, such as a real vessel with vessel dynamics. The aim is to develop a path-planning solution that complies with current rules and regulation. The model will therefore be based on a supervised machine learning-method which is chosen based on benchmarking. The model will be trained on a simplified model of a real vessel, and then verified on the real vessel to solve head-on situations based on rules extracted from COLREG. In addition, the digital twin would be added in the real vessel to conduct re-planning when an unforeseen act is executed by the head-on vessel.

Work description

1. Perform a background and literature review to provide information and relevant references on:
    - General Path planning-methods
    - Previous methods within marine vessel path-planning based on AI/Machine learning
    - Collision avoidance approaches
    - Autonomous vehicles/vessels
    - COLREG
    - Authority's view on autonomous vessel i.e. IMO and class societies.

    Write a list with abbreviations and definitions of terms, explaining relevant concepts related to the literature study and project assignment.

2. Develop simplified model (digital twin) of the research vessel Gunnerus which is to be used for training.
3. Develop a supervised model combined with Bézier curves to generate paths
4. Develop a maneuvering controller to navigate the path
5. Use the simplified model as a digital twin inside the real vessel to re-plan steps
6. Conduct simulations on a model of the research vessel Gunnerus

Tentative:

7. Write a journal paper submitted to Control Engineering Practice.

Specifications

The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts., and it is not expected to be longer than 60-80 A4 pages, from introduction to conclusion, unless otherwise agreed upon. It shall be written in English (preferably US) and contain the following elements: Title page, abstract, acknowledgements, thesis specification, list of symbols and acronyms, table of contents, introduction with objective, background, and scope and delimitations, main body with problem formulations, derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with a printed and electronic copy to the main supervisor, with the printed copy signed by the candidate. The final revised version of this thesis description must be included. The report must be submitted according to NTNU procedures. Computer code, pictures, videos, data series, and a PDF version of the report shall be included electronically with all submitted versions.

Start date:	**1**5 January 2020	**Due date:**	As specified by the administration.
Supervisor:	Dong T. Nguyen		

Trondheim, 24.06.2020

[illegible]

Dong T. Nguyen
Supervisor

Abstract

In this master thesis a supervised model is developed to generate a cubic Bézier curve. The aim is to develop a curve that avoid collision, comply to COLREG and attempt to take the shortest path whenever possible. To achieve this a score paradigm was developed, which is used to collect the optimal pair of control points used to generate the path. To sample data, a digital twin is developed in Simulink of the research vessel R\V Gunnerus and converted to FMU-format to generate a simulator in Python. The digital twin is a simplified model based on control plant model as defined in control theory. Further, the simulator was used to sample training date for the machine learning model. The machine learning model was chosen based on benchmarking of available models, which resulted in Gradient Boosting regressor. To follow the path a traditional maneuvering model based on Roger Skjetne's the Maneuvering problem was developed.

The motivation behind the proposed solution is mainly addressing the weaknesses of a deep learning-based solution such as unpredictability of the actions, and lack of transparency. Therefore, the developed solution is based on a guideline for autonomous solutions presented by DNV GL. The aim was to comply to the expectation of the authorities, while still working on state of the art solutions. To verify the model, simulations with static obstacle and dynamic vessel was simulated using an accurate model of Gunnerus. The aim was to simulate head-on situation to evaluate if the path-planner could generate paths that complies with COLREG rule 14a). In addition, to consider unexpected actions from the head-on vessel, the digital twin was applied on the research vessel. The objective was to forecast possible collision by running simulations on the digital twin based on the real states of the vessel during mission. If a collision was forecast a new path was generated for the real vessel. From the results, it was concluded that the path-planning model performed generally well, showcasing satisfying behaviour both during obstacle-avoidance, as well as re-planning. However, the model do have some limitations such as no environmental variables considered and limited state-space. It is noted that to consider environmental variables, accurate measurements are necessary, which is not always possible. Hence, an alternative solution was proposed where the problem can be delegated to the controller by introducing i.e. hybrid controller. Likewise, to solve the limited state-space an approach similar to way-point tracking is suggested. Such that the way-points are placed within the trained environment, and the environment is moved and reset as a way-point is reached.

Sammendrag

I denne masteroppgaven undersøkes en veiledet læringsmetode for å generere kubiske Bézier kurver. Målet er å utvikle en kurve som unngår kollisjon, følger COLREG og forsøker å velge den korteste banen, når mulig. For å oppnå dette utvikles et poengsystem som brukes for å finne den optimale kombinasjonen av kontrollpunkter for å generere kurven. For å samle treningsdata brukes en digital tvilling i Simulink basert på forskningsskipet R\V Gunnerus, som ble konvertert til FMU-format for å produsere en simulator i Python. Den digitale tvillingen er en forenklet modell av kontroll plantemodellen slik den er definert i kontroll teori. Videre brukes simulatoren for å generere treningsdata for maskinlæringsmodellen. Maskinlæringsmodellen ble valgt basert på sammenlikning blant tilgengelige modeller, hvor valget falt på gradient boosting regresjon. For å følge banen utvikles en tradisjonell manøvringsmodell basert på Roger Skjetnes the Maneuvering problem.

Motivasjonen bak denne løsningen er hovedsakelig å adressere noen av problemstillingene knyttet til dyplæring-baserte løsninger som for eksempel uforutsigbarhet og mangel på åpenhet av modellen. Derfor er modellen basert på en retningslinje for autonomone løsninger presentert av DNV GL. Målet er å utvikle fremtidens løsninger samtidig som man innretter seg til myndighetenes forventninger. For å verifisere modellen ble simuleringer med statisk hindring og dynamisk skip simulert ved å bruke en sannferdig modell av Gunnerus. Målet var å simulere front mot front situasjon for å se om baneplanleggeren kunne generere kurver som fulgte COLREG regel 14a). I tillegg ble uventende handlinger av motkommende skip tatt i betraktning, ved å anvende den digitale tvillingen på det ekte skipet. Målet var å predikere mulig kollisjon ved å simulere den digitale tvillingen med nyeste tilstandsvariablene til det ekte skipet. Hvis en kollisjon ble predikert, ville en ny oppdatert bane bli generert for det ekte skipet. Fra resultatene ble det konkludert med at baneplanleggeren generelt gjorde det bra, både ved kollisjonsunngåelse og re-planlegging. Imidlertid innehar modellen enkelte svakheter som å ikke ta i betraktning variabler fra omgivelsen, samt begrenset område. Det er påpekt at for å ta omgivelse i betraktning kreves det nøyaktig målinger, noe som ikke alltid er mulig. Derfor ble en alternativ løsning presentert, hvor problemet delegeres til kontrolleren, for eksempel en hybrid kontroller. På samme måte, for å løse det begrensede området foreslås det å utvikle en løsning basert på veipunkt-følging. Slik at punkter på veien er plassert innenfor en tilsvarende område, og området forflyttes når et veipunkt er nådd.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial intelligence. 1
BM	Breadth middle. 40
CLF	Control Lyapunov function. 42
COLREG	International Regulations for Preventing Collision at Sea. 5, 26
CS	Computer Science. x, 1
DDPG	Deep Deterministic Policy Gradients. 7
DNV GL	Den Norske Veritas Germanischer Lloyd. 9
FMI	Functional Mock-up Interface. 11
FMU	Functional Mock-up Unit. 40
FRP	Formation Reference Point. 5
GNSS	Global Navigation Satellite System. 3
GPS	Global Positioning System. 4
IMO	International Maritime Organization. 5, 9, 26
LOA	Length over all. 40
LPP	Length between perpendicular. 40
MPC	Model Predictive Control. 5
MRC	Minimum Risk Condition. 10
MSE	Mean-squared error. 31
NFL	No Free Lunch-theorem. 28

NTNU	Norwegian University of Science and Technology. 13, 40
RSS	Residual Sums of square. 35
SC	Self-Controlling. 9
SOLAS	Safety Of Life At Sea. 9
UGES	Uniformly Globally Exponentially Stable. 44

Chapter 1

Introduction

1.1 Background

Following the recent breakthroughs within the Artificial Intelligence- (AI) and Machine learning-field, combined with the advancement of computation power, the popularity of the topics have increased tremendously. This can be confirmed by studying the statistics presented by AI Index in figure 1.1. As the article [1] highlights, the amount of recent publications related to the subjects have surpassed all expectations. Naturally, this has lead to an extensive research on

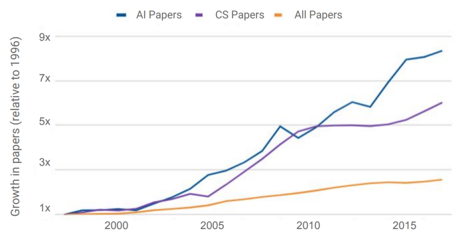

Figure 1.1: Growth of annually published papers by topic within AI, Computer Science (CS) and general topics (1996–2017)

new potential domains of application. One such particular domain that will be considered in this thesis are autonomous vehicles. An autonomous system is defined as a system capable of decision-making without human interference [2]. A well-known example of this is the self-driving car, which has the ability to maneuver by itself in traffic. Few decades ago, this was considered a dream scenario, but have have now become achievable owing to companies such as Tesla. The revolution within self-driving automobiles has not unsurprisingly stimulated research within the maritime industry as well. One such outcome is the upcoming zero-emission autonomous container ship Yara Birkeland. This was considered as a revolutionary announcement by Yara and the Kongsberg group in 2018, and is currently under development [3]. Similar to self-driving cars, autonomous vessels aim to possesses the ability to navigate in the sea without human interference. However, compared to the roadway, the ocean can be classified as significantly complex environment. Especially, when considering environmental forces such as currents, weather, waves and collision-avoidance. Based on this, it is certain that an autonomous vessel demands high level of intelligence for maneuvering and evaluating risk factors. This comes in addition to the ability to re-plan the mission in terms of reaching the goal, when unforeseen scenarios occurs. Naturally, this brings up the question of what the true motivations behind autonomous vessels are. The most obvious answer is unquestionably cost savings. In [4] dry-bulk was identified as a suitable candidate for unmanned vessel. A dry-bulk is typically a slow navigating vessel with simple loading/off-loading-condition, thus requiring minor human intervention. It is estimated that the expenses related to the crew alone could make up to 10% of the trip expenses, thus reflecting the potential savings of an unmanned vessel. In addition, there are also opportunities for reduction of fuel emission. For instance, Enova has estimated that Asko's new pair of autonomous electric vessels possibly can save the transportation industry for 5.000 ton CO2-emission yearly. Thereby reducing two million kilometer of travelling on the roadway [5]. The final incentive is augmentation of the safety on the ocean. By developing intelligent systems capable of obeying the rules and regulations at the sea, one can potentially avoid typical human mistakes. Indeed, the fatal consequence of such an example was recently observed with Helge Ingstad [6].

1.2 Literature review

1.2.1 Autonomy vs automation

The misunderstanding of the differences between autonomy and automation have commonly led to the terms being interchangeably used. This may be related to the evolution of the concept *intelligence* in the industrial domain. For instance, in the 1940-1960s one would have identified automation systems as intelligent systems [7]. However, in present time we would associate control systems like Dynamic positioning as an automatic system, because our perception of intelligent systems have evolved. There exist several frameworks describing the distinction between autonomy and automation by levels. In [8] a 10-level scale of degree developed by Sheridan is presented. This is a widely recognized model which is referenced to in many literature. The first level is defined as only human based intervention, while the seventh level indicates automatic decision making. At the tenth level the computer fully ignores the human and is able to execute work autonomously. A much broader and well-defined definition is presented in [9] which is based on National Institute of Standards and Technology's definition of level of autonomy. Here autonomy is described as a system's ability to sense, perceive, analyze, communicate, plan, make decision and make action such that it can achieve the goals it is assigned to. Based on the definition of autonomy, a general discussion on approaches towards using AI for autonomous ship navigation can be found in [10].

1.2.2 Structure of an autonomous vessel

In the process of developing autonomous marine vessels, there are many areas that potentially can be researched upon. However, based on [11] we can reduce the area to four important concepts or modules that have to be addressed: Path-planning, Guidance, Navigation and Control. Path-planning considers the problem of generating a suitable path to follow. When generating a path several dynamic constraints have to be considered related to the vessel, such as the curvature, velocity or acceleration. Guidance on other hand, is related to the computation of desired position, velocity and acceleration of the marine vessel. In other words, information that is fed into the control system. Navigation is related to monitoring of the vessel while moving from one position to another. This is done through logging the vessel position, distance travelled or the velocity. A widely known tool for this is the Global Navigation Satellite System (GNSS). The final module, control is the step that determines the essential forces and moments of the actuators in the vessel, such that the vessel can reach the control objective. Example of an objective includes, but not limited to: set-point tracking, path-following and maneuvering control. Notice that although the process is divided into four modules, in principle all the modules are interconnected.

Path-planning for vessel

Usually when we start driving a car or sailing a ship we have a common objective: To reach a desired place or position. In the old days, one would get into the vehicle and start driving until the final destination was reached. Naturally, this involved several downsides such as detours, increased time spent driving as well as increased fuel usage. As a consequence, the development of built-in Global Positioning System (GPS) happened, a device popularly used to generate a pathway through a mobile device or computer. Because the task of driving solely depends on the human, and the human primarily needs the direction, the constraints on the path-planner are naturally few. On other hand, for an autonomous vehicle, the task now remains entirely on the vehicle itself. Hence, much more complex and strict constraints are essential. Based on our own experience we can easily draw relevant examples such as maintaining a satisfying velocity and acceleration while driving, keeping an appropriate turning-rate when swinging the car, or avoiding other cars (collision-avoidance) on the road.

Control system for vessel

For a control system to be able to steer the vessel correctly based on the generated path, a guidance law (algorithm) is required. By computing appropriate reference trajectories based on the desired dynamics through the guidance law, the controller is able to maneuver the ship safely. As stated in [11] this requires stability of both the guidance law and the controller. However, proving stability of such a system by finding a unique Lyapunov function which incorporates all the states can be difficult. Hence, it is apparent that both the guidance law and the control design are affiliated. Based on [12] one can separate the control objective in two different problems: the tracking problem and the path-following problem. In the tracking problem the goal is to track a point tracing out the path, while the path-following problem considers the whole path. In the latter, the control objective is to follow the path defined as the *geometric task*. However, in the former the system also have to satisfy dynamic constraints such as a desired speed or acceleration, leading to the second task, the *dynamic task*.

1.2.3 Path-planning algorithms

Within the field of path-planning and maneuvering there have been carried out extensive research on several strategies, mainly within robotics. Among these is the Dijkstra's algorithm, a well-known algorithm which is guaranteed to find the shortest path between two points. In [13] it was applied with multi-layer dictionaries for robotic path-planning taking energy and rotational capabilities into consideration. The A* (A-star) algorithm is a more superior algorithm which is also widely recognized. It combines the best features of Dijkstra's algorithm in finding the shortest path, while being faster with the help of a heuristic (greedy approach). A modified version of A* was applied for underwater vehicle considering various ocean currents in [14]. However, in [15] pointed out that if the ocean current exceeded the vessel speed, the algorithm will face complications fulfilling the task. In other words, this illustrates that developing a path-planning algorithm in a complex environment such as the ocean can be challenging. A major drawback of these studies is that they only consider static object-avoidance. In [16] various algorithms for both static- as well as dynamic object are discussed, although only for aerial vehicles. One such approach is using Kalman Filter to predict the motion of the object, thereby predicting its path. Other mentionable algorithms are the Genetic algorithm [17], Angle potential field [18] and Rapidly-exploring random tree [19]. It has to be noted that the presented algorithms only represents a small portion of the strategies available, which obviously cannot be summarized to a couple of pages.

So far, the algorithms considered are mainly applied in a (x, y)-environment. However, for a marine vessel the heading also plays a substantial role during maneuvering. In [20] a modified A* algorithm named Theta* algorithm (Theta-star), was applied in research of path-planning for marine surface vessel. Unlike A*, the Theta* is also able to consider the angle. In fact, [21] justified that in general, Theta* performed better than A* both in terms of computing time, as well as finding the optimal path. Despite the satisfying results, it has to be noted that the simulations were executed with only static obstacles without any environment forces. In [22] a collision avoidance strategy for multiple marine vessel formation control was developed. Based on the Formation Reference Point (FRP) a virtual structure was built for the collision avoidance. To model some of the non-linear dynamics, an adaptive neural-network was used. [23] proposed a novel approach on building a path-planning algorithm based on potential flow. Although the probability of object-collision as well as possibility of producing very long paths were emphasized as major drawbacks, the results showed a huge potential for a velocity potential-based solution. Meanwhile, in [24] an early stage attempt on generating path based on Bézier-curve was applied, including collision-objects and constraints on the curvature. Despite the early stages, the proposal still showcased a promising approach in developing suitable paths.

Similar to traffic rules, the International Maritime Organization (IMO) has designed Convention on the International Regulations for Preventing Collisions at Sea (COLREG). This convention contains among other things, navigation rules to be followed during sailing to prevent collision with other vessels. Consequently, it is expected that an autonomous vessel is able to comply to these rules while sailing. In [25] COLREG was applied combined with a Model Predictive Control (MPC). The simulation was carried out with both current from North and East in proportion to the heading, which proved to be successful. However, during collision-avoidance virtual objects was applied. Hence, real-life motion of an object such as a vessel was not considered. The model also showcased difficulties in cases of multiple obstacles. A corresponding approach using MPC was conducted in [26] with compelling result, even with drastic changes in the environment. Unlike the previous study, this one simulated with both wind speed of 15 m/s as well as ocean currents in order of 0.5 m/s, while considering COLREG.

1.2.4 Vessel Path-planning using AI

The research mentioned until now have mainly revolved around traditional algorithms and control strategies. But as mentioned initially, the breakthrough within AI and machine learning has made new set of tools available. The objective in this thesis is to look into how this can be applied in ship navigation to increase the autonomy of the vessel, while considering collision-avoidance. Hence, a detailed discussion on previous work is necessary. Therefore, the aim of this chapter is to shed light on previous achievements to enlighten the reader. Additionally, discussions on their advantages and drawbacks would be discussed to reveal possible gaps that should be addressed in the thesis.

In general, supervised machine learning have mainly been applied for problems such as ship fuel consumption application [27] and vessel telemetry system [28]. This is due to the fact that deep learning generally have shown remarkable results, leading to neglection of machine learning-based solutions. One of the main research currently carried out within the intersection of deep learning and robotics are application of deep learning agents to control or maneuver a system. Several examples can be found such as [29], [30], [31] and [32] to mention a few. Even the new deep learning-tool implemented in Simulink, a model-based design tool which is extensively used in developing dynamic systems, is designed such that one simply can replace the control block with a reinforcement learning-agent [33]. As a consequence, in the preceding research leading up to this thesis [34] an application of reinforcement learning for vessel maneuvering with collision-avoidance was studied. The results, showcased the vessels ability to maneuver successfully from an initial state to a terminal state. However, due to constraints on the state-space in the North-East frame, the model suffered limitations. This was due to the fact that increased state-space naturally led to increased training time on the model before convergence. As a consequence, one therefore cannot fully assure that the model can generalize well for an arbitrary environment. Secondly, the experiment studied only static object-avoidance. Meanwhile, collision avoidance of dynamic object i.e other vessels, are of significant importance in marine environment. For the algorithm to adapt to dynamic objects, an extension of the state-space was required such as considering environmental variables, which also would lead to increased training time. In addition, since the model implicitly calculates the optimal action based on cross-track error for straight-line, one does not have prior knowledge on the actual path when the vessel is approaching an object.

As mentioned in the previous section, a noteable disadvantage of traditional reinforcement learning is the limitation of the state-space. This led to the development of deep reinforcement learning. Deep reinforcement learning has the advantage of efficiently working with high dimensional state-spaces, while still being effective in regards to training time. For continuous action-space Deep Deterministic Policy Gradient-method (DDPG) have proven to be promising. In [35] a path-planning and collision avoidance approach based on DDPG was proposed. The goal of the agent was to steer the rudder angle such that the vessel reached the terminal state. The experiment was first conducted in an environment with no obstacles trying to solve the straight-line-following problem, while in the second experiment obstacles was added as well. The reward for reaching the goal state was given as 1. However, since the probability of agent finding the particular point with a reward was zero an additional multivariate Gaussian distribution reward was applied, such that if the vessel position was within a radius b_g of the goal state, a small reward was given which increased the closer the vessel was to the terminal state. An identical reward strategy was applied during collision, but now with a negative reward of -1. However, it has to be noted that the agent does not have any ability to learn which direction to head before it reaches a position where it starts accumulating a reward. Hence, the proposed solution still did not fully address the initial problem. As mentioned in [36]: *it may be extremely difficult to stumble upon rewards by chance and the time to learn a policy to maximize the rewards exponentially increases*, especially for problems with *long trajectories of actions and delayed rewards*. Furthermore, the state-space consist of relative positions between the vessel and goal state and relative position between the vessel and the collision object, as well as the changes in distance (the derivatives). In [37] and [38] a detailed discussion on the importance of normalization for deep learning have been made. At its most fundamental level, training within deep learning can be considered as finding a set of weights. The weights are modified based on the magnitude of each input-state. In other words, larger states have more substantial impact on the weights, than states with smaller values. Since the difference between the initial position and the desired path was smaller in the East-axis compared to the North-axis, the relative distance in North will have much larger impact than the other states. In fact, more than three times larger. Further, studying the results, the vessel seems to be initially steering towards the terminal state before diverging. Thus, it was concluded that the agent have learned to find the target to some degree. However, considering the fact that the initial rudder angle was set to the desired rudder angle, this statement may appear dubious. Because, even if the agent made several random action it may still terminate close to the end-state. In other words, the agent does not truly showcase that it has learned the correct policy. By studying figure 5.2 in the thesis this assumption can be justified, as the output of the agent remains relatively constant even though the vessel was drifting off. This issue was also addressed in [34], where it was solved by relocating the initial position and heading, to verify that the vessel could turn and steer to the path correctly.

Similarly, in [39] DDPG was applied as a guidance system feeding the surge command u_c and heading command ψ_u to the controllers. The model also incorporated rules of COLREG for collision-avoidance situations. It has to be noted that this study is an extension of the work done in [40], now considering collision-avoidance as well. Unlike [35], the initial position was set to a random position with a distance less than 500 meters close to the desired path. Hence, justifying that the agent actually learned a policy by maneuvering correctly back to the desired path. The input states were also normalized accordingly, to avoid biased input-state. Initially, the agent exhibits high level of ability in fulfilling path-following. However, as stated in the thesis, the agent was not able to achieve smooth control inputs. Hence, the resulting inputs revealed a Bang-Bang controller type of behaviour as discussed in [34]. In general, one desires to avoid such behaviour as this generally leads to wear and tear on the actuators. It was pointed out that various reward strategies had been attempted without any success. This issue is referenced to in [41]. The reason behind this problem may be that initially, the reward function is shaped such that one metric is optimized, which is minimizing the distance to terminal state. However, looking at the output of the agent, which is equivalent to the control input, it is clear that higher level of smoothness is required. In other words, the reward function has to be shaped to optimize the output as well, adding another metric. As one continues to develop the model, one may discover additional metrics that needs consideration. Besides, it is highly desired that the agent performs well on all the metrics. Hence, the resulting global reward function has to be designed such that it combines all the sub-goals of the model, which leads to a multi-objective reward function. Developing such function may be challenging in many cases, and the underlying consequences are not always evident at first-sight. In terms of collision-avoidance, the vessel was capable of evading the on-coming vessel successfully. But since the vessel maneuvered on the port-side, it violated the rules of COLREGS, since the expected alteration is on the starboard-side. To force the vessel to take an early substantial action, a teardrop-shaped "safe-zone" was added. The results showcased that the vessel was able to take actions accordingly, but the resulting rudder command still stood out as extremely noisy.

1.2.5 Autonomous vessel: A class society perspective

The previous section displayed some of the weaknesses related to deep learning-based solutions. In general, it is expected that new solutions can demonstrate the ability to safeguard human life, environment and other properties while operating. However, in terms of deep learning-based solutions this is not always possible, as demonstrated by by AlphaGo in 2016 [42]. AlphaGo is a computer program that was trained with Deep Learning to play the board-game Go. It was capable of beating the best human player at the time Lee Sedol, in a five-game match by making actions that were unthinkable even by the experts. Hence, in terms of safety, the possibility of an agent making unexpected action, even if the action is convenient at the time, may not be favourable in terms of safety. Especially, considering the resulting unreliability that the model causes in the domain. Similar issues have been discussed extensively in the medical care where human lives are at stake, such as in [43] and [44]. As a result, the application of deep learning at the medical field have mainly been focused on image processing, where the consequences of the uncertainties related to deep learning-solutions are small.

As mentioned previously, for the maritime community rules and regulations are defined by the International Maritime Organization (IMO), which is an organization under the United Nations. IMO outlines conventions and legal instruments such as the International Convention for the Safety of Life at Sea SOLAS and COLREG, to maintain the safety and security of the shipping industry. In a similar manner, the roles of the class societies such as Den Norske Veritas Germanischer-Lloyd (DNV GL) and Lloyds Register, are to verify that the design, construction and the maintenance of vessels satisfies the legal standards. Each class society have their own set of class rules that covers the technical requirements related to each component on the vessel. At present time, IMO has not yet provided any specific regulations for novel technologies such as the autonomous vessel. However, as stated in [45], national and regulatory bodies can support the implementation of such solutions within their local territorial waters. As a consequence, DNV GL has outlined a guidance for development of autonomous solutions in [45]. Note that Lloyds Register has published a similar guidance in [46], but the content was considered rather narrow. In general, the navigation task can be divided into four sub-tasks, as represented by the blocks in figure 1.2. Each task, can be performed by either a human or a system, or

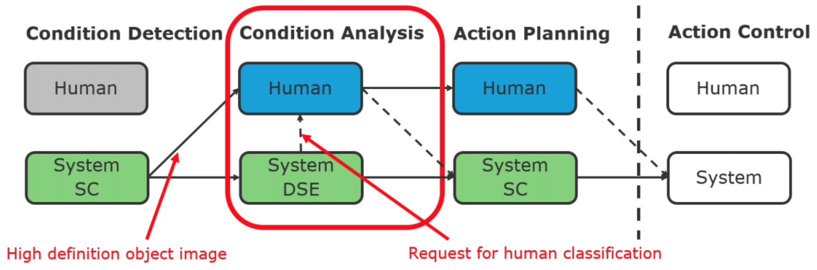

Figure 1.2: The four sub-tasks of navigation

combination of both. In this example, condition system, action planning and action control is made by the system, while condition analysis is partly performed by the human operator. In general, any new solution has to be as good as, or better than the conventional solution, in order to achieve an equivalent or better level of safety. The objective in this thesis is to develop a self-controlling SC action planning (path-planning) system, which have the following requirements stated in the guideline: *Based on the object classification information, the system has capabilities to calculate an updated passage plan in accordance with COLREG that are equivalent or better than that of a navigator on board the vessel.* In addition, it is necessary that the remote operator have a supervising role during navigation. In other words, the operator needs to be provided with sufficient information to be able to derive independent conclusions on the best actions. If the navigation situation becomes too complex for the system to handle, the vessel should be brought to a Minimum Risk Condition (MRC) automatically. MRC is defined as a state that causes least risk to life, environment and property, and a state the vessel should enter when an abnormal situation occurs. In a similar manner, if the operator based on his/her conclusion forecast a hazardous situation, they should have the option to manually intervene and bring the vessel to MRC. In general, it is also expected that the collision avoidance system clearly indicates the updated plan before a control action is made, giving the remote operator enough time to make own analysis and intervene if necessary. The system requirements can therefore be summarized to the following two conditions:

1. The system has to comply with COLREG rules
2. The system has to offer transparency of the planned maneuvering

The previous chapters presented several examples considering COLREG, with varying success. However, in terms of transparency the examples may be considered insufficient. Especially, since the actions are made instantaneous, which provides zero transparency. A final concern to be addressed in terms of deep learning-based system is the explainability of the model. Deep learning have long been criticized for being a "Black Box"-solution, as understanding the process within multiple complex layers is often considered impossible as referenced in [47], [41] and illustrated by AlphaGo.

1.2.6 Digital twin

In a similar manner to machine learning, digital twins are a result of the expansion within computer science [48] [49]. As the term implies, a digital twin is a digital model or software that tries to replicate a process or physical asset [50]. The model is developed based on stored data, algorithms, sensor and other similar resources that offers information on the actual system. Although, the concept has a wide application, it is mostly implemented within the tech industry. Wind farm [51], NASA Air force vehicles [52] and power systems [53] are only some examples of actual possible applications of digital twins. Correspondingly, digital twin also have a significant relevance within maritime industry. For instance, a digital twin of DNV GL's fully autonomous electric vessel ReVolt was designed in [54]. Likewise, the Kongsberg group developed Kognitwin, a digital twin used to replicate physical assets such as oil platforms [55]. Digital twins can also be applied in much smaller scale i.e. marine propulsion system as in [56]. In general, the motivation behind a digital twin, is to be able to better understand, predict and optimize the actual process or system that is being mirrored to increase the performance. For instance in DNV GL's hull condition monitoring, the aim is to forecast possible faults in the hull such that preventive actions can be made [57]. Therefore, a digital model based on wave, position and sensor monitoring is created. In other words, digital twins offer a new and unique tool to both develop, as well as verify real systems more effectively.

Digital twin can also be utilized to generate accurate simulators of real processes for software development. As stated in [41]: *... systems grounded in the physical world ... can range in size from a small drone to a data center, in complexity from a one-dimensional thermostat to a self-driving car, and in cost from a calculator to a spaceship. In all these scenarios there are recurring themes: there is rarely a good simulator, the systems are stochastic and non-stationary, have strong safety constraints, and running them is expensive and/or slow.* Hence, having a simulator that represents the real process as accurate as possible is essential for solving real life-tasks. This is similar to the requirements within control theory. To develop adequate controllers, accurate models of the real process is strictly necessary. In the educational setting Mathworks' Simulink, mentioned previously, have been embraced as a successful tool for building realistic process models. However, in terms of machine learning, it is not until recent that Mathworks have been developing machine learning tools in their own programming language Matlab. Thus, the relating libraries are still in an early stage. The substantial advancements of AI and machine learning libraries are mainly achieved in the programming software Python. But, unlike Matlab, Python does not offer any replacement to Simulink. A resolution to this may therefore be the Functional Mock-up Interface (FMI) standard, which is currently being embraced by the control community [58]. The standard was initially developed to convert and exchange dynamic models, independent of the software it was built in, such that a different software is capable of communicating and running the models. By using the open-source Simulink library FMI Kit, one can convert a Simulink-model to FMI-standard, such that it can be simulated in Python using the open-source library FMPY [59]. This application have two major advantages, the first being rapid execution time. Simulations that may take up to one minute can be executed within few seconds in Python. Secondly, the library allows interference during simulations, such as modification of input values, which Simulink does not allow.

1.3 Objective and scope

The previous sections proves that extensive research on autonomous vessel is currently being made. As a consequence, the purpose of this thesis is to further address the path-planning problem for autonomous vessel. More specifically, developing a self-controlling (autonomous) system for action planning as presented in figure 1.3.

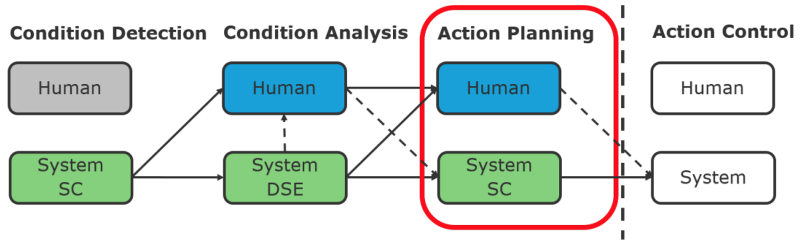

Figure 1.3: Self-controlling path-planning system

In general, the aim of the solution is to consider the following conditions:

1. Develop a self-governing system
2. Generate smooth output to avoid unpredictable behaviour
3. Create transparency of the planned maneuvering
4. Application of rules from COLREG during collision-avoidance

The three first conditions may conflict with the anticipated behaviour of a deep learning-based solution. Although, deep learning have proven to be the most promising solution, it may fall too short in regards to the current rules and regulations seen from class societies perspective. Especially considering the issues presented in the previous sections. Therefore, a supervised method will be embraced in thesis, as the model offers both predictive behaviour, as well as transparency in terms of explainability. To achieve the second point it may be reasonable to build a model that utilizes a traditional controller, and rather focus on the path-planning task. Especially, since control and guidance law have proven to have high-level of performance during maneuvering and path-following in recent times. By choosing a suitable well-tuned controller, we can assure that the behaviour of the system remains satisfying, as long as the path is adequate. In addition, the approach also offers predictability, considering the remote operator is provided sufficient information regarding the path that is being followed. For collision-avoidance a set of rules will be extracted from COLREG and applied on the system to showcase the capability of adapting to the constraints. In general, when developing a system for a vessel, one usually does not have access to the real vessel. However, to train a model, one needs to be able to sample data of the real process. Therefore, the digital twin concept will be utilized to develop a simulator of the actual vessel. The digital twin will be used to train the model, and tested on the real vessel. The motivation behind this is that the simulators can never be fully identical to the reality. Thus, training on a realistic simulator does not guarantee that the model is applicable on the real problem. However, this topic has not been given much importance. Especially, in the experiments reviewed so far, the agents are mainly tested and trained on the same environment. Thus, the simulation does not fully evaluate the generalization capability of the model. Initially, the idea was to showcase the results on DNV GL's model-scaled vessel ReVolt. However, due to COVID19 this had to be cancelled [60]. Instead, Norwegian University of Science and Technology's NTNU simulator of the research vessel R\V Gunnerus, will portray as the real vessel, and the digital twin will be built based on this similar to the approach applied in [34]. Finally, the digital twin will be applied on the real vessel to predict possible collision situations to execute a re-plan step of the actual vessel. Hence, complying with the base requirements of DNV GL's guideline.

In other words, in this thesis a step-wise development of a self-controlling path-planning system will be conducted. More specifically, a supervised machine learning method will be used in combination with Bézier curves to execute route planning. The path will be generated based on a set of constraints that are based on COLREG and constraints related to navigation. Additionally, the digital twin will be developed and applied on the real vessel to append a re-plan step. Further, to carry out the maneuvering, a model based on Roger Skjetne's the Maneuvering problem given in [12] will be implemented. Finally, the model will be evaluated under different scenarios using the real vessel.

1.4 Outline of report

The thesis is made up of five chapters. The first chapter, introduction aims to give the reader an overview of related work, as well as justify the motivation behind the thesis, which is followed by specifications of the objective and scope of the thesis. Chapter 2 presents relevant theories and explain the main concepts related to modelling, maneuvering and path-planning for vessels. This is followed by a detailed description of Bézier curves and machine learning, as well as a brief definition of geometrical concepts used later. In chapter 3, a step-wise development of the final path-planning solution is made, as well as defining the test scenarios. Finally, in chapter 4 the results are discussed, followed by a conclusion in chapter 5.

Chapter 2

Theory

2.1 Mathematical modelling of the vessel

As mentioned in section 1.2.6 a suitable simulator of the real life-problem is necessary to achieve appropriate results. In the following chapters a mathematical definition from control theory for a dynamic vessel is presented.

2.1.1 Process plant model for vessel

To develop an adequate robust controller, good knowledge of the process or system to be controlled is necessary. In control theory this is accomplished by developing mathematical equations describing the process described as a process plant model [2]. The objective of the process plant model is to describe the real process as detailed as possible. In general, process plant model is used to evaluate the performance and robustness of the designed controller. The equations are usually derived from laws of physics which describes the dynamics and motion of the system. For marine vessels, the model is usually separated into a low-frequency model and wave-frequency model, which is summed using superposition to describe the real vessel motion. The low-frequency model considers loads from second-order mean- and slowly varying waves, which are characterized as non-linear. On other hand, the wave-frequency takes motion created by first-order wave in account. The latter is usually presented as a linear model. An universally accepted process model defined by Thor I. Fossen in [61] for motion in 6 degrees-of-freedom is presented below:

$$M_{RB}\dot{\nu} + C_{RB}(\nu)\nu + M_A\dot{\nu}_r + C_A(\nu_r)\nu_r + D_L(\nu_r) + D_{NL}(\nu_r)\nu_r + G\eta = \tau \quad (2.1)$$

$$\dot{\eta} = J_\Theta(\eta)\nu \quad (2.2)$$

- M_{RB}: Rigid-body mass matrix
- C_{RB}: Rigid-body centripetal and Coriolis matrix

- M_A: Added mass matrix
- C_A: Added mass Coriolis and centripetal matrix
- D_L: Linear damping matrix
- D_{NL}: Non-linear damping matrix
- G: Restoring forces and moments vectors
- τ: Generalized thrust and environment force vector

2.1.2 Control plant model for vessel

When developing a controller or observer, a model of the process is usually also incorporated in the design. Generally, the process plant model presented in the previous section may occur overly complicated in most cases. Hence, we often try to establish a more simplified model. In control theory this model is termed as the control plant model. Although the model is characterized as simplified, the control plant model still describes the main dynamics of the process adequately. For a vessel the control plant model usually neglect the non-linear components, which results in the model below:

$$M_{RB}\dot{\nu} + C_{RB}(\nu)\nu + M_A\dot{\nu_r} + D_L(\nu_r) + G\eta = \tau \tag{2.3}$$

$$\dot{\eta} = J_{\Theta}(\eta)\nu \tag{2.4}$$

- M_{RB}: Rigid-body mass matrix
- C_{RB}: Rigid-body centripetal and Coriolis matrix
- M_A: Added mass matrix
- D_L: Linear damping matrix
- G: Restoring forces and moments vectors
- τ: Generalized thrust and environment force vector

2.1.3 Generalized inertia forces

System inertia matrix **M** including added mass is given as:

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & -X_{\dot{w}} & 0 & mz_G - X_{\dot{q}} & 0 \\ 0 & m - Y_{\dot{v}} & 0 & -mz_G - Y_{\dot{p}} & 0 & mx_G - Y_{\dot{r}} \\ -Z_{\dot{u}} & 0 & m - Z_{\dot{w}} & 0 & -mx_G - Z_{\dot{q}} & 0 \\ 0 & -mz_G - K_{\dot{v}} & 0 & I_x - K_{\dot{p}} & 0 & -I_{xz} - K_{\dot{r}} \\ mz_G - M_{\dot{u}} & 0 & -mx_G - M_{\dot{w}} & 0 & I_y - M_{\dot{q}} & 0 \\ 0 & mx_G - N_{\dot{v}} & 0 & -I_{zx} - N_{\dot{p}} & 0 & I_z - N_{\dot{R}} \end{bmatrix} \tag{2.5}$$

The parameter m represents mass of the vessel, while I_x, I_y, I_z are moments of inertia in the directions x, y and z, respectively. $I_{zx} = I_{xz}$ on other hand, is the products of the inertia matrices. $X_{\dot{u}}$, $X_{\dot{w}}$, $X_{\dot{q}}$ and $Y_{\dot{v}}$ symbolize the zero-frequency added mass coefficients.

2.1.4 Generalized Coriolis and centripetal forces

In equation 2.6 the matrix $C_{RB}(\nu)$ represents the skew-symmetric Coriolis and centripetal matrix for the rigid body, while C_A symbolize the Coriolis and centripetal matrix of added mass.

$$C_{RB}(\nu) = \begin{bmatrix} 0 & 0 & 0 & c_{41} & -c_{51} & -c_{61} \\ 0 & 0 & 0 & -c_{42} & c_{52} & -c_{62} \\ 0 & 0 & 0 & -c_{43} & -c_{53} & c_{63} \\ -c_{41} & c_{42} & c_{43} & 0 & -c_{54} & -c_{64} \\ c_{51} & -c_{52} & c_{53} & c_{54} & 0 & -c_{65} \\ c_{61} & c_{62} & -c_{63} & c_{64} & c_{65} & 0 \end{bmatrix} \tag{2.6}$$

$$C_A(\nu_r) = \begin{bmatrix} 0 & 0 & 0 & c_{41} & -c_{51} & -c_{61} \\ 0 & 0 & 0 & -c_{42} & 0 & -c_{62} \\ 0 & 0 & 0 & -c_{43} & -c_{53} & 0 \\ 0 & c_{42} & c_{43} & 0 & -c_{54} & -c_{64} \\ c_{51} & 0 & c_{53} & c_{54} & 0 & -c_{65} \\ c_{61} & c_{62} & 0 & c_{64} & c_{65} & 0 \end{bmatrix} \tag{2.7}$$

2.1.5 Generalized damping forces

In general, we divide the damping forces into a linear and non-linear component. For increasing speed and turbulent flow, the linear damping is almost distinguishable as compared to the contribution from the non-linear damping. Likewise for velocities close to zero, the linear damping becomes more important in accordance with the non-linear damping.

$$D_L = -\begin{bmatrix} X_u & 0 & X_w & 0 & X_q & 0 \\ 0 & Y_v & 0 & Y_p & 0 & Y_r \\ Z_u & 0 & Z_w & 0 & Z_q & 0 \\ 0 & K_v & 0 & K_p & 0 & K_r \\ M_u & 0 & M_w & 0 & M_q & 0 \\ 0 & N_v & 0 & N_p & 0 & N_r \end{bmatrix} \tag{2.8}$$

$$D_{NL}(\nu_r, \gamma_t) = 0.5\rho_w L_{pp} \begin{bmatrix} DC_{cx}(\gamma_r)|U_{cr}|U_{cr} \\ DC_{cy}(\gamma_r)|U_{cr}|U_{cr} \\ BC_{cz}()\gamma_r|w|w \\ B^2C_{c\phi}(\gamma_r)|p|p + z_{py}DC_{cy}(\gamma_t)|U_{cr}|U_{cr} \\ L_{pp}BC_{c\theta}(\gamma_r)|q|q - z_{pz}DC_{cx}(\gamma_t)|U_{cr}|U_{cr} \\ L_{pp}DC_{c\psi}(\gamma_r)|U_{cr}|U_{cr} \end{bmatrix} \tag{2.9}$$

2.1.6 Generalized restoring forces

The generalized restoring matrix **G** consist of the linear gravitation and buoyancy force coefficients. Since a ship in general is symmetric in the xz-plane, we get the following matrix:

$$G = -\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_z & 0 & Z_\theta & 0 \\ 0 & 0 & 0 & K_\phi & 0 & 0 \\ 0 & 0 & M_z & 0 & M_\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.10}$$

2.2 Path following: The Control problem

The universal goal of a vessel is to get from an initial location to a desired location. This is done by sailing or steering the vessel along a desired path. In general, to achieve this, the control problem is separated into two sub-tasks. The first task defined as the *Geometric task* is usually the main task. That is for the output of the system, such as the position of a vessel, to converge to a desired path $y_d(s)$. y_d is a parametric function of the *path variable s*. By using the path variable we can add constraints on the dynamic behaviour of the vessel while following the path. This leads us to the second task, the *Dynamic task*. Usually, the dynamic tasks revolves around controlling the speed or acceleration of the system along the path. In motion control for a vessel one of two strategies are usually applied [11]:

- The tracking control problem: The control objective is to trace a target or point moving through its trajectory. The trajectory describes the motion of the object described mathematically by the geometric of the path or position over time. If y(t) represents the output of a control system, and y_d(t) trajectory, we can define this mathematically as:

$$\lim_{x\to\infty}(y(t) - y_d(t)) = 0 \tag{2.11}$$

- The path following control problem: The control objective is to converge and follow a predefined path which is independent of time, with non-zero motion. Let us define a function d(y;$\mathcal{P}$), where $\mathcal{P}$ is equivalent to the desired path $y_d(s)$, which measures the distance from y to $\mathcal{P}$ for each y $\in R^m$.
  That is for y $\in \mathcal{P}$ d(y;$\mathcal{P}$) = 0 and for y $\notin \mathcal{P}$ d(y;$\mathcal{P}$) > 0. Then we can define the path following problem mathematically as:

$$\lim_{x\to\infty} d(y(t);\mathcal{P}) = 0 \tag{2.12}$$

In the latter one usually sets a desired forward speed, and control the heading to follow the desired path, such that the problem is only focused on solving the geometric path. However, in the former $y_d(t)$ is a time-dependant function. Hence, by differentiating with respect to time one and two times, we can obtain the desired speed and acceleration, respectively. In other words, the tracking problem combines the geometric and dynamic task, because $y_d(t)$ defines the desired position, velocity and acceleration of the vessel. In 2005 Roger Skjetne proposed an alternative method called the Maneuvering Problem [12]. The Maneuvering problem breaks the problem into two sub-tasks, the geometric task and the dynamic task. During path-following the vessel initially attempts to satisfy the dynamic task, such as a speed assignment, while tracing the path. However, if the vessel faces difficulties following the path, it can sacrifice the speed assignment to improve the path-following. A detailed presentation of the Maneuvering problem follows below.

2.2.1 The Maneuvering problem

In general for a system with output y $\in \mathbb{R}^m$, we can define the points in the desired path the set:

$$\mathcal{P} : \{y \in \mathbb{R}^m : \exists s \in \mathbb{R} \; s.t. \; y = y_d(s)\} \tag{2.13}$$

$y_d(s)$ represents the desired path parametrized by the continuous path variable s. Based on this we can establish the maneuvering problem based on the following two tasks as presented in [12]:

1. **Geometric task:** For any continuous function $s(t)$, force the output y to converge to the desired path $y_d(s)$:

$$\lim_{t\to\infty} |(y(t) - y_d(s(t)))| = 0 \tag{2.14}$$

2. **Dynamic task:** Satisfy one or more of the following assignments:

   (a) **Time assignment:** Force the path variable s to converge to a desired time signal $v_t(t)$:

$$\lim_{t\to\infty} |s(t) - v_t(t))| = 0 \tag{2.15}$$

   (b) **Speed assignment:** Force the path speed $\dot{s}$ to converge to a desired speed $v_s(s(t), t)$

$$\lim_{t\to\infty} |(\dot{s}(t) - v_s(s(t), t))| = 0 \tag{2.16}$$

   (c) **Acceleration assignment:** Force the path acceleration $s\ddot{(t)}$ to converge to a desired acceleration $v_a(\dot{s}(t), s(t), t)$

$$\lim_{x\to\infty} |\ddot{s}(t) - v_a(\dot{s}(t), s(t), t)| = 0 \tag{2.17}$$

2.3 Path parametrization

Recall the parametrized desired path $y_d(s)$ used to define the Geometric task in equation 2.14. In this section we will consider details concerning the development of the parametrized path. In general we represent the desired path with one of two types: *straight-line path* or *curved path*. The former is the preferred due to simplicity. Straight-line path-following is usually based on way-point tracking, where way-points are defined along the path, and connected by straight-lines and circle arcs. The circle arcs between two straight-lines describes the desired turning rate to avoid sharp turns. One drawback with this method is the jump in desired yaw-rate r_d during transition from one path to another. This is due to the fact that $r_d = 0$ while on the path, but $r_d =$ constant along the circle arc, thus creating a jump. This can be avoided with interpolated paths, such as a curved path. For curved path-following we define the entire desired path with a geometric curve parametrized by the continuous path variable s. There are numerous ways to design such a curve, but in this thesis we consider the Bézier Curve.

2.3.1 Bézier Curve

A Bézier curve is a parametric curve based on Bernstein polynomials, which is mainly used in Computer Aided Geometric Design [62]. It was originally formulated by Dr. Pierre Bézier during the 1960s for sketching the design of Renault cars. In general, a Bézier curve of n degree consist of *n+1* control points P_0, P_1,..., P_n as observed in figure 2.1. The Bézier curve is designed such that it always passes through the first and the last end-point. In addition, it has the property of being tangential to the control polygon at end-points. The control polygon is the polygon created by connecting the control points in ascending order. The equation for Bézier curve can be derived by considering

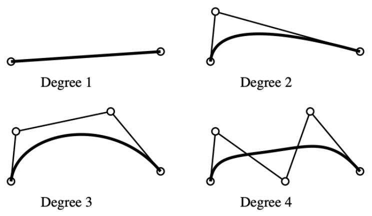

Figure 2.1: Various Bèzier curves and their respective degree n

the equation for center of mass for a set of point masses. Let us imagine four control points P_0, P_1, P_2, P_3 placed as illustrated in figure 2.2. Each point with a point mass m_0, m_1, m_2, m_3, respectively. From elementary physic courses we know that the center of mass defined by point P can be derived as:

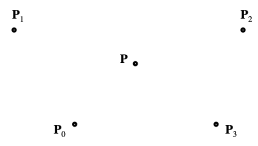

Figure 2.2: The four control points P_0, P_1, P_2, P_3 and the center of mass P

$$P = \frac{m_0 P_0 + m_1 P_1 + m_2 P_2 + m_3 P_3}{m_0 + m_1 + m_2 + m_3} \tag{2.18}$$

Further, let us extend the point masses to be a function of an arbitrary parameter t ∈ [0, 1]. In particular let:

$$m_0(t) = (1-t)^3, \quad m_1 = 3t(1-t)^2 \quad m_2(t) = 3t^2(1-t) \quad m_3 = t^3 \tag{2.19}$$

When the variable t is adjusted, the center of mass is moved accordingly. If we draw all the center of masses for each t, based on the control points in figure 2.2, we get the following curve:

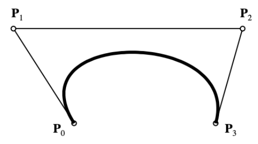

Figure 2.3: The final curve representing center of mass for t ∈ [0, 1]

Since the point of masses are functions of degree three, the resulting Bézier curve becomes a cubic curve. Finally, note that when we sum the point masses we get:

$$(1-t)^3 + 3t(1-t)^2 + 3t^2(1-t) + t^3 = [(1-t)+t]^3 = 1^3 \tag{2.20}$$

Thus equation 2.18 reduces to:

$$P = m_0P_0 + m_1P_1 + m_2P_2 + m_3P_3 \tag{2.21}$$

As previously mentioned, Bézier curve has the property of always passing through the first and last control point. This becomes evident if we look at the plot of the point masses as functions of t: Notice that for t = 0, all the masses are

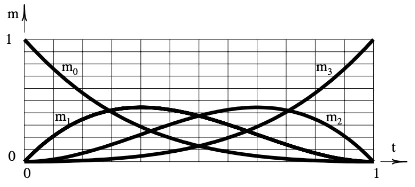

Figure 2.4: The point masses as function of t ∈ [0, 1]

equal to 0 except for $m_0 = 1$. Similarly for t = 1 $m_0 = m_1 = m_2 = 0$, while $m_3 = 1$.

In general, the mass functions $m_i(t)$ are labelled as "blending functions" described mathematically $B_i^n(t)$. The blending function which will vary depending on the type of curve. Particularly for the Bézier curve, the blending function is based on Bernstein polynomials defined as:

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i \quad i = 0, 1, .., n \tag{2.22}$$

Thus we can derive the more general equation of Bézier curve for degree n as:

$$P(t) = \sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^iP_i \tag{2.23}$$

Finally, two additional key properties of the Bèzier curve has to be noted. Let us first consider an arbitrary Bézier curve where a nail is hammered at each control point. Further, imagine that we enclose this area with a rubber-band resulting in the shaded areas seen in the figure 2.5 below. This area is defined

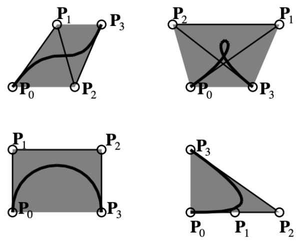

Figure 2.5: The convex hull for various Bézier curves

as the convex hull. This illustrates a key property of Bézier curve, namely that the Bézier curve always lie within the convex hull area as observed in the figure. This becomes apparent if we recall the center of mass example, because the center of mass cannot lie on the outside the convex hull. The second key property is termed as Variation diminishing property. Principally, it states that the actual curve does not "wiggle" more than the actual control polygon. In simple manner, it assures that the curve is smoother than the control polygon.

2.4 Geometrical intersections

In the following two chapters, the mathematical definitions to determine geometrical intersections are derived. The motivation behind this is to be able to discover collisions between objects later in the thesis.

2.4.1 Intersection between two line segments

Consider two line segments p and q given by the coordinates (p_1, p_2) and (q_1, q_2) in the x,y-plane. The aim is to determine if the lines intersect. To achieve this, let us first consider the definition of *orientation* for triplet of points as seen in figure 2.6. If there is a left turn as seen in the far left example, the orientation

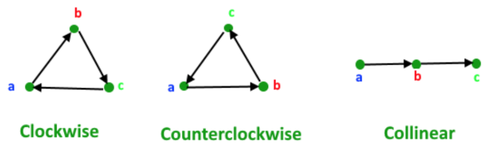

Figure 2.6: Types of orientation. From left: Counterclockwise, clockwise and collinear.

is defined as counterclockwise. In a similar manner, if there is a right turn, the orientation is clockwise. If there is no turn as in the far right case, the orientation is defined as collinear. To determine the orientation mathematically one can compute the slopes between two pairs of the triplet of points and compare the values. For the example in figure 2.7 the slopes are given as:

$$(p_1, p_2) : \sigma = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.24}$$

$$(p_2, p_3) : \tau = \frac{y_3 - y_2}{x_3 - x_2} \tag{2.25}$$

- If $\sigma > \tau$ the orientation is clockwise.
- If $\sigma < \tau$ the orientation is counterclockwise.
- If $\sigma = \tau$ the orientation is collinear.

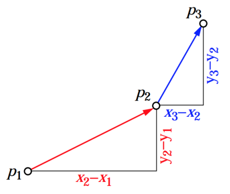

Figure 2.7: The distances in x- and y-direction between triplet of points.

Based on orientation one can verify if two line segments intersect if and only if one of the two conditions are fulfilled [63]:

1. **The general case:**
   - The triplet of points (p_1, p_2, q_1) and (p_1, p_2, q_2) have different orientations
   - The triplet of points (q_1, q_2, p_1) and (q_1, q_2, p_2) have different orientation.
2. **The special case:**
   - The triplet of points (p_1, p_2, q_1), (p_1, p_2, q_2), (q_1, q_2, p_1) and (q_1, q_2, p_2) are all collinear
   - The x-projection of line segment p intersect line segment q
   - The y-projection of line segment p intersect line segment q

For more detailed understanding, it is advised to study appendix A.7 with illustrative examples.

2.4.2 Intersection between a circle and line segment

Consider the problem in figure 2.8. The aim is to determine if line segment AB intersects with the circle with radius r. To do this, one simply can project the vector AC onto line AB, such that the projected vector AD is constructed. From the figure, it is obvious that if CD is less than or equal to the radius of

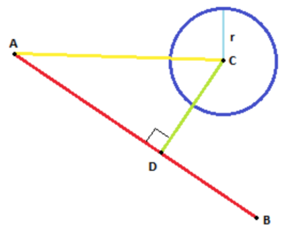

Figure 2.8: Problem formulation of determining the intersection between circle and line segment.

the circle, the line segment is intersecting the circle. Using Pytaghoras theorem CD can be found as:

$$CD = \sqrt{AC^2 - AD^2} \tag{2.26}$$

where criteria for intersection is satisfied for CD $\leq$ r.

2.5 COLREG

To prevent collision in ocean traffic IMO established the convention International Regulations for Preventing Collision at Sea (COLREG), which defines a set of navigation rules to be complied by the vessels and crew travelling on the sea. Therefore, in order to develop adequate path-planning solution, it is important to incorporate the legal rules in the model. However, unpredicted situations may arise, where other vessels are either unable or unwilling to comply to the rules, as already exemplified in [6]. Hence, highlighting the necessity of a human operator in a supervising role. In total COLREG, covers 40 rules and regulations for different scenarios at the sea, as well as requirements on equipment to prevent collision. Obviously, implementing and testing for all the scenarios in one thesis is impracticable. Therefore, only Rule 14 Head-on situation (see figure 2.9 for illustration) is considered, which consist of the following three conditions [64]:

(a) *When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other.*

(b) *Such a situation shall be deemed to exist when a vessel sees the other ahead or nearly ahead and by night she could see the masthead lights of the other in a line or nearly in a line and/or both sidelights and by day she observes the corresponding aspect of the other vessel.*

(c) *When a vessel is in any doubt as to whether such a situation exists she shall assume that it does exist and act accordingly.*

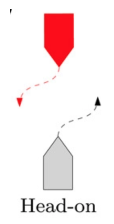

Figure 2.9: Head-on situation as defined by COLREG rule 14.

2.6 Machine learning: Supervised learning

Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed [65]. This is a popular quote presented by Arthur Samuel. In 1959 he developed the Samuel Checkers-playing Program, which was one of the earliest invention demonstrating self-learning capabilities. Hence, he is often considered the godfather of machine learning (occasionally also termed *statistical learning*). Principally, the quote states that a computer program that is programmed such that it is able to learn without interaction from a third-party (i.e human operator), is applying machine learning to some extent. A more formal description is presented in [66]: *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.* At first-sight the definition may seem ambiguous. To clarify consider an example of a computer-program learning to play chess. In this example we would define the task T of the program as playing chess. Further, to measure the performance P of the program we have to identify a good criteria. This can be percentage of games won or some other suitable criterion. Finally, the experience E is gained through playing matches against an opponent. Based on this we can state that the computer-program is learning to play chess, if the percentage of games won is increasing as the program is playing games against opponents.

Machine learning certainly have an advantage compared to a solution based on traditional programming. If we reconsider the previous example and try to implement every move and countermove explicitly, we quickly realize that it becomes an impossible task. Although, the popularity of machine learning-based solution have increased recently, one can find several successful invention existing already. Such as E-mail spam filtering based on keywords, speech recognition software like Alexa and Apple Siri, and Face detection application to unlock phones. In general, we differentiate between three main types of learning within machine learning: Supervised learning, Unsupervised learning and Reinforcement learning. In Supervised learning the computer program is given a set of data containing details about the attributes, also termed features, and an associated target or *prediction* variable. This could for instance be type of dogs, were the features describes their colour, size, age and so on. Based on the features, the program attempts to learn the relation between the features and the prediction variable, which is type of dogs. Later, the algorithm can be applied to predict correct dog-type on new unseen data. This can draw similarity to how humans learn and differentiate objects. Initially, we perceive an object based on its features, and retain information about the definition or "target name" to later identify similar objects. In the case of unsupervised learning, one does not give any prior information about the target variable or characteristics of data. Instead, the algorithm attempts to analyze and detect any underlying structure not apparent to the human eye. An example would be uncovering similarities between patients with a particular decease, such as blood type or age-group. On other hand, in reinforcement learning one applies a reward-based learning. Initially, the program also termed as the *agent* makes an action. Based on the new state the agent attains, it is either rewarded or penalized. Thus, implicitly approving correct actions, in a similar manner to learning by trial and error. Reinforcement learning have a wide application within self-learning programs and robots like Google DeepMind's famous agents who learned to play the Atari games. Additionally, combination and variations of these three also exists, such as semi-supervised learning which combines unsupervised learning and supervised learning. In this chapter a more detailed explanation on supervised learning and types of algorithms will be given.

2.6.1 Supervised Learning

The main objective of supervised learning is to make accurate predictions for new data. This is done through identifying how the features influence the output, which is as mentioned the prediction variable. The name supervised originates from the fact that the target variable "supervises" the model's analysis in order to predict the correct target. In general, one distinguishes between two types of prediction model, classification and regression. In the former, the target variable is a qualitative variable such as dog, cat, fish or group number 1,2,... Hence, the model tries to predict the class membership. On other hand, in regresseion the target variable is a quantitative value part of a continuous set, which means it potentially can take any number. Building a machine learning model can be divided into three steps:

1. **Data collection**: This is the initial step where the aim is to collect any relevant data from databases, file systems and other resources.

2. **Data preparation**: As a consequence of computer- and information technology development, massive quantity of data is available only few keystrokes away. However, in most cases the data is stored in a useless state. Additionally, only a small amount of the data is considered valuable. Thus, data preparation is required, which involves cleansing, manipulating and assembling the collected data to an applicable state. This step is also often called *feature engineering*, because it involves selecting the features to be used in the model. Therefore, it is considered as the most important part of the process, because selecting the correct data decides the final performance of the model.

3. **Model selection**: In 1997 David Wolpert and William Macready stated the *No free lunch-theorem* (NFL) [67]. The NFL states that there is no one universal model or algorithm that can be applied for all problems. In other words, a model that performs well in one problem, may be unsuitable in a different domain. Hence, in model selection one usually have to evaluate and compare different models to uncover the most suitable model. Usually one selects a set of models that are fitted or *trained* using the training

data extracted from previous step. Each model has its own set of hyper-parameters that is tuned during training, based on a common criteria across the models. Model selection often involves many repeating loops of tuning and evaluation of the models before one is finally selected.

4. **Prediction**: The final step is to implement the model to predict on the real problem.

The previous process can also be termed in a mathematical way. Assume we observe a quantitative value Y, and p number of features. X_1, X_2,... , X_p. Suppose there is a relation between Y and X = (X_1, X_2, ..., X_p). Then supervised learning says that we can develop the following general relationship:

$$Y = f(X) + \epsilon \tag{2.27}$$

f is a fixed unknown function of the features X, while ϵ is a random error term independent of X and zero mean. We say that f provides the systematic information that the features X provides about the target Y. To clarify the notations, let us consider an example with only one feature to simplify the illustration. By studying the left graph of figure 2.10 we can conclude that based

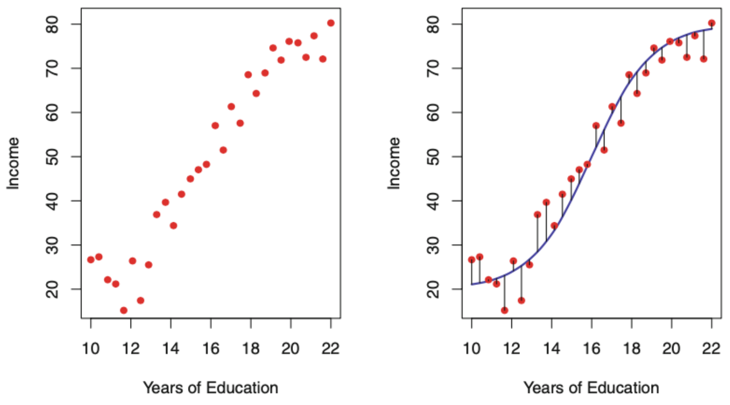

Figure 2.10: Left graph: Simulated data-sample of points used to determine f(x). Right graph: The true form of f(x). The black lines represents the random error ϵ

on the feature *Years of education* there may be some function that gives us the *Income*. But let us presume f is known for this example as it is a simulated example, resulting in the right graph in figure 2.10. The black vertical line represents the error term ϵ. Looking at the distribution of data-points above and below the function f, it is sensible that the total error has zero mean. However, generally the function f is unknown, and we therefore have to try to estimate the function based on the set of points. Hence, supervised learning outlines the possible approaches to estimate this function. If we define the estimate function f as $\hat{f}$ and the corresponding prediction of target variable Y as $\hat{Y}$, supervised learning gives us the following relationship:

$$\hat{Y} = \hat{f}(X) \approx Y \tag{2.28}$$

In general, the form of $\hat{f}$ can be characterized as either non-parametric or parametric. A parametric approach consist of two steps:

1. Depending on the chosen approach, one establish a mathematical equation with parameters. For linear regression, well-known from elementary math courses, with p numbers of features and ϵ as defined previously, the following model can be derived:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon \tag{2.29}$$

2. The problem now reduces to estimating the coefficients β_0, β_1, ..., β_p. This step is termed as *fitting* or *training* the model. For linear regression this is usually achieved by applying least squares.

However, for non-parametric approach we do not make any assumption of the form of function f. One such approach is K-nearest neighbor regression. In K-nearest neighbor the whole training set is kept within the model. During the prediction step, the given features are used to identify the K-nearest data-points using for example euclidean distance. The predicted value would then be the weighted average of the K-points' target variables. In this case, the variable K is a hyperparameter that has to be optimized. Generally, non-parametric and parametric both have advantages and disadvantages. A parametric model is usually simple to understand, requires little training data and computationally cheap as there are fewer parameters to be estimated. But since we have defined the form of function f, the model is constrained. For example, if we choose linear regression, we assume the relationship between the features and target variable are linear. Hence, if the true function is non-linear the model will underfit the true function, thus having the ability to never give an accurate prediction. This can be solved by choosing a more flexible model. However, choosing a flexible model has the drawback of increased number of parameters needed to be tuned. Additionally, if we choose an overly complex model, it may overfit the data. Principally, overfitting implies that the model is following each data-point too closely. Since noise naturally occurs in a data-set, overfitting may lead to ambiguity, especially when predicting on new unseen data.

An illustrative overview of these concepts can be studied in figure 2.11. On other hand, because non-parametric models do not make any assumptions, they can fit large numbers of functional forms. Thus, the model is more likely to be close to the true function f. But similar to choosing a more flexible parametric model, there is still a possibility of overfitting the model. In general, non-parametric model requires more data to estimate $\hat{f}$ because of the complexity of the model in addition to having more parameters to be estimated.

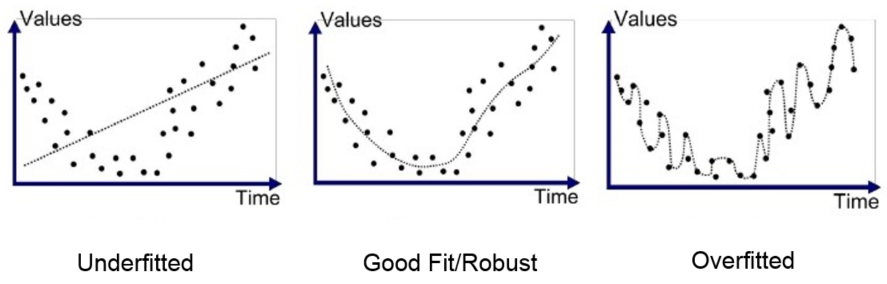

Figure 2.11: Illustration of underfitted model versus overfitted model

The evaluation of the performance of the model consist of three steps:

- Initially, the dataset is split into a training set and test set.
- The training set is used to fit the chosen model.
- Finally, the model is applied on the test set to measure the performance.

In general, this is done by studying the mean-squared error (MSE) of the model, given by the following mathematical equation:

$$MSE = \frac{1}{n}\sum_{n=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{2.30}$$

Here y_i represents the true target value and $\hat{f}(x_i)$ the predicted target value for the ith observation. When the prediction value $\hat{f}(x_i)$ is equal to the true value y_i, MSE is 0. Thus, the aim is to minimize the MSE, either by tuning the parameters or choosing a different model.

Consider the measured noisy data-points presented in the left graph of figure 2.12. The black line represents the true model f, while the yellow, blue and green line represents estimation of model f. If we compare the MSE with increased flexibility as in the right graph, we see that initially the MSE decreases. However, at some point the MSE starts increasing again. This implies that there

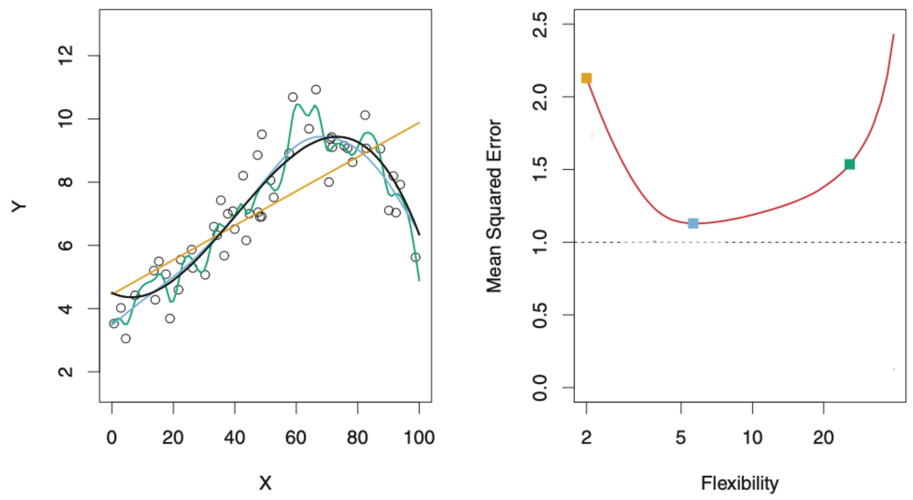

Figure 2.12: Left graph: The circular points represents the measured noisy data-points, while the black line represents the true model f. Yellow line is a linear regression estimation, while blue and green is a more flexible model based on smoothing splines. Right graph: The MSE plotted against increased flexibility.

is an additional property impacting the MSE. If we derive the equation of the expected value of MSE squared $\mathrm{E}(y_0 - \hat{f}(x_0))^2$ it can be shown that it consists of three terms:

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))] + Var(\epsilon). \tag{2.31}$$

The last term $\mathrm{Var}(\epsilon)$ represents the irreducible error, and will always be present unless we have measurements without noise. Note that since the first and second term always will be non-negative, the total MSE cannot be less than the irreducible error independent of the model selection. The first term $\mathrm{Var}(\hat{f}(x_0))$ represents the variance of our model. Choosing a more flexible model will generally result in higher variance. In other words, if we change the training set slightly, the fitted model will have large changes, resulting an overfitted model. The $\mathrm{Bias}(\hat{f}(x_0))$-term represents how much the predicted values differ to the true values. For instance, if we choose a linear model to predict on a non-linear problem, the generated predictions will have a considerable error. The bias term will therefore lead to a substantial impact on the MSE. Hence, to reduce the total MSE we have to minimize the variance and bias of the model. To reduce the bias, a more flexible model has to be chosen. However, this will increase the variance of the model, and at some point the variance will overtake the bias. This trade-off is defined as the *Bias-variance trade-off*, and illustrates that fitting a perfect model is theoretically impossible as seen in figure 2.13.

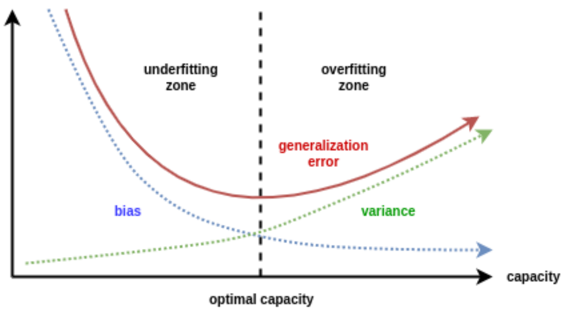

Figure 2.13: The blue line represents the bias, the green line represents the variance and the red line represents the total error.

During the development of the solution several regression models were considered such as different types of Decision trees, Support Vector machine, K-mean regression, Lasso regression and Linear regression. Due to the resulting performance, only the three best performing models are presented, which are all by coincidence a branch of the Decision tree-models. This is due to limitation on the length of the thesis. However, for the curious reader, [68] is recommended for further reading on the latter models.

2.6.2 Decision tree

In contrast to linear regression, decision tree performs efficiently in non-linear problems. The idea behind decision tree is best explained with an example. Consider a problem where the aim is to predict the salary of base ball players, based on the features *numbers of hit* and *experience in years*. An artificial training set is presented in figure 2.14, where salaries are presented in a heat map. Low salaries are given in blue and green, while high salaries are given

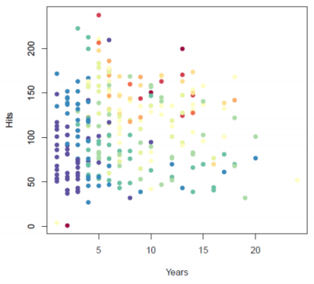

Figure 2.14: Heat map of salaries based on experience in years and numbers of hit.

in yellow and red. Notice that the salaries are given in a log-scale. The main objective of decision regression tree is to divide the area in smaller rectangles R_1, R_2,..., R_J, such that observations that fall within the same area are given the same prediction, that is the average of all the training observation in the rectangle. The resulting model after application of decision tree can be studied in figure 2.15. In other words, for combinations of the features *numbers of*

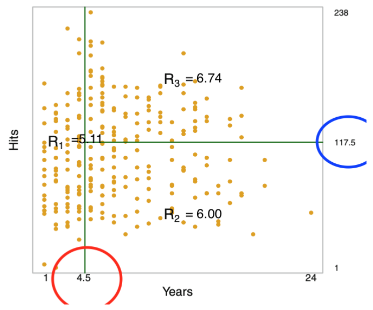

Figure 2.15: The resulting rectangular split applying decision tree regression of the training data.

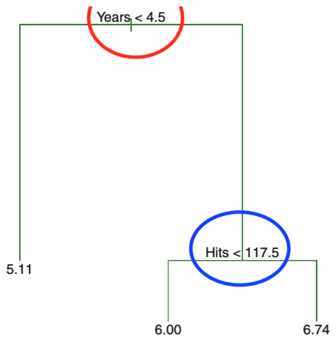

Figure 2.16: The resulting tree visualization of the rectangular split.

hit and *experience in years* that fall into a region R_i, will all have the same prediction. The name decision tree arise from the fact that the resulting model can be visualized using an upside down tree consisting of branches and leaves connected by nodes. The resulting tree for this example is presented in figure 2.16. Notice that the bottom values (leave nodes) are equivalent to the average value of the rectangles R_1, R_2 and R_3 in figure 2.15. At each node, a test is applied on one of the features, and the test generates the rectangular areas. The test and equivalent rectangle boundary is indicated by the circles. Based on the outcome of the test, we either move down the left or right branch to the next node. This process is repeated until a leave node is reached, which gives the final prediction. For instance, let us consider a baseball player with 110 hits, and 6 years experience. Applying the model, at the first node, since the player have more than 4.5 years experience, we would move down the right branch. Similarly, the natural choice would be to move down the left branch at second node, since the player has less than 117.5 hits. Thus, the estimated salary would be the inverse log_{10} of 6, which is 1 000 000. The general process can be summarized to the following two steps:

1. Based on a training set, the feature space, which is the set of all possible values of X_1, X_2,..., X_p, is divided into non-overlapping regions R_1, R_2,..., R_J similar to figure 2.15.

2. All observations that falls into a region R_j are given the same prediction, that is the mean target value of all the training observation in region R_j.

Although, the region in theory can take any shape, high-dimensional rectangles are chosen for simplicity of the predictive model as rectangles are easier to interpret. To determine the rectangles R_1, R_2,..., R_J, decision trees tries to minimizes the residual sums of square (RSS) given by:

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{2.32}$$

where $\hat{yRj}$ represents the mean response of the training observations in R_j. Note that $\hat{yRj}$ will be equivalent to the estimated value of any observation that falls into the region. However, since every partition of the feature space has to be considered, the resulting problem may be computationally infeasible depending on the complexity of the problem. As a consequence, a top-down greedy approach is applied known as recursive binary split. The method is top-down, because it starts in the top node when all the training observation belongs to one region. When a split in predictor space is made, two new branches are created. The approach is termed greedy because at each node, the best step at the current state is made instead of looking ahead and picking a split that would potentially lead to the overall best tree. To make a binary split, a predictor X_j and cutpoint s is first selected, such that the split results in the regions {X | X_j < s} and {X | $X_j \geq$ s} that leads to greatest reduction of RSS. This test is made for all the predictors and all possible values of cutpoint s, until the best is found. Mathematically this is given as:

$$R1(j,s) = \{X|X_j < s\} \text{ and } R_2(j,s) = \{X|X_j \geq s\} \tag{2.33}$$

where the goal is to find j and s that minimizes:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{2.34}$$

The parameter $\hat{y}_{R_j}$ represents the the average of the training observation in R_j, which again is equivalent to the prediction of any observation that falls into the region. After a region is split into two, we continue and split one of the two regions into two smaller regions, and repeat this process until a stopping criterion is met, for instance until no region contains than ten training observation. The hyperparemeters to be tuned is the size of the tree, which is implicitly decided by choosing a stopping criterion. In general, fitting a too large tree may lead to overfitting. Similarly, fitting a too small tree may lead underfitting. Therefore, the size of the tree has to be considered as a bias-variance trade-off problem.

Because of the resulting tree structure illustrated in figure 2.16, regression trees are generally easy to interpret and visualize even for someone with limited domain knowledge. Especially, since it mirrors human decision-making process. However, as the trees grow larger, they become harder to interpret. In addition, decision trees are usually exposed to high variance, and small changes in the data may lead to large changes on the final tree. Hence, random forest and gradient boosting are generally stronger extensions of decision trees, as they address these drawbacks.

2.6.3 Random forest

As mentioned, decision trees suffer from high variance, which means a small change in the training set may lead to big change in the final tree. One approach to reduce the variance, is to apply bootstrap aggregation, also known as bagging. Consider B independent identically distributed observations of a random variable X with same mean and variance σ^2. The mean can be found computationally as:

$$\bar{X} = \frac{1}{B}\sum_{b=1}^{B} X_b \tag{2.35}$$

Further, the variance of the mean can be found as:

$$Var(\bar{X}) = Var(\frac{1}{B}\sum_{b=1}^{B} X_b) = \frac{1}{B^2}\sum_{b=1}^{B} Var(X_b) = \frac{\sigma^2}{B} \tag{2.36}$$

This implies that by averaging, the variance of the sample reduces. The idea behind bagging (bootstrap aggregation) is to draw, with replacement, B numbers of smaller training samples from the original training set, assuming each sample point has the probability $\frac{1}{n}$ of being drawn, and build a separate tree using each of the training sample. This results in $\hat{f}^1(x)$, $\hat{f}^2(x)$,...,$\hat{f}^B(x)$, prediction models. To make a prediction, we average over all the available B models, which results in a prediction with lower variance. Mathematically said, for each bootstrap sample b = 1, 2,..., B a decision tree $\hat{f}^{*b}(x)$ is trained. To make a prediction we take the average of the prediction of all trees such that:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x) \tag{2.37}$$

However, if the data set contains one strong feature, the decision trees produced from each of the bootstrap sample may become too similar. This is due to the fact that most of the trees will use the same strong feature, resulting in a highly correlated prediction. Therefore, an approach that decorrelates the trees is necessary. This is achieved by using same strategy as above using the bootstrapped sample, but instead when a split is made, the model is only allowed to consider m features of all the p features available where m < p. This method is called random forest, and forces the trees to also consider other weaker predictors, resulting in a less correlated prediction. The choice of m depends on the strength of the predictors; The stronger the predictors, the smaller the selection of m. However, in general, m is chosen as $\frac{p}{3}$. Note that for m = p the model becomes bagging.

2.6.4 Gradient Boosting Regressor

Recall that in bagging, we fitted B prediction models, each on a training sample drawn from the original set, and averaged over all models to create the final prediction. Boosting is a similar method, however each model is built sequentially, by using information from the previous tree. Unlike bagging, boosting does not use bootstrapped samples. Instead, a decision tree $\hat{f}(x)$ with d splits is built, which results in a tree with d+1 terminal nodes. The prediction model is then fitted to the residuals of the model resulting in a new tree $\hat{f}^c$. This three is then added to the original tree, but with a weight such that the new prediction model is:

$$\hat{f}(x) = \hat{f}(x) + \lambda\hat{f}^c(x) \tag{2.38}$$

This is repeated until C numbers of trees are made. This approach is termed as slow learning, because the model tends to slowly improve in its weak areas by fitting multiple trees on the residuals. In general, models that learn slowly tends to perform very well. Since the aim of the model is to find the lowest MSE, a gradient descent-approach is applied [69]. The main idea behind gradient descent is to move in the direction with the steepest change in the error, similar to downhill climbing. In general, the following hyperparameters have to be considered for Gradient Boosting Regressor:

- Number of trees C: In general, too big C would lead to overfitting, while too small C means not much information is used.
- Learning rate λ: This parameter decides how fast the model learns, by deciding how much each new tree contributes to the total model. Choosing

a small value, will result in slower learning. In general, smaller learning rate would require larger number of trees C, which therefore is a trade-off problem.

- Numbers of split d: This parameter decides the complexity of the tree. Usually, trees with small d results well, such as d = 1 which result in a stump.

Chapter 3

Design and implementation

Based on the discussion in the literature review, it is evident that deep learning-based solutions do not thoroughly satisfy the expected requirements of the class society. Especially, in terms of safety and providing sufficient transparency, while executing the task. At the same time, research on supervised methods for path-planning have been limited. Therefore, in the following section a step-wise description of a self-controlling path-planning algorithm is developed. The proposed solution will generate a parametrized Bézier curve from an initial state to a terminal state. The generated curve will then be utilized by a maneuvering model based on Roger Skjetne's the Maneuvering problem, to reach the terminal state. Further, to comply with navigation rules, a score paradigm will be developed to stimulate the algorithm to choose suitable paths in terms of collision-avoidance, rules from COLREG and distance travelled. The supervised method to be used in the algorithm is selected based on benchmarking. In addition, the simplified model of the real vessel will be added as a digital twin on the real vessel. The aim is to forecast possible collision to initiate re-pathing. Finally, a set of test scenarios are established to evaluate the model.

3.1 Simulator

In this section a detailed presentation of the test- and verification simulator is presented. The simulator is developed and run in a Python environment. Even though the sea can be considered infinitely large, modelling such a large environment may be computationally expensive. As a consequence, the environment is bounded in $x_{boundary} \in$ [0, 4000] and $y_{boundary} \in$ [-400, 400]. The main motivation behind choosing such a rectangular environment, is simply to easier experiment on head-on situations. It has to also be emphasized that the environment does not model environment forces.

3.1.1 Vessel model

In general, when developing an algorithm, one does not have access to the real vessel. Additionally, the information on the real physical unit is usually constrained. Hence, to portray these limitations, two vessel models were developed in Simulink. The design of the vessel models are based on NTNU's research vessel R/V Gunnerus, of which the main dimensions can be found in table 3.1 below. The model of the real vessel is developed at Norwegian University of Science and Technology (NTNU) by the Department of Marine Technology, and is based on the process plant model of a vessel presented in section 2.1.1. The second model is based on the control plant model for a vessel with 3-degrees of freedom. Recall that the control plant model only contains the most essential dynamics of the actual process. Therefore, it will be used as a digital twin of the real vessel, utilized for training of the proposed solution, and integrated in the real vessel. The numerical matrices of the real vessel and digital twin can be found in appendix A.1 and A.2, respectively. Both models were developed in Simulink and was transfered to Python using Functional Mock-up Unit-format (FMU). This was achieved by using the open-source library FMI Kit to convert the Simulink models to FMU-format, and then running the model in Python using FMPY [70].

Parameter	**Value [unit]**
LENGTH OVER ALL (LOA)	31.25 [m]
LENGTH BETWEEN PERPENDICULAR (LPP)	28.90 [m]
BREADTH MIDDLE (BM)	9.60 [m]
BREADTH EXTREME	9.90 [m]
DEPTH	2.79 [m]
DEAD WEIGHT	107 [tonn]
MAIN ELECTRIC PROPULSION	2 X 500 [kW]
MAIN GENERATORS	3 X 450 [kW]
BOW TUNNEL THRUSTER	1 X 200 [kW]
SPEED AT 100% MAXIMUM CONTINUOUS RATING	28.90 [m]
CRUISING SPEED	4.8 [m/s]

Table 3.1: Main dimensions of R\V Gunnerus

3.1.2 The Maneuvering model

To steer the vessel along the path from A to B, a controller or maneuvering model is necessary. Hence, a maneuvering model was developed based entirely on Roger Skjetne's the Maneuvering problem as defined in section 2.2.1 combined with work done in the course Marine Control Systems 2 at NTNU. In similar fashion to the vessel model, the control design was developed in Simulink.

Recollect that a parametrization of the desired path η_d is necessary. Hence, a path parametrization was developed using a cubic Bézier curve. In general, increasing the degree-of-freedom allows a more flexible path-making which may be ideal, especially in situations with multiple obstacles. However, in the thesis the research is limited to scenarios with one obstacle present. The explicit form of the curve is given mathematically as:

$$\mathbf{B}(s) = (1-s)^3\mathbf{P}_0 + 3(1-s)^2t\mathbf{P}_1 + 3(1-s)t^2\mathbf{P}_2 + s^3\mathbf{P}_3 \ , 0 \le s \le 1. \tag{3.1}$$

P_0, P_2, ..., P_n defines the n+1 control points for a Bézier curve that has n-degrees of freedom. For a cubic curve (n = 3), the four control points in the x, y-plane is made up of the initial position, the final-position, and two points in-between. It can be assumed that the start- and end-position is known, as they are defined by the specific mission. In other words, the remaining task is to determine the two control points in-between. In general, the control points in-between define the curvature and shape of the path as seen in figure 2.5. The aim is to choose the control points such that the vessel is capable of avoiding the obstacles. However, recall that for a vessel the minimum turning rate has to be considered to avoid sharp turns. By deriving the equation of the curvature κ for a curve as:

$$\kappa(s) = \frac{\dot{x}(s)y(s) - \ddot{x}(s)\ddot{y}(s)^{\frac{3}{2}}}{\dot{x}(s)^2 + \dot{y}(s)^2} \tag{3.2}$$

One can establish the following relationship between the curvature and the minimum turning radius R_{min}

$$|\kappa(s)| < \kappa_{max} = 1/R_{min} \tag{3.3}$$

The parameter κ_{max} defines the maximum curvature.

As previously mentioned, the maneuvering problem consist of the following two tasks:

$$\textbf{Geometric task:} \lim_{t\to\infty} |(\eta(t) - \eta_d(s(t)))| = 0 \tag{3.4}$$

$$\textbf{Dynamic task:} \lim_{t\to\infty} |(\dot{s}(t) - v_s(s(t),t))| = 0 \tag{3.5}$$

Here, the dynamic task represents the speed assignment defined in section 2.2.1. From equation 3.1 for cubic Bézier curve we can derive the desired path η_d as:

$$\eta_d(s) = \begin{bmatrix} x_d \\ y_d \\ \psi_d \end{bmatrix} = \begin{bmatrix} (1-s)^3\mathbf{x}_0 + 3(1-s)^2s\mathbf{x}_1 + 3(1-s)s^2\mathbf{x}_2 + s^3\mathbf{x}_3 \\ (1-s)^3\mathbf{y}_0 + 3(1-s)^2s\mathbf{y}_1 + 3(1-s)s^2\mathbf{y}_2 + s^3\mathbf{y}_3 \\ atan2(\frac{\partial y_d}{\partial s}, \frac{\partial x_d}{\partial s}) \end{bmatrix} \tag{3.6}$$

Similarly, differentiating the desired path with respect to s once and twice gives:

$$\frac{\partial \eta_d}{\partial s} = \begin{bmatrix} \frac{\partial x_d}{\partial s} \\ \frac{\partial y_d}{\partial s} \\ \frac{\partial \psi_d}{\partial s} \end{bmatrix} = \begin{bmatrix} 3(1-s)^2(\mathbf{x}_1 - \mathbf{x}_0) + 6(1-s)s(\mathbf{x}_2 - \mathbf{x}_1) + 3s^2(\mathbf{x}_3 - \mathbf{x}_2) \\ 3(1-s)^2(\mathbf{y}_1 - \mathbf{y}_0) + 6(1-s)s(\mathbf{y}_2 - \mathbf{y}_1) + 3s^2(\mathbf{y}_3 - \mathbf{y}_2) \\ \frac{\frac{\partial x_d}{\partial s}\frac{\partial^2 y_d}{\partial s^2} - \frac{\partial y_d}{\partial s}\frac{\partial^2 x_d}{\partial s^2}}{(\frac{\partial^2 x_d}{\partial s^2})^2 + (\frac{\partial^2 y_d}{\partial s^2})^2} \end{bmatrix} \tag{3.7}$$

$$\frac{\partial^2 \eta}{\partial s^2} = \begin{bmatrix} \frac{\partial^2 x_d}{\partial s^2} \\ \frac{\partial^2 y_d}{\partial s^2} \\ \frac{\partial^2 \psi_d}{\partial s^2} \end{bmatrix} \begin{bmatrix} 6(1-s)(\mathbf{x}_2 - 2\mathbf{x}_1 + \mathbf{x}_0) + 6s(\mathbf{x}_3 - 2\mathbf{x}_2 + \mathbf{x}_1) \\ 6(1-s)(\mathbf{x}_2 - 2\mathbf{x}_1 + \mathbf{x}_0) + 6s(\mathbf{x}_3 - 2\mathbf{x}_2 + \mathbf{x}_1) \\ \frac{((\frac{\partial x_d}{\partial s})^2 + (\frac{\partial y_d}{\partial s})^2)(\frac{\partial x_d}{\partial s}\frac{\partial^3 y_d}{\partial s^3} - \frac{\partial y_d}{\partial s}\frac{\partial^3 x_d}{\partial s^3}) + 2(\frac{\partial y_d}{\partial s}\frac{\partial^2 x_d}{\partial s^2} - \frac{\partial x_d}{\partial s}\frac{\partial^2 y_d}{\partial s^2})(\frac{\partial x_d}{\partial s}\frac{\partial^2 x_d}{\partial s^2} + \frac{\partial y_d}{\partial s}\frac{\partial^2 y_d}{\partial s^2})}{((\frac{\partial x_d}{\partial s})^2 + (\frac{\partial y_d}{\partial s})^2)^2} \end{bmatrix} \tag{3.8}$$

where

$$\begin{bmatrix} \frac{\partial^3 x_d}{\partial s^3} \\ \frac{\partial^3 y_d}{\partial s^3} \end{bmatrix} = \begin{bmatrix} -6x_0 + 18x_1 - 18x_2 + 6x_3 \\ -6y_0 + 18y_1 - 18y_2 + 6y_3 \end{bmatrix} \tag{3.9}$$

The speed assignment U_s in the dynamic task is given as:

$$U_s = \frac{U_{ref}}{\sqrt{(\frac{\partial x}{\partial s})^2 + (\frac{\partial y}{\partial s})^2}} \tag{3.10}$$

U_{ref} defines the reference speed of the vessel. For R\V Gunnerus the reference velocity is rounded up to 5 [m/s]. With Lyapunov analysis the maneuvering problem for the vessel can be solved using a backstepping controller. Consider the following Control Lyapunov function (CLF):

$$V_1(\eta, s) = \frac{1}{2} z_1^\top z_1 \tag{3.11}$$

where

$$z_1 = R(\psi)^\top (\eta - \eta_d) \tag{3.12}$$

The time differentiation of CLF is then given as:

$$\dot{V}_1 = z_1^\top (\nu - R(\psi) \frac{\partial \eta_d(s)}{\partial s} \dot{s}) \tag{3.13}$$

Now choosing $\nu = \alpha_1$ as the control law, where

$$\alpha_1(\eta, s, t) = -K_p z_1 + R(\psi)^\top \frac{\partial \eta_d(s)}{\partial s} U_s(s, t), \tag{3.14}$$

$$K_p = K_p^\top > 0 \tag{3.15}$$

and deriving the relationship:

$$\frac{\partial V_1}{\partial s} = -\frac{\partial \eta_d}{\partial s}^\top (\eta - \eta_d(s)) \tag{3.16}$$

equation 3.13 can be rewritten to:

$$\dot{V}_1 \leq -\lambda_{min} K_p \left| z_1 \right|^2 - \frac{\partial V_1}{\partial s} (U_s(s, t) - \dot{s}) \tag{3.17}$$

In other words, if the dynamic task is solved, the choice of α_1 is stable.

For the vessel to follow the path $\eta_d(s)$, an update law for the path variable has to be specified. Introducing the tracking update law [12] $\dot{s}$ can be chosen as $\dot{s} = U_s(s,t)$. Notice that this solves the dynamic task. Hence, the right side of equation 3.17 cancels out and $\dot{V}_1$ is semi-definite. Recall that z_1 is the tracking-error of the desired path. Since $\dot{V}_1$ is only dependant on z_1 it is called the tracking update law.

The update law can be further modified by using the gradient of the CLF with respect to path variable s, and dividing with the normalizing modification $\left|\frac{\partial \eta_d(s)}{\partial s}\right|$, which results in:

$$\dot{s} = U_s(s,t) - \frac{\mu \frac{\partial V_1}{\partial s}}{\left|\frac{\partial \eta_d(s)}{\partial s}\right|} \tag{3.18}$$

This is called the modified gradient update law, and reshapes equation 3.17 to:

$$\dot{V}_1 \leq -\lambda_{min} K_p \left|z_1\right|^2 - \frac{\mu \frac{\partial V_1^2}{\partial s}}{\left|\frac{\partial \eta_d(s)}{\partial s}\right|} \tag{3.19}$$

To make $\dot{V}_1$ semi-definite, μ has to be greater than or equal to 0, which solves the geometric task. However, the dynamic task is reduced to $\lim_{t\to\infty} = \left| - \frac{\mu \frac{\partial V_1}{\partial s}}{\left|\frac{\partial \eta_d(s)}{\partial s}\right|} \right|$. By recollecting that $\frac{\partial V_1}{\partial s} = 0$ when the geometric task is fulfilled, it is obvious that the dynamic task is fulfilled as well. Using the maneuvering control law from equation 3.13 combined with a backstepping control law, a control output τ can be developed which solves the maneuvering problem for a vessel. By introducing $z_2 = \nu - \alpha_1(\eta, s, t)$, equation 3.13 can be rewritten to:

$$\dot{V}_1 = z_1^\top (z_2 + \alpha_1 - R(\psi)\frac{\partial \eta_d(s)}{\partial s}\dot{s}) \tag{3.20}$$

Further, consider the new modified CLF:

$$V_2 = V_1 + \frac{1}{2} z_2^\top M z_2 \tag{3.21}$$

Differentiating V_2 with respect to time gives:

$$\dot{V}_2 = -z_1^\top K_p z_1 + z_2^\top (z_1 - C(\nu)\nu - D(\nu)\nu + \tau - M\alpha_1) \tag{3.22}$$

Observe that by choosing the control output of the vessel τ as:

$$\tau = C(\nu)\nu + D(\nu)\nu - z_1 + M\dot{\alpha}_1 - K_d z_2 \tag{3.23}$$

where

$$\dot{\alpha_1} = -K_p(-S(r)R(\psi)^\top(\eta - \eta_d(s)) + R(\psi)^\top(\dot{\eta} - \frac{\partial \eta_d(s)}{\partial s}\dot{s})) \tag{3.24}$$

$$-S(r)R(\psi)^\top \frac{\partial \eta_d(s)}{\partial s} U_s(s,t) + R(\psi)^\top(\frac{\partial^2 \eta_d(s)}{\partial s^2}\dot{s}U_s + \frac{\partial \eta_d(s)}{\partial s}\dot{U}_s) \tag{3.25}$$

Hence, $\dot{V_2}$ becomes negative definite:

$$\dot{V_2} = -z_1^\top K_p z_1 - z_2^\top K_d z_2 \tag{3.26}$$

Which is equivalent to uniformly globally exponentially stability (UGES). The parameters K_p, K_d and μ had to be manually tuned, which resulted in the values found in table A.1 in appendix A.3.

3.1.3 Obstacles

The obstacles in the environment are represented through a rectangular box and a vessel. The rectangle is defined by a length L and breadth B with zero velocity. The square is enclosed by a circular buffer-zone with diameter 100 [m]. The vessel on other hand, is equivalent to the real vessel presented in section 3.1.1. Similar to the square, the vessel is defined by a length L and breadth B, and a steady-state velocity. The velocity is chosen randomly each time in the continuous range [3, 6] to generate additional uncertainty in the environment. By defining a start-position A and final position B, the vessel will follow a straight-path from A to B. However, for the re-pathing case, a sudden change is made in the path of the collision vessel to create unpredictability.

3.2 Machine learning

Because the controller of the vessel is tuned such that the vehicle is able to converge to the path, the remaining problem is to find a suitable path. A suitable path in this manner is a pathway that reaches the endpoint, while satisfying a set of constraints. For a vessel the constraints depends on the mission, environment, and possible obstacles. However, in this thesis we consider the most fundamental ones based on COLREG rule 14 presented in section 2.5, which is related to head-on situation. Hence, the resulting constraints are as follow:

- **Avoid collision**: The most important assignment is to avoid collision at any cost.
- **Satisfy navigation rules**: When maneuvering and avoiding collision, it is vital to comply to seafaring rules such as COLREG, more specifically rule 14a).
- **Take the shortest route**: A large amount of costs is related to seafaring, thus an important objective is to reduce the fuel usage. This can be achieved by taking the shortest route available, among others.

In the following sections a step-wise development of machine learning-based solution is presented to generate a suitable Bézier curve that satisfies the constraints. More specifically, determining the two control points in order to shape the path.

3.2.1 Target variable

In section 3.1.2 it was stated that to generate the path, the remaining problem was to find the control points. Hence, introducing the control points as the target variable of the machine learning-model, may seem as the obvious choice at first sight. However, doing so raises several issues. Assuming the only known information is the vessel- and environment states, how can the training data be sampled, such that when the model is trained, the model is is guaranteed to choose a set of points that satisfy all three constraints as far as possible? An ideal solution may be be to manually recognize adequate paths and later sample them as training data. This is similar to how data extraction is done today. Initially data is collected from databases, file systems and other sources. The data is then manually cleansed, transformed and stored in a suitable way for machine learning. However, considering the complexity of the problem, doing this manually is considered an impossible task. Hence, a score paradigm was developed based on the constraint instead. The aim is to develop a scoring function to differentiate the good paths from the bad paths. Notice that since the score is unavailable during actual decision-making, it cannot be used as a feature in the model. A possible approach may therefore be to use this to filter out adequate paths as training data. However, experiments revealed a new issue. In certain situations, the model frequently predicted control points that violated the first constraint. This is a result of the model trying to generalize and the fact that the model has no information about the inadequate paths that were filtered out. As a consequence, an alternative solution was established where the score instead is used as the target variable. The idea is to use the model to search for a set of control points that is positive and maximizes the score. This is achieved by initially "guessing" a combination of the control points. To limit the problem and satisfy the condition of maximum curvature mentioned in section 3.1.2, the x-points of the control points C_{x_1} and C_{x_2} are manually placed 1/3 and 2/3 along the shortest path between initial position and terminal position. Hence, the remaining points C_{y_1} and C_{y_2} defined between $[y_{boundary\,min}, y_{boundary\,max}]$ = [-400, 400] generates the arc necessary to avoid an obstacle. The chosen control points are then given into the model along with other states, which is then used to predict a score. The combination of of C_{y_1} and C_{y_2} that returns the highest positive score is saved and used to to generate the parametrized cubic Bézier curve, which then is given as input to the maneuvering model. The guessing is made by a random function, which chooses a value between the boundaries for each control point. The resulting model can be summarized to following steps:

1. A simulator containing the simplified Digital Twin is utilized to generate training samples by creating a random path each simulation which is then followed. Information about the control points necessary to recreate the path along with other important states are saved as the features. Additionally, based on the performance of the path in terms of the constraints, the path is given a reward which is saved as a target variable.

2. The data is then used to train a machine learning model, which is selected based on benchmarking.
3. The model is implemented on the real vessel to generate a path between initial position and a terminal position. To generate the path a guess on a set of control points is made, which is then passed to the model to predict a score. By repeating this process K times, an optimal combination of control points are found by maximizing the score.

Additionally, the aim of the solution is to be able to address the re-plan stage as mentioned in section 1.2.5, especially when the head-on vessel makes an unexpected action. To achieve this, the simulator of the digital twin is implemented on the real vessel. For a given rate of repetition along the mission, the current states of the real vessel, obstacle position and the generated path is handed to the simulator to forecast possible collision. If the simulation of the digital twin results in collision, a new path is generated by the model for the real vessel using the current state as initial state.

3.2.2 The Score Paradigm

As mentioned, the score paradigm is developed based on the constraints developed in section 3.2. To address the first constraint a circular safe-zone is established enclosing the obstacle with a radius r. The objective is to define r such that the vessel can avoid doubtful situation as defined in 14c) of COLREG by taking early actions. If the vessel crosses the safe-zone a negative penalty of $R_{Collision}$ = -100 is added to the score and the simulation is terminated. The large negative value is due to the objective of avoiding collision at any cost. Thus, any combination of control points that results in a path with negative score will be distinguished by the model. To determine if a collision occured, equation 2.26 is used.

To comply with COLREG rule 14a) a rectangular reward-zone is created as seen in figure 3.1 and 3.2. From section 2.5 it can be derived that the correct action for a head-on situation is to maneuver starboard. Therefore, the zone is placed on the star-board side relative to the headings of the vessels. The headings of the vessels are determined by the velocity vectors as indicated in the figures. Whenever the vessel navigate on the star-board side in respect to the obstacle, a score of R_{COLREG} is added to the score function. Note that although the circular safe-zone and the rectangular reward-zone intersect, the net score would be strictly negative due to the large value of $R_{Collision}$.

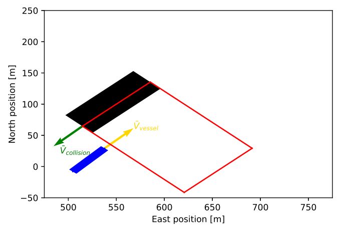

Figure 3.1: Illustration of the definition of reward zone in terms of COLREG 14a), relative to the heading of opposite vessel.

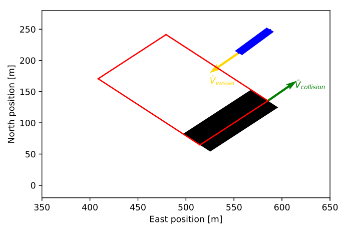

Figure 3.2: Illustration of the definition of reward zone in terms of COLREG 14a), relative to the heading of opposite vessel.

In general, the shortest path from A to B is defined by a straight-line. Hence, the desired behaviour of the vessel is to maneuver in a straight-path whenever there is no object in sight. Additionally, in situation where objects are present, it is still preferred that the vessel is as close to the shortest path as possible in terms of minimizing distance travelled. To achieve this a Gaussian reward function based on the cross-track error introduced in [40] is implemented. In general, the cross-track gives the error between the vessel and the desired straight-line path, and is given by the equation:

$$y_e = -sin(\psi_d)(x(t) - x_0) + cos(\psi_d)(y(t) - y_0) \tag{3.27}$$

$$\psi_d = atan2(y_1, x_1) \tag{3.28}$$

where (x(t), y(t)) is the position of the vessel, (x_0, y_0) and (y_1, x_1) is the initial and final point of the desired straight-path, and ψ_d is the heading of the path. Hence, minimizing $|y_e|$ is equivalent to the ship converging to the desired path. The resulting Gaussian reward function then becomes:

$$R_{Path} = ae^{-\frac{y_e^2}{2\sigma}} \tag{3.29}$$

y_e is the computed cross-track error, σ is the standard deviation and a is the maximum attainable reward, where latter have to be manually chosen. Hence, the Gaussian reward function represents a Gaussian curve with amplitude a, and standard deviation σ as seen in figure 3.3. As illustrated, when the cross-track error between the vessel and path path is below 5 [m], the vessel starts gaining a considerable reward, otherwise nearly 0.

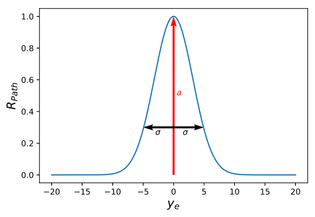

Figure 3.3: Distribution of R_{Path} over y_e for σ=5 and a=1.

In general, R_{Path} had to be manually tuned relative to R_{COLREG}. If R_{COLREG} was chosen too large, the path-planner may generate path that maneuvers star-board even on obstacles in a safe distance from the vessel. At the same time, choosing R_{Path} large would result in straight-line paths being prioritized by the model even when obstacles are present, which naturally would lead to collision. The tuned parameters can be found in table A.2 in appendix A.4, and the resulting score function R_{Total} is given as:

$$R_{Total} = R_{Collision} + R_{COLREG} + R_{Path} \tag{3.30}$$

(3.31)

$$R_{Collision} = \begin{cases} -100, & \text{if inside safe-zone.} \\ 0, & \text{otherwise.} \end{cases} \tag{3.32}$$

(3.33)

$$R_{COLREG} = \begin{cases} 20, & \text{if inside reward-zone.} \\ 0, & \text{otherwise.} \end{cases} \tag{3.34}$$

(3.35)

$$R_{Path} = ae^{-\frac{y_e^2}{2\sigma}} \tag{3.36}$$

3.2.3 Feature selection

Choosing appropriate features is crucial for the model to predict accurately. In our problem the natural selection would be the states of the vessel.

- Distance between the current position and the goal position
- Heading of the vessel
- Velocity of the vessel
- Distance between the current position and position of a potential obstacle
- Velocity of a potential obstacle

Notice that each feature have different unit and size. Hence, to avoid biased influence, each feature had to be individually normalized. Hence, a min-max normalization is applied, where the mathematical expression for each feature can be found in appendix A.5. Before the model can determine a suitable path, it has to be trained on a data sample. This is achieved by initially running numerous simulations with the digital twin where an indiscriminate path is chosen, and based on the performance the path is given a score. More specifically, a random choice of the control points C_1 and C_2 is made, and the selected values are stored as features to recreate the path. To train the re-pathing portion of the model, similar paths are generated. But now the initial states are different than zero.

3.2.4 Model selection

To apply model selection, a training set of size $\approx$ 400 000 is extracted from the data sampled, which is then split into a training and test set. The algorithms mentioned in section 2.6 was considered and compared by benchmarking. The machine learning algorithms are predefined as python functions in the machine learning library Sklearn. Based on the selection criteria MSE, each model was initially fitted with a sample size of 10, 1 000, 10 000 and 100 000 samples with default hyperparameters. The aim was to examine their behaviour and select the three best models for further hyperparameter-tuning. As studied in figure 3.4 the majority of the models performed similarly well. However, it can be

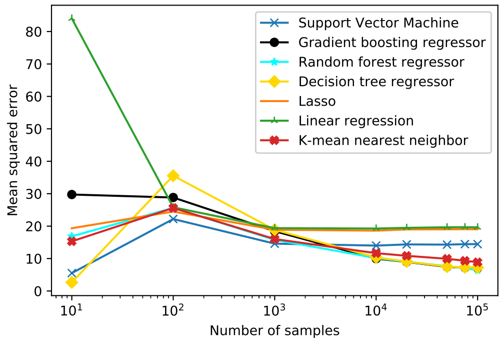

Figure 3.4: Benchmarking a set of regression models in terms of MSE.

noted that model such as the Linear regression performed significantly worse, which implies a non-linear relationship may be present between the features and target variable. Based on the MSE at 100 000 samples, Random forest, Gradient Boosting regressor and Decision tree proved to be the best performing models. Hence, each of the three models were hyperparameter-tuned using a built in cross-validation grid search function. Based on predefined intervals for each respective parameter, the cross-validation grid function fits mixed combinations of the parameters, and returns the best model in terms of lowest MSE. The resulting mean-squared errors were 5.96, 5.67, and 6.35 for Random forest, Gradient Boosting regressor and Decision tree, respectively. Hence, Gradient Boosting regressor was chosen as the final model with the corresponding hyperparameters found in table A.3. The resulting feature importance plot of the model seen in figure 3.5 gives an overview of how significant each feature is relative to each other in terms of predicting the target variable. Based on the plot, it is evident that the relative distance to an obstacle in y-direction, which corresponds to the East-axis, has the most impact. This is not a surprise considering the vessel mainly maneuvers in this direction. Hence, depending on whether the obstacle is nearby or not, the resulting score changes notably. Further, it can be noted that the control points have the second most impact, which is to be expected as they define the final path taken. Naturally, when a bad path is chosen, the score is affected accordingly. Finally, it can be noted that the initial heading ψ have zero impact on the target variable. This is due to the fact that the initial heading is always initialized at 0, thus having limited effect on the outcome. The same goes with the velocity, as it usually is kept constant. However, it is still included in the model in terms of future extensions of the model.

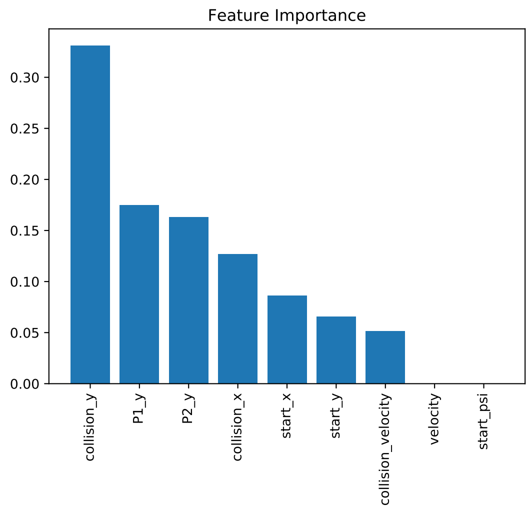

Figure 3.5: Importance plot of the features, where y-scale indicates how big impact from 0 to 1 the respective feature have on the prediction.

Afterwards, the number of iterations K had to be found. Larger K, results in more optimal combinations of C_{y_1} and C_{y_2} in exchange for run-time. To find a suitable value, K was set initially to a large value K = 40 000. Following this, 100 000 model cases were simulated, and the number of iterations that resulted in the best combination was noted. Only few cases required more than 8 000 iterations, as observed in figure 3.6. The mean and standard deviation values of the observations were found to be 1 565.12 and 1 249.10, respectively. Hence, K = 3 000 was chosen to cover the significant areas. Although K = 3 000 may seem like a large number, the whole process of generating path takes less than 10 seconds on an average computer.

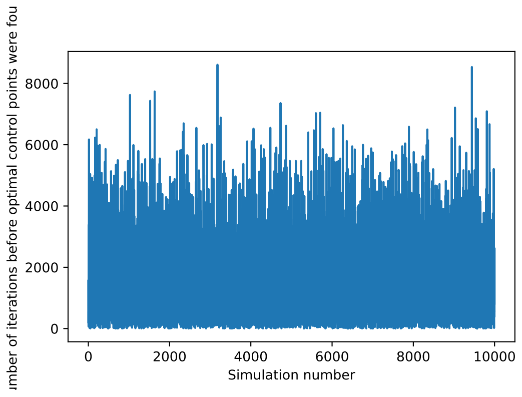

Figure 3.6: Number of iterations before optimal control points were found, for 10000 simulations.

3.3 Test scenarios

In order to evaluate the trained model, a set of scenarios are established with the simulator of the real vessel in the following sections. Note that the model was never trained on the real vessel. Hence, the objective is to determine how well a model that is trained on a simplified model performs on the real model.

3.3.1 Test scenario 1: Straight-line path-planning

The first scenario aims to verify that the vessel chooses a straight-line pathing from initial position to goal position when no obstacle is present. To assure that the chosen path is not by accident, three different cases are considered:

1. End-point is placed straight-ahead of the vessel to verify that the vessel can stay on the straight-path.
2. End-point is placed in upper diagonal of the vessel. The objective is to showcase that the vessel can choose the shortest-path, even when the initial heading is different from the path heading. if the initial heading of the vessel is different than the heading of the path to verify that when the initial heading is different than the heading of the path
3. The last case is similar to the second case, but the goal is now placed in the lower diagonal.

3.3.2 Test scenario 2: Path-planning with collision-avoidance of static obstacle

In second scenario, the obstacle-avoidance of the path-planning is evaluated. Hence, a static obstacle is added where the aim is to demonstrate that the vessel maneuvers in accordance with COLREG rule 14a) during head-on situations. Similar to previous scenario, three different cases will be evaluated:

1. In first case the end-point is placed straight-ahead of the vessel. The aim is to showcase that the vessel maneuvers correctly in terms of COLREG.
2. In the second case, the end-point is placed diagonal to the vessel, and the obstacle is placed on the path, to further justify that COLREG is followed.
3. Finally, the obstacle is placed in the environment, but outside the desired path of the vessel. The goal is to showcase that the vessel is capable of pathing along the shortest path, when obstacle is present, but not in sight of the path.

3.3.3 Test scenario 3: Path-planning with collision-avoidance of dynamic vessel

Equivalent to previous scenario, the obstacle-avoidance property of the solution will be evaluated again. However, now the static obstacle is replaced with a real vessel containing vessel dynamics. The following three cases are considered:

1. The first case is equivalent to previous scenario, the head-on vessel is placed straight-ahead of the vessel, and moving in a straight-line towards the vessel. Hence, the objective is to showcase that the model satisfy COLREG 14a).
2. In the second case, the end-point is placed diagonal to the vessel, and the obstacle is placed on the path, moving towards the vessel, to justify the model.
3. Lastly, the obstacle is placed outside the path, to again showcase that the path-planner is able to generate a straight-path.

3.3.4 Test scenario 4: Re-planning using the Digital twin

In the final scenario, the developed digital twin will be implemented inside the real vessel. As previously mentioned, for a defined frequency, the current states of the vessel, position and velocity of the present obstacle and the generated path is given to the model to simulate and forecast possible collision. If this is the case, a new path is generated by the model for the real vessel, as part of the re-plan step. To simulate this, the head-on vessel in the real simulator is forced to make a sudden change in the direction, and the resulting behaviour of the vessel will then be evaluated.

Chapter 4

Results and discussion

In the following chapter the result from each scenario is presented, followed by a detailed discussion of the outcome. Additional plots of velocity ν, commanded thrust $\tau_{command}$ and position η can be found in appendix A.8. Note that the dashed black lines represents the bounded environment defined previously, with the goal state indicated by red cross. The black and gray path represents true and desired path of the real vessel, respectively. Finally, the red rectangle indicates the obstacle, where the blue line is the respective path in the case of dynamic vessel.

4.1 Results

4.1.1 Test scenario 1: Straight-line path-planning

The respective cases can be found below. As observed, the path-planning is capable of generating a straight-line when no obstacle is present. Although there is a small deviation < 10 [m], recall that σ was chosen as 10 for the Gaussian reward. Hence, further tightening of the parameter may lead to reduced error, but would result in lower probability of discovering the shortest-path, and thus longer time to reach the terminal state. Further, it can be noticed that the commanded thrust from the controller is generally smooth, except for a small spike during the initial state. This is partly due to the initial heading being different than desired heading when the simulation is executed, but can also be further tuned to achieve more smoothness. This addresses some of the issues discussed previously concerning unpredictable output from deep learning-based models, for instance showcased in [39]. Since the action control is isolated to the controller, the noisy input to the thrusters can simply be tuned based on general control theory.

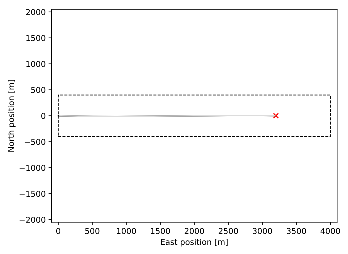

Figure 4.1: Case 1.1, straight-line path-planning. The desired point is placed in (3200, 0).

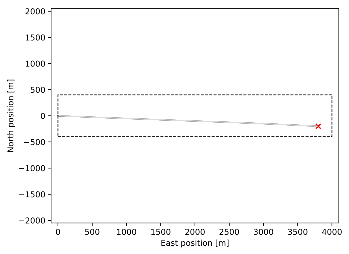

Figure 4.2: Case 1.2, straight-line path-planning. The desired point is placed in (3800, -200).

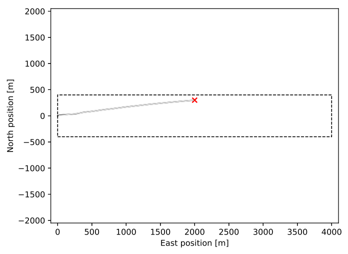

Figure 4.3: Case 1.3, straight-line path-planning. The desired point is placed in (2000, 300).

4.1.2 Test scenario 2: Path-planning with collision-avoidance of static obstacle

Studying the figures 4.4 and 4.5, it is evident that the path-planning is able to generate a path that complies with COLREG. Additionally, figure 4.6 verifies that the path-planning chooses the shortest path when the present obstacle is not blocking the path.

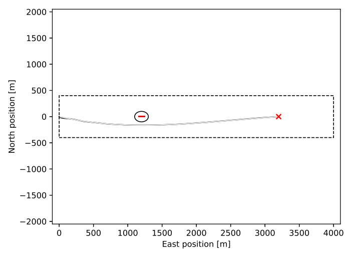

Figure 4.4: Case 2.1, static obstacle-avoidance. The desired point is placed in (3200, 0).

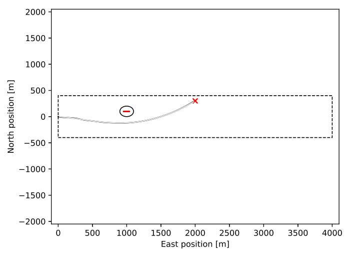

Figure 4.5: Case 2.2, static obstacle-avoidance. The desired point is placed in (2000, 300).

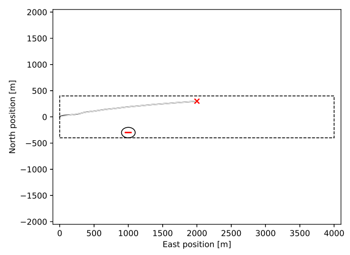

Figure 4.6: Case 2.3, static obstacle-avoidance. The desired point is placed in (2000, 300).

4.1.3 Test scenario 3: Path-planning with Collision-avoidance of dynamic vessel

Figure 4.7 and 4.8 confirms that the model is capable of complying with COLREG during head-on situation with real vessel as well. Further, the path-planner showcases in figure 4.9 that it is able to generate a straight-path when the vessel is not present in the path.

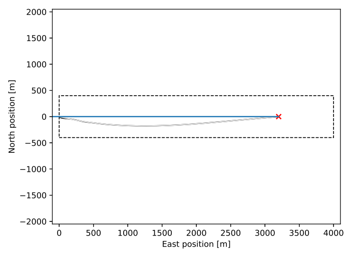

Figure 4.7: Case 3.1, dynamic obstacle-avoidance. The desired point is placed in (3400, 300).

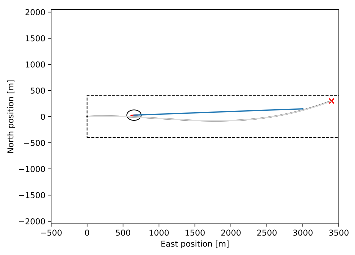

Figure 4.8: Case 3.2, dynamic obstacle-avoidance. The desired point is placed in (3400, 300).

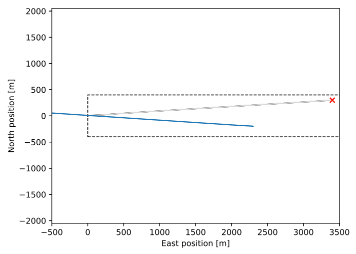

Figure 4.9: Case 3.3, dynamic obstacle-avoidance. The desired point is placed in (3400, 300).

4.1.4 Test scenario 4: Re-planning using the Digital Twin

Based on figure 4.10 the vessel is capable of adjusting, when the head-on vessel make a sudden change in its pathing. However, notice that the change in path is aggressive. This is to be expected as the resulting action is a consequence of an unexpected behaviour of the head-on vessel. Thus, a similar instantaneous act is necessary. Still, the response can possibly be further improved by adding a filter. An alternative solution may be to define a Minimum Risk Condition as defined in section 1.2.5, and force the vessel to such a state, when similar situation occurs.

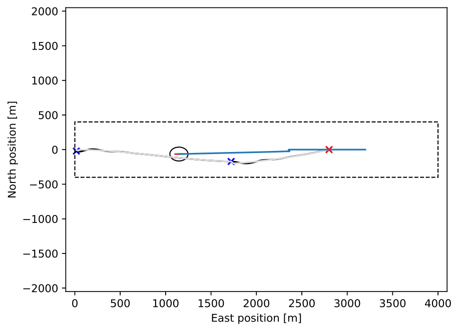

Figure 4.10: Case 4, re-planning using digital twin. The desired point is placed in (2800, 0)

4.2 General discussion

Although the model performs well in general, there are still a few drawbacks that need to be addressed. The first being the limited rectangular state-space. As previously mentioned, the main motivation behind the shape is simply to experiment on head-on situations easier. Besides, the model still allows re-scaling of the environment, but re-sampling in the new environment in addition to re-training is necessary. In terms of real sea state, where the state-space is undisclosed, this may cause a challenge. However, considering the states of the model are scaled to relative values, a possible way to confront the issue is to develop a similar design to way-point tracking presented in [61]. The main idea is to then generate a path between two way-points, which are placed in a similar area defined by the boundaries of the model. Then, when a way-point is reached, the environment can be "moved" to enclose the next to way-points. A second weakness of the model, is the lack of environmental variables. Especially, due to the resulting complexity of the sea state. Although the performance of the model was not tested with environment forces present, it is obvious the model would perform bad considering supervised-models do not adapt well to unknown environment in contrast to deep learning-models. Hence, a possible solution may be to measure the environmental variables and add it to the feature space. However, accurate measurements can in many cases be impossible to obtain. Therefore, a more superior solution may be to delegate the problem to the controller. For instance, applying a hybrid controller given in [71] or an adaptive controller similar to solution presented in [72].

Chapter 5

Conclusion and further work

5.1 Conclusion

The development of the proposed solution in this thesis, presents an adequate process in building a digital twin in generating software-based solutions. In addition, an approach on utilizing the digital twin to attain a higher level of autonomy is successfully presented. Predominantly the model performs well on the simulated scenarios. Especially, in demonstrating the ability to comply to COLREG when necessary. Additionally, the model is also capable of answering to the base requirements of the class society. In contrast to some of the deep learning-based solutions presented in section 1.2.4, the model is able to give the transparency of the chosen path. However, the solution displays some weakness such as limited state-space and no consideration of environmental variables. On other hand, a few possible solutions have been suggested such as a way-point-based approach and hybrid controller. The latter showcase the advantage of not combining path-planning and action control in the same model, in a similar manner to [34] and [35]. Especially, since it enables the possibility of utilizing traditional control theory, which generally have performed successfully on real life applications.

5.2 Further work

In terms of further development, there are several areas that can be considered, such as introducing additional COLREG rules and increasing the degree-of-freedom of the Bézier curves to generate more complex paths. Additional constraints can also be considered in the score paradigm, such as the turning rate of the path. Considering the initial idea was to apply the solution on a model-scaled vessel, this still remains relevant. Finally, a development of way-point-based based path generation or application of a hybrid controller for environmental consideration may be suitable in order to extend the application of the model

Bibliography

[1] Yoav Shoham, Raymond Perrault, Erik Brynjolfsson, and Jack Clark. The ai index 2018 annual report. 12 2018.

[2] Asgeir J. Sørensen. *Marine Cybernetics Towards Autonomous Marine Operations and Systems*. Department of Marine Technology, NTNU, 2018.

[3] Kongsberg group. Autonomous ship project, key facts about yara birkeland. https://www.kongsberg.com/maritime/support/themes/autonomous-ship-project-key-facts-about-yara-birkeland/?OpenDocument=. Accessed: 20.September.2019.

[4] J. Ørnulf Rødseth and H. C. Burmeister. Developments toward the unmanned ship. *in proceedings of International Symposium Information on Ships – ISIS 2012*, August 2012.

[5] F. Eiliv. 119 enova-millioner til askos autonome fartøy. http://presse.enova.no/news/119-enova-millioner-til-askos-autonome-fartoey-362196, mar 2019. Accessed: 10.September.2019.

[6] Einar O. Stangvik, Oda L. Skjetne, Tom Byermoen, Endre Alsaker-Nøstdahl, and Harald Vikøyr. Krigsskipet som krasjet og sank. https://www.vg.no/spesial/2018/helge-ingstad-ulykken/. Accessed: 10.Februar.2020.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

[8] Ryan Proud, Jeremy Hart, and Richard Mrozinski. Methods for determining the level of autonomy to design into a human spaceflight vehicle: A function specific approach. page 15, 09 2003.

[9] Ingrid Utne, Asgeir Sørensen, and Ingrid Schjølberg. Risk management of autonomous marine systems and operations. page V03BT02A020, 06 2017.

[10] L. Perera. Deep learning towards autonomous ship navigation and possible colregs failures. *Journal of Offshore Mechanics and Arctic Engineering*, May 2019.

[11] Anastasios M. Lekkas. *Guidance and Path-Planning Systems for Autonomous Vehicles*. PhD thesis, 04 2014.

[12] Roger Skjetne. The maneuvering problem. 03 2005.

[13] Fadzli Syed Abdullah, Sani Iyal, Mokhairi Makhtar, and Azrul Amri Jamal. Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries. 12 2015.

[14] B. Garau, A. Alvarez, and Gabriel Oliver. Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an a* approach. volume 2005, pages 194 – 198, May 2005.

[15] Byunghyun Yoo and Jinwhan Kim. Path optimization for marine vehicles in ocean currents using reinforcement learning. *Journal of Marine Science and Technology*, 21, December 2015.

[16] Tsourdos Antonios, Steven White, and Madhavan Shanmugavel. *Cooperative Path Planning of Unmanned Aerial Vehicles*. John Wiley Sons, 2011.

[17] J.S. Yoon, B.C. Min, S.O. Shin, Wonse Jo, and D.H. Kim. Ga-based optimal waypoint design for improved path following of mobile robot. *Advances in Intelligent Systems and Computing*, 274:127–136, 01 2014.

[18] Qiu Quan and Han Jian. 2.5-dimensional angle potential field algorithm forthe real-time autonomous navigation of outdoormobile robots. *Sciece China. Information Sciences*, 2011.

[19] A. Abbadi, Radek Matousek, S. Jancik, and Jan Roupec. Rapidly-exploring random trees: 3d planning. pages 594–599, 06 2012.

[20] Hangeun Kim, Taehwan Lee, Hyun Chung, Namsun Son, and Hyun Myung. Any-angle path planning with limit-cycle circle set for marine surface vehicle. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2275–2280, 05 2012.

[21] Kenny Daniel, Alex Nash, Sven Koenig, and Ariel Felner. Theta*: Any-angle path planning on grids. *J. Artif. Intell. Res. (JAIR)*, 39, January 2014.

[22] Quan Shi, Li Tieshan, Qihe Shan, Yuchi Cao, Xiaoqing Fan, and Shengrui Tang. Multiple marine vessels formation control with collision avoidance. *2019 Chinese Control And Decision Conference*, 2019.

[23] Morten Pedersen. Marine vessel path planning and guidance using potential flow. pages 188–193, 09 2012.

[24] Vahid Hassani and Simen Lande. Path planning for marine vehicles using bézier curves. *IFAC-PapersOnLine*, 51:305–310, 01 2018.

[25] Tonje Midjås. Sbmpc collision avoidance for the revolt model-scale ship. Master's thesis, NTNU, June 2019.

[26] I. Hagen, D. Kufoalor, Edmund Brekke, and T. Johansen. Mpc-based collision avoidance strategy for existing marine vessel guidance systems. pages 7618–7623, May 2018.

[27] Tayfun Uyanık, Yasin Arslanoğlu, and Özcan Kalenderli. Ship fuel consumption prediction with machine learning. 04 2019.

[28] Herry Susanto and Gunawan Wibisono. Machine learning for data processing in vessel telemetry system: Initial study. pages 496–501, 03 2019.

[29] Xiaoyun Lei, Zhian Zhang, and Peifang Dong. Dynamic path planning of unknown environment based on deep reinforcement learning. *Journal of Robotics*, 2018:1–10, September 2018.

[30] Winfried Lötzsch. Using deep reinforcement learning for the continuous control of robotic arms. Master's thesis, Technische Universität Chemnitz, 2019.

[31] Xiaowei Xing and Dong Eui Chang. Deep reinforcement learning based robot arm manipulation with efficient training data through simulation, 2019.

[32] Thomas Duriez, Steven Brunton, and Bernd Noack. *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Springer, September 2016.

[33] Mathworks. Reinforcement learning. https://www.mathworks.com/solutions/deep-learning/reinforcement-learning.html, 2020. Accessed: 20.April.2020.

[34] Chanjei Vasanthan. *Combining reinforcement learning and model-based training for collision avoidance*. Norwegian University of Science and Technology, 2019.

[35] Mathias G. Aronsen. *Path Planning and Obstacle Avoidance for Marine Vessels using the Deep Deterministic Policy Gradient Method*. Norwegian University of Science and Technology, 2019.

[36] A. Hussein, E. Elyan, M. M. Gaber, and C. Jayne. Deep reward shaping from demonstrations. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 510–517, 2017.

[37] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. 03 2018.

[38] Samit Bhanja and Abhishek Das. Impact of data normalization on deep neural network for time series forecasting, 2018.

[39] Ingunn J. Vallestad. *Path Following and Collision Avoidance for Marine Vessels with Deep Reinforcement Learning.* Norwegian University of Science and Technology, 2019.

[40] Andreas M. Bell and Anastasios M. Lekkas. Curved path following with deep reinforcement learning: Results from three vessel models. *Conference: OCEANS 2018 MTS/IEEE Charleston*, oct 2018.

[41] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *Modeling, Identification and Control*, 30, April 2019.

[42] Fangxing Li and Yan Du. From alphago to power system ai: What engineers can learn from solving the most complex board game. *IEEE Power and Energy Magazine*, 16:76–84, 03 2018.

[43] Marzyeh Ghassemi, Tristan Naumann, Peter Schulam, Andrew Beam, Irene Chen, and Rajesh Ranganath. A review of challenges and opportunities in machine learning for health. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2020:191–200, 05 2020.

[44] Christopher J. Kelly, Alan Karthikesalingam, Mustafa Suleyman, Greg Corrado, and Dominic King. Key challenges for delivering clinical impact with artificial intelligence. *BMC Medicine*, 17(1):195, 2019.

[45] DNV GL. Class guideline: Autonomous and remotely operated ships. 9 2018.

[46] Lloyd's Register. Lr code for unmanned marine systems. 2 2017.

[47] Vanessa Buhrmester, David Muench, and Michael Arens. Analysis of explainers of black box deep neural networks for computer vision: A survey, 11 2019.

[48] Roland Rosen, Georg Wichert, George Lo, and Kurt Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 48:567–572, 12 2015.

[49] Stefan Boschert, Christoph Heinrich, and R. Rosen. Next generation digital twin. 05 2018.

[50] Gaute Storhaug. Digital twins and sensor monitoring. https://www.dnvgl.com/expert-story/maritime-impact/Digital-twins-and-sensor-monitoring.html. Accessed: 1.June.2020.

[51] M. A. Lund, K. Mochel, W. J. Lin, R. Onetto, J. Srinivasan, J. Gregg, E. J. Bergman, D. K. Hartling, and A. and Ahmed. Digital twin interface for operating wind farms. june 2018. uS Patent 9,995,278.

[52] Edward Glaessgen and David Stargel. The digital twin paradigm for future nasa and u.s. air force vehicles. 04 2012.

[53] Wil Danilczyk, Yan Sun, and Haibo He. Angel: An intelligent digital twin framework for microgrid security. pages 1–6, 10 2019.

[54] Alexander Danielsen-Haces. Digital twin development. Master's thesis, NTNU, jun 2018.

[55] The Kongsberg group. Kognitwin energy. https://www.kongsberg.com/no/digital/solutions/kognitwin-energy/. Accessed: 1.June.2020.

[56] Yuan Tian. Development and application of digital twin in marine propulsion system. Master's thesis, NTNU, jun 2019.

[57] General Electrics digital. What is a digital twin? https://www.ge.com/digital/applications/digital-twin. Accessed: 1.June.2020.

[58] T. Blochwitz, Martin Otter, Johan Åkesson, Martin Arnold, C. Clauß, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. 09 2012.

[59] Catia-systems. Fmpy. https://github.com/fmi-tools/FMPy. Accessed: 31.September.2019.

[60] Şule Şahin, Carmen Boado Penas, Corina Constantinescu, Julia Eisenberg, Kira Henshaw, Maoqi Hu, Jing Wang, and Wei Zhu. Covid19. 04 2020.

[61] Thor I. Fossen. *HANDBOOK OF MARINE CRAFT HYDRODYNAMICS AND MOTION CONTROL*. John Wiley Sons, 2011.

[62] Thomas W. Sederberg. *Computer Aided Geometric Design*. Brigham Youth University, All faculty publication, 2012.

[63] GeeksForGeeks. How to check if two given line segments intersect? https://www.cdn.geeksforgeeks.org/check-if-two-given-line-segments-intersect/. Accessed: 20.April.2020.

[64] Lloyd's Register. Colregs - international regulations for preventing collisions at sea. http://www.mar.ist.utl.pt/mventura/Projecto-Navios-I/IMO-Conventions Accessed: 10.June.2020.

[65] Arthur L. Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. *IBM*, nov 1967.

[66] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

[67] Biagio Ciuffo and Vincenzo Punzo. “no free lunch” theorems applied to the calibration of traffic simulation models. *Intelligent Transportation Systems, IEEE Transactions on*, 15:553–562, 04 2014.

[68] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013.

[69] Arvind Keprate and R.M. Ratnayake. Using gradient boosting regressor to predict stress intensity factor of a crack propagating in small bore piping. 12 2017.

[70] Catia-systems. Fmikit-simulink. https://github.com/CATIA-Systems/FMIKit-Simulink. Accessed: 20.September.2019.

[71] Dong Trong Nguyen, Asgeir J. Sørensen, and Ser Tong Quek. Design of hybrid controller for dynamic positioning from calm to extreme sea conditions. *Automatica*, 43(5):768–785, May 2007.

[72] Vahid Hassani, Asgeir Sørensen, and Antonio Pascoal. A novel methodology for robust dynamic positioning of marine vessels: Theory and experiments. pages 560–565, 06 2013.

Appendix A

Appendix

A.1 Gunnerus vessel plant model matrices

A.1.1 Rigid body mass matrix

$$M_{Rb} = \begin{bmatrix} 418061 & 0 & 0 & 0 & 0 & 0 \\ 0 & 418061 & 0 & 0 & 0 & 0 \\ 0 & 0 & 418061 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6164560 & 0 & 0 \\ 0 & 0 & 0 & 0 & 21823046 & 0 \\ 0 & 0 & 0 & 0 & 0 & 21823046 \end{bmatrix} \tag{A.1}$$

A.1.2 Added mass matrix

$$M_A = \begin{bmatrix} 74327.85 & 0 & 0 & 0 & 0 & 0 \\ 0 & 342230.06 & 0 & 356613.59 & 0 & -231035.80 \\ 0 & 0 & 2432295.50 & 0 & 5908110 & 0 \\ 0 & 356613.59 & 0 & 2024007.10 & 0 & -555383.94 \\ 0 & 0 & 5908110 & 0 & 128795620 & 0 \\ 0 & -231035.80 & 0 & -555383.94 & 0 & 20235290 \end{bmatrix} \tag{A.2}$$

A.1.3 Restoring forces matrix

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2334147.80 & 0 & 3470870.80 & 0 \\ 0 & 0 & 0 & 10923388 & 0 & 54.38 \\ 0 & 0 & 3470870.80 & 0 & 129373300 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.3}$$

A.1.4 Linear damping matrix

$$D_L = \begin{bmatrix} 17447.62 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80334.62 & 0 & 167.82 & 0 & 114.64 \\ 0 & 0 & 403981.84 & 0 & 934606.88 & 0 \\ 0 & 167.82 & 0 & 195452.03 & 0 & 432.05 \\ 0 & 0 & 934606.88 & 0 & 20702550 & 0 \\ 0 & 114.63 & 0 & 432.05 & 0 & 4974997.40 \end{bmatrix} \tag{A.4}$$

A.2 Simplified vessel plant model matrices

A.2.1 Rigid body mass matrix

$$M_{Rb} = \begin{bmatrix} 418061 & 0 & 0 \\ 0 & 418061 & 0 \\ 0 & 0 & 21823046 \end{bmatrix} \tag{A.5}$$

A.2.2 Added mass matrix

$$M_A = \begin{bmatrix} 74327.85 & 0 & 0 \\ 0 & 342230.06 & -231035.80 \\ 0 & -231035.80 & 20235290 \end{bmatrix} \tag{A.6}$$

A.2.3 Linear damping matrix

$$D_L = \begin{bmatrix} 17447.62 & 0 & 0 \\ 0 & 80334.62 & 114.64 \\ 0 & 114.63 & 4974997.40 \end{bmatrix} \tag{A.7}$$

A.3 The Maneuvering controller parameters

Parameter	**Value [units]**
K_p	diag($\begin{bmatrix}10 & 0.2 & 0.5\end{bmatrix}$) [-]
K_d	diag($\begin{bmatrix}500 & 100000 & 5000\end{bmatrix}$) [-]
μ	0.00005 [-]
U_{ref}	5 [m/s]

Table A.1: Controller gains and parameters

A.4 Parameters for the score paradigm

Description	**Symbol**	**Value**
Reward for collision	$R_{collision}$	-100
Reward for complying with COLREG 14a)	R_{COLREG}	20
Maximum reward attainable for R_{Path}	a	1
Standard deviation for R_{Path}	σ	10

Table A.2: Parameters for the score variables

A.5 Normalization of features

- **Relative position to the goal position**: The relative position from initial state to objective position is calculated based on the euclidean distance and scaled with the environment to be in the range [-1, 1]. Mathematically we derive them as:

$$x_{goalrelative} = \frac{x_{current} - x_{goal}}{x_{boundarymax} - x_{boundarymin}} \tag{A.8}$$

$$y_{goalrelative} = \frac{y_{current} - y_{goal}}{y_{boundary_{max}} - y_{boundary_{min}}} \tag{A.9}$$

- **Relative heading**: The heading is originally defined in radians, but adjusted such that every time it exceeds 2π, the heading is subtracted with a corresponding value. Similarly, when the heading is below 0, 2π is added. Additionally, the heading is scaled by dividing with 2π, such that the relative heading always remains in the boundary [0, 1].

$$\psi_{relative} = \frac{\psi_{current}}{2\pi} \tag{A.10}$$

- **Relative velocity**: The relative velocity is scaled by dividing the maximum allowed velocity of the vessel, thereby defined in the boundary [0, 1].

$$V_{relative} = \frac{V_{V_current}}{V_max} \tag{A.11}$$

- **Relative position to a potential obstacle**: The relative position is calculated in similar manner to the relative position to goal position with the equations:

$$x_{obstaclerelative} = \frac{x_{current} - x_{obstacle}}{x_{boundarymax} - x_{boundarymin}} \tag{A.12}$$

$$y_{obstaclerelative} = \frac{y_{current} - y_{obstacle}}{y_{boundarymax} - y_{boundarymin}} \tag{A.13}$$

  However, unlike the relative position to goal position, the distance is set to [-1, 1] (depending on orientation) if the obstacle is outside the boundary, too far away to be considered.

- **Relative velocity of collision obstacle**: The relative velocity of the obstacle is calculated equivalent to the vessel:

$$V_{relative} = \frac{V_{V_current}}{V_max} \tag{A.14}$$

- **Control point values**: The control points value C_{y1} and C_{y2} are defined in the boundary [-1, 1].

A.6 Hyperparameters for Gradient Boosting regressor

Description	**Symbol**	**Value**
Numbers of trees	C	104
Decides how much each tree contributes (Learning rate)	λ	0.1
Maximum depth allowed per tree	d	18

Table A.3: Tuned hyperparameters for Gradient Boosting regressor

A.7 Intersection and orientation of lines: Examples

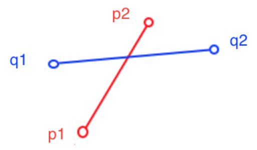

Figure A.1: Example 1: Orientation of (p_1, p_2, q_1) is counterclockwise and orientation of (p_1, p_2, q_2) is clockwise, and therefore different. Correspondingly, orientation of (q_1, q_2, p_1) is clockwise and orientation of (q_1, q_2, p_2) is counterclockwise. Thus, the general case is satisfied in this example.

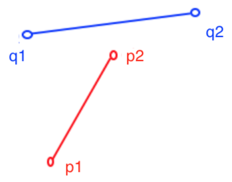

Figure A.2: Example 2: Orientation of (p_1, p_2, q_1) is collinear and orientation of (p_1, p_2, q_2) is counterclockwise, and therefore different. In a similar manner, orientation of (q_1, q_2, p_1) is counterclockwise and orientation of (q_1, q_2, p_2) is wise, which means the general case is again satisfied.

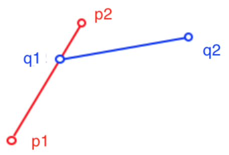

Figure A.3: Example 3: In a similar manner to example 1, orientation of (p_1, p_2, q_1) is counterclockwise and orientation of (p_1, p_2, q_2) is clockwise. However, both (q_1, q_2, p_1) and (q_1, q_2, p_2) are clockwise, and the general case is not satisfied.

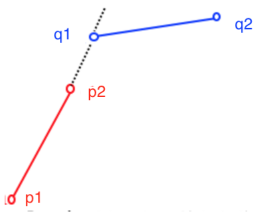

Figure A.4: Example 4: In the following example, only (p_1, p_2, q_1) is collinear, while (p_1, p_2, q_2), (q_1, q_2, p_1) and (q_1, q_2, p_2) are all clockwise, and the conditions for the general case is unfulfilled.

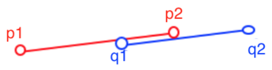

Figure A.5: Example 5: In this case all the triplet of points are collinear, which means the general case is not satisfied. in other hand, both x-projection and y-projection of line segment p and q intersect. Thus, the special case is satisfied.

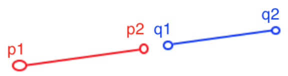

Figure A.6: Example 6: Equivalent to example 5, all the triplet of points are collinear. However, since the projections do not intersect, neither the general case nor the special case is satisfied.

A.8 Simulation results

A.8.1 Straight-line path-planning

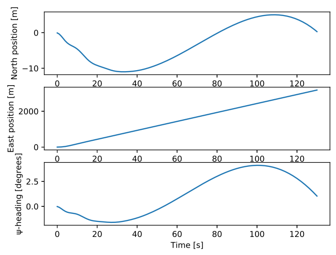

Figure A.7: Case 1: Plot of North-, East- and ψ-position.

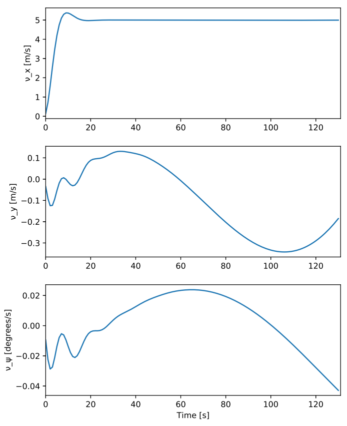

Figure A.8: Case 1: Plot of velocities in each direction with respect to the vessel coordinates.

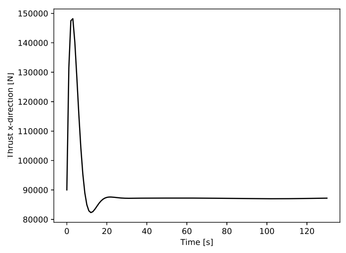

Figure A.9: Case 1: Plot of commanded surge thrust.

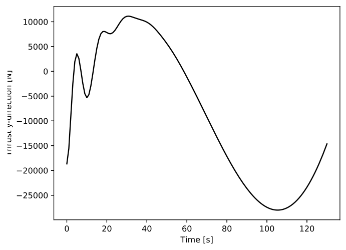

Figure A.10: Case 1: Plot of commanded sway thrust.

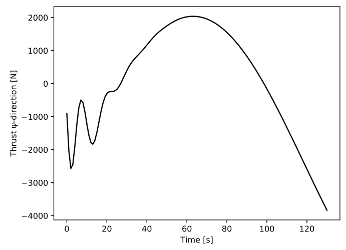

Figure A.11: Case 1: Plot of commanded yaw-moment.

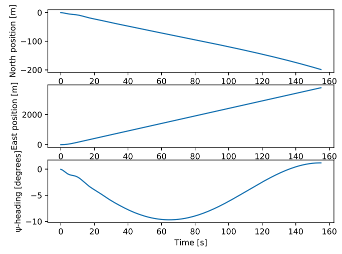

Figure A.12: Case 2: Plot of North-, East- and ψ-position.

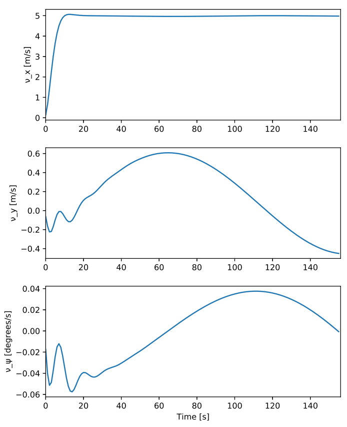

Figure A.13: Case 2: Plot of velocities in each direction with respect to the vessel coordinates.

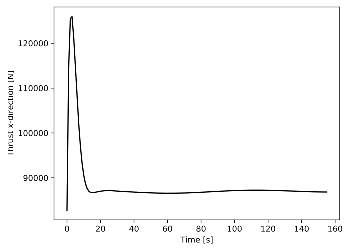

Figure A.14: Case 2: Plot of commanded surge thrust.

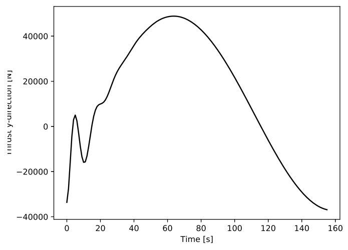

Figure A.15: Case 2: Plot of commanded sway thrust.

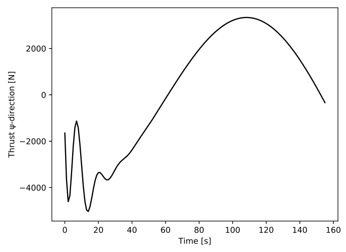

Figure A.16: Case 2: Plot of commanded yaw-moment.

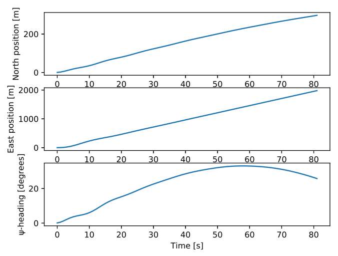

Figure A.17: Case 3: Plot of North-, East- and ψ-position.

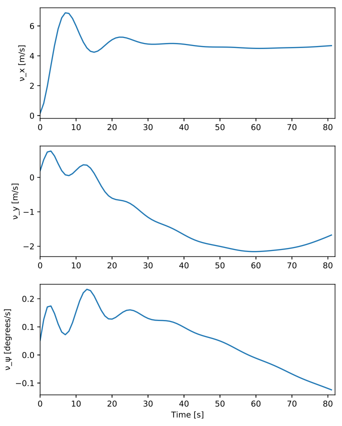

Figure A.18: Case 3: Plot of velocities in each direction with respect to the vessel coordinates.

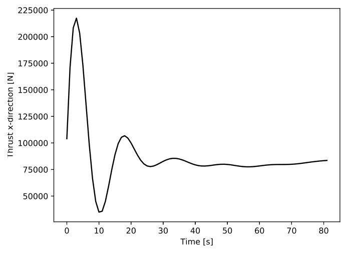

Figure A.19: Case 3: Plot of commanded surge thrust.

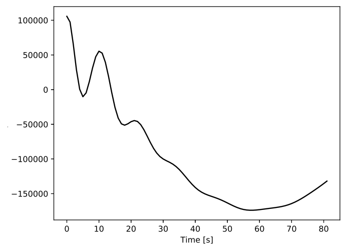

Figure A.20: Case 3: Plot of commanded sway thrust.

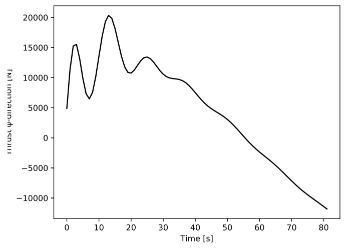

Figure A.21: Case 3: Plot of commanded yaw-moment.

A.8.2 Test scenario 2

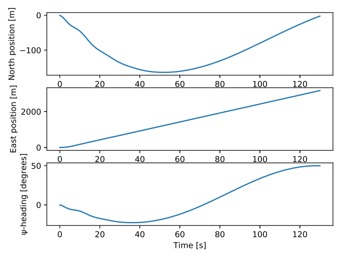

Figure A.22: Case 1: Plot of North-, East- and ψ-position.

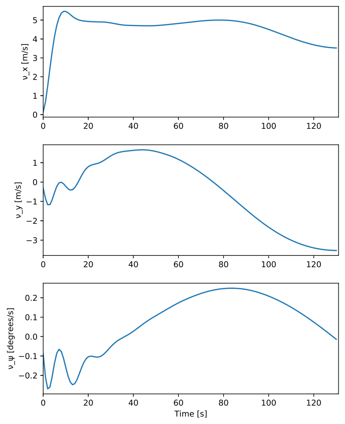

Figure A.23: Case 1: Plot of velocities in each direction with respect to the vessel coordinates.

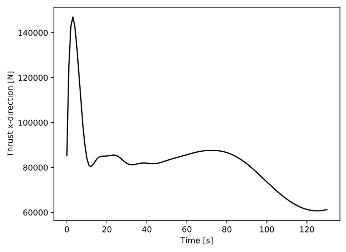

Figure A.24: Case 1: Plot of commanded surge thrust.

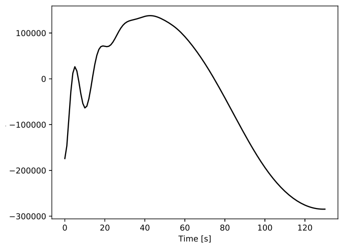

Figure A.25: Case 1: Plot of commanded sway thrust.

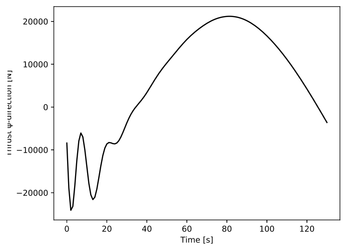

Figure A.26: Case 1: Plot of commanded yaw-moment.

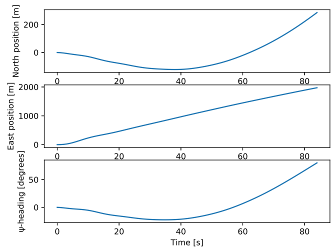

Figure A.27: Case 2: Plot of North-, East- and ψ-position.

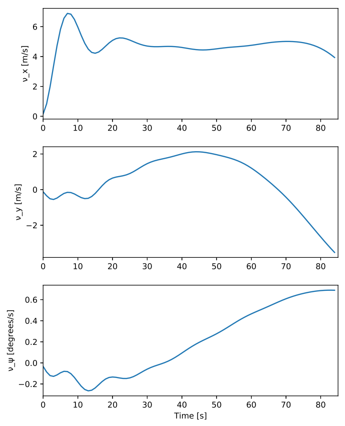

Figure A.28: Case 2: Plot of velocities in each direction with respect to the vessel coordinates.

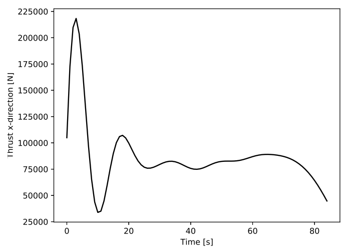

Figure A.29: Case 2: Plot of commanded surge thrust.

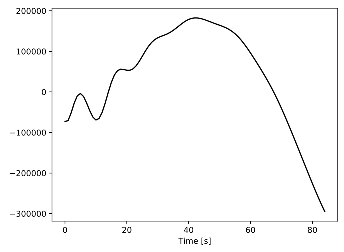

Figure A.30: Case 2: Plot of commanded sway thrust.

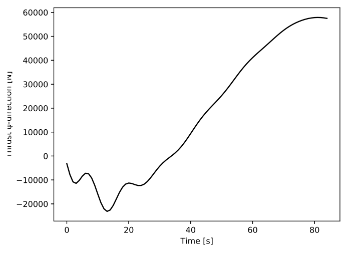

Figure A.31: Case 2: Plot of commanded yaw-moment.

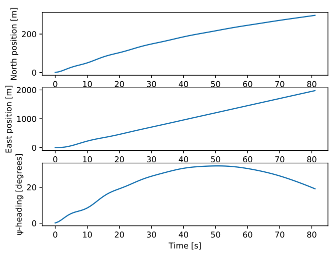

Figure A.32: Case 3: Plot of North-, East- and ψ-position.

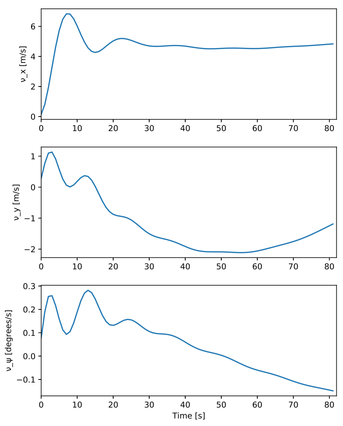

Figure A.33: Case 3: Plot of velocities in each direction with respect to the vessel coordinates.

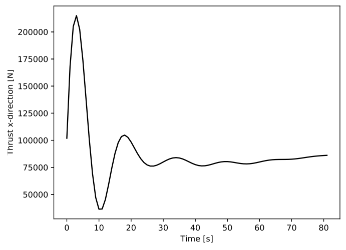

Figure A.34: Case 3: Plot of commanded surge thrust.

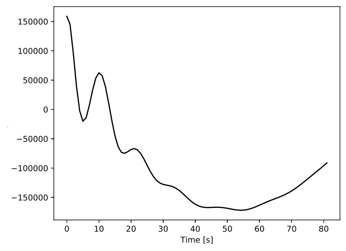

Figure A.35: Case 3: Plot of commanded sway thrust.

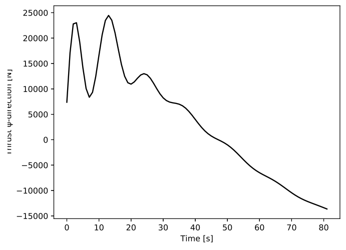

Figure A.36: Case 3: Plot of commanded yaw-moment.

A.8.3 Test scenario 3: Path-planning with Collision-avoidance of dynamic vessel

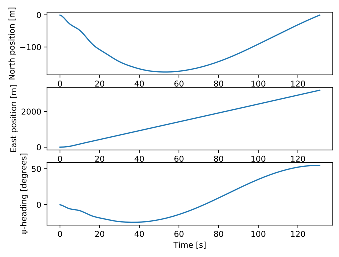

Figure A.37: Case 1: Plot of North-, East- and ψ-position.

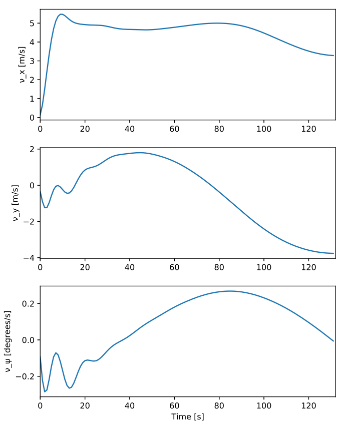

Figure A.38: Case 1: Plot of velocities in each direction with respect to the vessel coordinates.

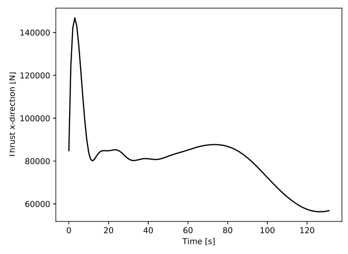

Figure A.39: Case 1: Plot of commanded surge thrust.

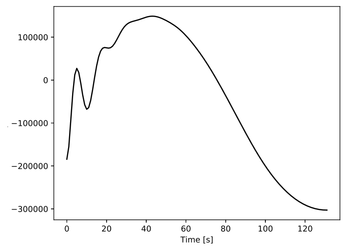

Figure A.40: Case 1: Plot of commanded sway thrust.

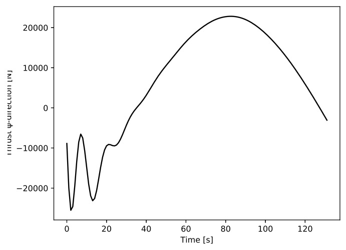

Figure A.41: Case 1: Plot of commanded yaw-moment.

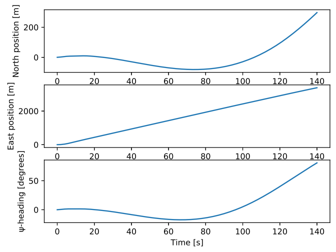

Figure A.42: Case 2: Plot of North-, East- and ψ-position.

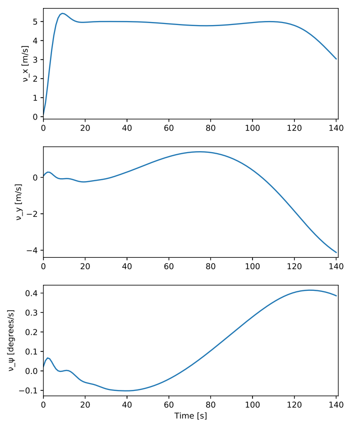

Figure A.43: Case 2: Plot of velocities in each direction with respect to the vessel coordinates.

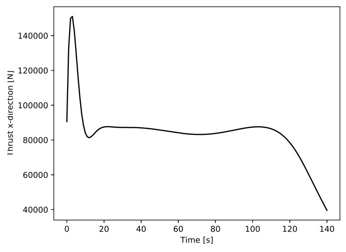

Figure A.44: Case 2: Plot of commanded surge thrust.

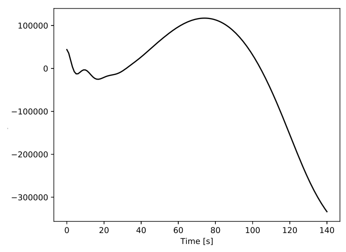

Figure A.45: Case 2: Plot of commanded sway thrust.

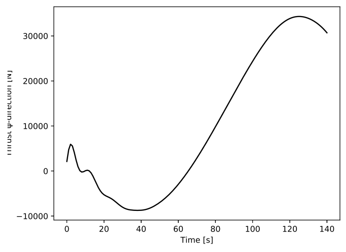

Figure A.46: Case 2: Plot of commanded yaw-moment.

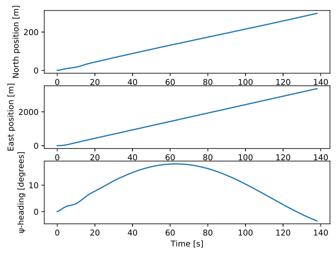

Figure A.47: Case 3: Plot of North-, East- and ψ-position.

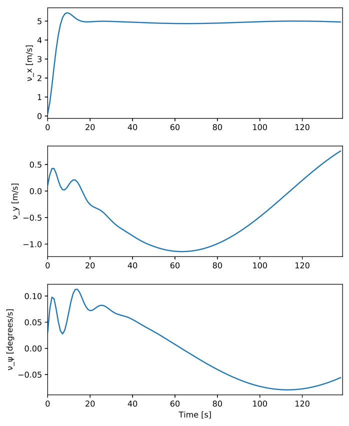

Figure A.48: Case 3: Plot of velocities in each direction with respect to the vessel coordinates.

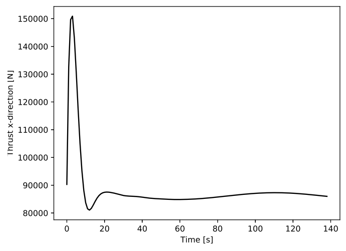

Figure A.49: Case 3: Plot of commanded surge thrust.

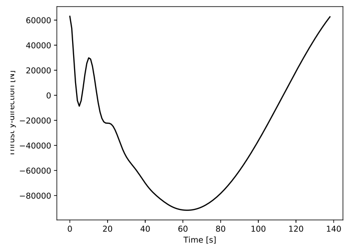

Figure A.50: Case 3: Plot of commanded sway thrust.

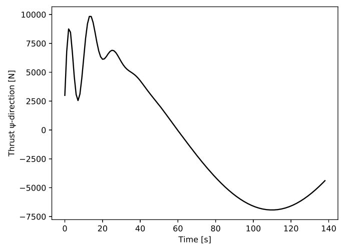

Figure A.51: Case 3: Plot of commanded yaw-moment.

A.8.4 Test scenario 4: Re-planning using the Digital twin

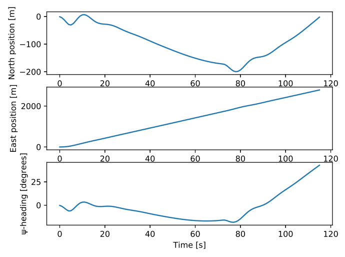

Figure A.52: Case 1: Plot of North-, East- and ψ-position.

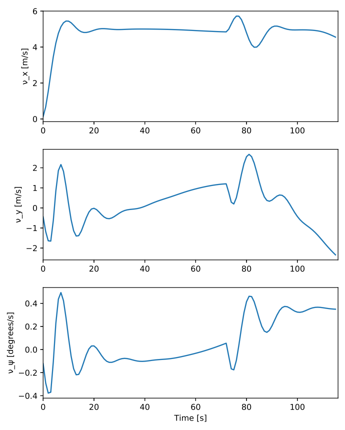

Figure A.53: Case 1: Plot of velocities in each direction with respect to the vessel coordinates.

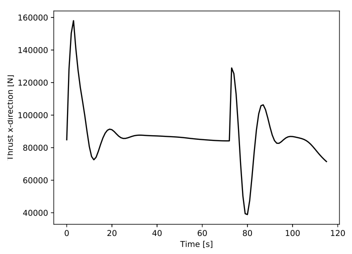

Figure A.54: Case 1: Plot of commanded surge thrust.

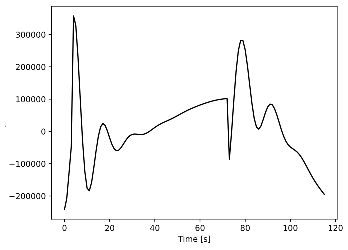

Figure A.55: Case 1: Plot of commanded sway thrust.

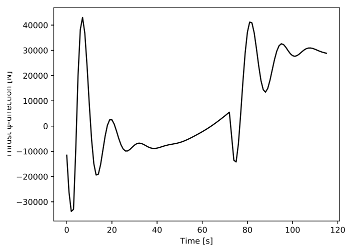

Figure A.56: Case 1: Plot of commanded yaw-moment.