Aksel Knudsen Nordstoga

AutoVoyage: Autonomous path-planning, path-generation, and path-following for autonomous ships in transit

Graduate thesis in MTMART
Supervisor: Roger Skjetne
June 2019

Graduate thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Aksel Knudsen Nordstoga

AutoVoyage: Autonomous path-planning, path-generation, and path-following for autonomous ships in transit

Graduate thesis in MTMART
Supervisor: Roger Skjetne
June 2019

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

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

MSC THESIS DESCRIPTION SHEET

Name of the candidate:	Nordstoga, Aksel Knudsen
Field of study:	Marine control engineering
Thesis title (Norwegian):	AutoVoyage: Autonom baneplanlegging, banegenerering, og banefølging for autonome skip under transit
Thesis title (English):	AutoVoyage: Autonomous path-planning, path-generation, and path-following for autonomous ships in transit

Background

Autonomous ships, especially unmanned autonomous ships, need to plan, navigate, and execute the voyaging maneuvers from departure to destination without human intervention. In particular, the autonomous ship pilot, a cognitive machine pilot, needs to sense and interpret (model) the ambient conditions with high certainty, plan a path and speed, perform navigation, and execute the plan by maneuvering the ship accordingly. If needed, the machine pilot must also perform obstacle avoidance and anti-collision and thereby replan the path and speed to reach the destination according to voyage specifications. This involves understanding of:

- the ship dynamics (inertial delays, responses to currents, wind, and propulsion, etc.),
- path feasibility requirements and safety maneuvers (crash stop, turning circle, …) for the ship,
- how to maneuver safely and optimally in waves and currents,
- under voyage path and speed decision making to satisfy local and global objectives, and
- the sensors and monitoring variables for data analysis to ensure situational awareness.

The objective of this thesis is to develop an intelligent guidance concept for an autonomous ship in transit from initial point p_0 to target p_t, where a global low-resolution path-planning method, such as A*, is combined with a dynamic method based on a bio-inspired neural network (BINN), to locally achieve a higher resolution reactive path-planning sensitive to local ambient conditions. This includes consideration of global and local partitioning of the operation area, how to integrate the global and local method in a smooth manner, how to online and recursively generate a feasible path, efficient computational representation of the neural network in code, optimization for deciding local progression to the next neuron, how to deal with currents, and finally how to do the path-following control.

The main goal is to develop a complete system for autonomous path planning, feasible and recursive path generation, and maneuvering control, with testing in simulations and the MC-Lab.

Work description

1. Provide a background and literature review with information and relevant references on:
   - Autonomous ship-voyaging system architecture and layers as designed for this problem.
   - Relevant partitioning method(s) and path planning methods for global path.
   - Use BINN for online and local path-planning for transit operation.
   - Method(s) for generation of feasible path segments.
   - Basic graph theory to represent a network of neurons with dynamic activity.
   - Relevant ship dynamical model(s).

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

2. Work together with other students working on C/S Artic Drillship (CSAD) in MC-Lab on establishing simulations model(s) and carrying out HIL- and MC-Lab testing. Formulate the problem for this case study, including description of setup, vessel and its equipment, dynamical models, operation workspace, test scenarios, specific assumptions, and a problem statement.

3. Study the Voronoi partitioning method for the global path. Perform waypoint reduction on the partitioned work space to allow for efficient path planning for AutoVoyaging from p_0 to p_t. Implement and test the method in a simulation relevant for your application. Present the results for a few relevant cases.

4. Find the optimal global path from p_0 to p_t using a suitable search algorithm, such as the A* algorithm, given the waypoints from the Voronoi diagram. Develop, implement, and test the method in a simulation system relevant for your application. Present the results for a few relevant cases.

5. From the global path-planning, a coarse (low-resolution) grid and path is obtained. With the task to move from waypoint p_k to p_{k+1}, let a local reference frame start at p_k and point towards p_{k+1}. Between these two waypoints, determine the resolution of a finer grid so that the vessel may maneuver efficiently while being sufficiently reactive to moving obstacles. Develop, implement, and test the BINN method for moving from p_k to p_{k+1} in a simulation system relevant for your application. Present the results.

6. For maneuvering in the local BINN network, develop a recursive path-generation algorithm that ensures a smooth (C^3) curve with continuous path derivatives also in the connection points. Derive also the heading reference along the path. At a specified *circle of acceptance*, the next local waypoint should be determined and a new path segment generated. Present the result by simulation.

7. Integrate the local and global path-planning and path-generation techniques, with bumpless transfer in the switching points, so that the overall path becomes a C^3 curve from p_0 to p_t. Implement and test the overall system in a simulation relevant for your application, and present the results.

8. Test the AutoVoyage system for CSAD in MC-Lab. Present the implementation and results.

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 80 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 an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this thesis description shall be included after the title page. Computer code, pictures, videos, dataseries, etc., shall be included electronically with the report.

Start date: January, 2019 **Due date:** As specified by the administration.

Supervisor: Roger Skjetne
Co-advisor(s): Jon Bjørnø, Einar Ueland, and Henrik Schmidt-Didlaukies

Trondheim, 14.03.2019

Roger Skjetne
Supervisor

Abstract

This thesis presents a complete system for autonomous path-planning, path generation and maneuvering control for a marine surface vessel.

The path-planner is divided into a global low-resolution path-planner, which plans a coarse route from point of departure to point of arrival, and a local high-resolution path-planner, which produces the intermediate waypoints on the global path and performs obstacle avoidance. For the global path, the operation area is partitioned using Voronoi diagrams, due to its low computational cost and built-in clearance to obstacles. A* search is then used to find in terms of waypoints the shortest route through the partitioning. Excess waypoints are then removed from the path to reduce heading changes and path length.

A bio-inspired neural network approach for real-time trajectory generation is used to implement the local path-planner. The approach is based on a dynamic activity landscape representation of the environment that the vessel operates in. By associating each cell in a grid decomposition with a neuron in the neural network architecture, the environment is translated into a dynamic activity landscape where the target locations become peaks and obstacle locations become valleys. The optimal path is then found by following a steepest gradient ascent rule, until the peak of the activity landscape is reached.

The local and global path-planning is integrated using hybrid path parametrization, such that the total path from point of departure to point of arrival becomes a smooth curve, ensuring bumpless transfer where path segments are connected. The control problem, which is formulated as a maneuvering problem, is to follow this path with a given speed assignment. This is achieved using a backstepping controller design.

The experimental platform used in this thesis is C/S Artic Drillship, which is a 1:90 scale model of an Artic drillship designed by Inocean. Testing is carried out at the Marine Cybernetics Laboratory at NTNU.

The system has been validated both through simulations and experiments, which yielded good results.

Sammendrag

Denne avhandlingen presenterer et komplett system for autonom baneplanlegging, banegenerering og banefølging for et marint overflateskip.

Baneplanleggeren er delt inn i en global planlegger med lav oppløsning, som planlegger en grov rute fra startposisjon til målposisjon, og en lokal baneplanlegger som produserer mellomliggende veipunkter og utfører kollisjonsunngåelse. For den globale banen er operasjonsområdet partisjonert ved hjelp av Voronoi-diagrammer, på grunn av metodens effektivitet og innebygde klaring til hindringer. Den optimale banen gjennom partisjoneringen blir så funnet ved hjelp av søkealgoritmen A*, og raffineres slik at den bare inneholder nødvendige veipunkter.

En tilnærming basert på biologisk inspirerte nevrale nettverk for sanntids-baneplanlegging brukes til å implementere den lokale baneplanleggeren. Metoden er basert på en dynamisk aktivitet landskapsrepresentasjon av miljøet som fartøyet opererer i. Ved å knytte hver celle i en partisjonering med et nevron i det nevrale nettverket, blir miljøet oversatt til et dynamisk aktivitetslandskap der målposisjoner blir topper og hindringer blir daler. Den optimale banen gjennom partisjoneringen blir funnet ved å følge den dynamiske aktiviteten til toppen av aktivitetslandskapet er nådd.

Den lokale og globale baneplanleggingen integreres ved hjelp av en metode for hybrid baneparametrisering, slik at den totale banen fra start til slutt blir en glatt kurve. Dette sikrer gode overganger mellom banesegmentene som utgjør den totale banen. Kontrollproblemet, som er formulert som et manøvreringsproblem, er å følge denne banen med en gitt hastighet. Dette oppnås ved hjelp av en kontroller designet med backstepping.

Den eksperimentelle plattformen brukt i denne oppgaven er modellskipet C/S Artic Drillship, som er en 1:90 skala modell av et arktisk drillskip designet av Inocean. Tester er utført på Marine Cybernetics Laboratory, NTNU.

Systemet har blitt validert både gjennom simuleringer og eksperimenter, som viste gode resultater.

Acknowledgments

I have received much help from other people during my thesis work, and I would like to acknowledge them for their contributions.

First, I want to thank my supervisor, Professor Roger Skjetne, for help and guidance on all parts of the project, and for providing me with an interesting subject.

I would also like to thank my co-advisors Jon Bjørnø, Henrik Schmidt-Didlaukies and Einar Ueland. Special thanks to Jon Bjørnø for a lot of help with issues related to CSAD and MCLab. Torgeir Wahl has also been a great help when I have faced technical issues at MCLab.

Also, thanks to Edvard Flaatten, Sondre Haug and especially Håvard Løvås, whom I have cooperated with on lab-related work.

Contents

List of Tables

List of Figures

Nomenclature

Acronyms

AAWA - Advanced Autonomous Waterbone Applications
AUV - Autonomous Underwater Vehicle
BINN - Bio-inspired Neural Network
CA - Collision Avoidance
CS - Cybership
cRIO - NI CompactRIO
DNV - Det Norske Veritas
CSAD - C/S Artic Drillship
DOF - Degrees of Freedom
DP - Dynamic Positioning
GNSS - Global Navigation Satellite System
GPS - Global Positioning system
IMU - Inertial Measurement Unit
MCLab - Marine Cybernetics Lab
NED - North-East-Down
NI - National Instruments
NTNU - Norwegian University of Science and Technology
MSc - Master of Science
P2P - Peer-to-peer
PhD - Doctor of Philosophy
PID - Proportional-integral-derivative
PC - Propulsion Control
PRM - Probabilistic Road Map
QTM - Qualisys Track Manager
RP - Route Planning
RRT - Rapidly-exploring Random Trees
SA - Situational Awareness
SSD - Ship State Definition
VD - Voronoi Diagram
VRU - Vertical Reference Unit

Symbols

x_i - Dynamics of *i*th neuron
α_i - Virtual control i
A - Passive decay rate
b - Bias term
B - Upper bound neural activity
β - Sideslip angle
C - Coriolis and centripetal term
D - Upper bound neural activity / Linear damping matrix
I - Identity matrix
G - Graph data structure
V - Vertex
E - Edge
$f(a)$ - Function of a
i - Path segment
I_i - Input to *i*th neuron
η - Generalized position vector in {n}
ν - Generalized velocity vector in {b}
K_i - Gain matrix
κ - Gain
M - Mass/inertia matrix
w_{ij} - Connection weight
d_{ij} - Euclidean distance
$p(s)$ - Parametrization of path
p_k - Global waypoint
ρ_i - Tuning function i
q_i - Position in state space
λ - Weighting parameter / Tuning parameter
r - Yaw rate
s - Path parameter
ψ - Yaw angle
τ - Force
v_s - Speed assignment
R - Radius circle of acceptance
R, J, T - Transformation matrices
T - Bias time constant

V - Lyapunov function
X_w, Y_w - Size of local operation area

Chapter 1

Introduction

1.1 Motivation

In recent years research on autonomous vehicles has received a lot of attention, and the level of autonomy in ships is steadily increasing.

The potential benefits from autonomous ships are many. By increasing autonomy, one can reduce accidents caused by human errors, which is a major cause of accidents at sea. Reduced crewing results in lower operation costs for ship owners, and it also allows for ship design to focus on more efficient use of space. Less human intervention is also a benefit in itself, as there is less danger for human life.

In order for an autonomous ship to be able to operate safely and reliable, it has to be able to sense and interpret its surroundings, and use that information to plan a path and speed and execute the plan accordingly. The system must also be able to perform obstacle avoidance and anti-collision when encountering unexpected obstacles (e.g. other ships), that is, re-plan the path whilst keeping the final destination in mind.

This motivates the development of an intelligent guidance system that is able to execute a ship voyage from A to B. Such a system needs an efficient partitioning of the operation area, a path planner that is able to plan a safe and efficient route, and it has to be able to generate and follow paths that are feasible for the vessel. The system also has to be computationally efficient, to respect the hardware limitations on-board.

1.2 Objectives of the thesis

The superior objective of this thesis is to develop a complete system for autonomous path-planning, path generation and maneuvering control. This is to be achieved by

1. Performing a literature review on path-planning, path generation, and other relevant topics
2. Develop a global path-planner, which plans a coarse route from point of departure to point of arrival
3. Use a bio-inspired neural network (BINN) approach to develop a high-resolution local path-planner
4. Develop a recursive path generation algorithm and integrate the local and global path-planning techniques
5. Validate the complete system

1.3 Scope and delimitations

The thesis aims at developing a system that is feasible not only for simulations but also for real-world applications. However, some simplifications and assumptions are made:

- This thesis considers a low speed, fully actuated DP vessel
- It is assumed that the only environmental load present is current
- It is assumed that information about obstacles are made available by maps and on-board sensors
- In real-world applications, there are regulations that dictate how one should perform obstacle avoidance maneuvers (COLREGs). The algorithms developed here are not made to be COLREG compliant.

1.4 Contributions and thesis outline

The contribution of this thesis is the development of a computationally efficient complete system for autonomous path-planning, path generation and maneuvering control. Established methods for global path-planning are integrated with a bio-inspired neural network approach to local path-planning to achieve safe and efficient paths, which are made feasible using hybrid path parametrization. The system has been validated both through simulations and experimental results.

The outline of the thesis is as follows:

Chapter 2 presents relevant background information on autonomous system architecture, partitioning and path-planning methods, the BINN approach to path-planning, path generation and graph theory

Chapter 3 presents the dynamic models used for simulation and controller design, and the problem statement

Chapter 4 presents the experimental setup at the marine cybernetics lab that is used to validate the system

Chapter 5 explains for the global path-planner, how the operation area is partitioned and how the optimal path through the resulting partitioning is found

Chapter 6 explains how the BINN approach is used to create a local path-planner sensitive to ambient conditions

Chapter 7 explains how feasible paths are generated using hybrid path parametrization

Chapter 8 presents the observer, guidance and controller designs

Chapter 9 presents the simulation and experimental results, and a discussion of these results

Chapter 10 gives the final conclusions and suggestions for further work

Chapter 2

Background

2.1 Autonomous system architecture and layers

2.1.1 System layers

In Sørensen (2018), it is suggested to divide the control structure for a marine control system into three layers: the mission layer, the guidance and optimization layer, and the control execution layer.

The control execution layer is divided into a high-level plant control and a low-level actuator control. The plant control receives commands from the guidance and optimization layer, which again receives setpoints from the mission layer. This hierarchy is shown in Fig. 2.1. The mission layer plans and re-plans the mission according to the mission objectives. In a conventional system, the mission layer receives it commands from a human pilot. An autonomous system however, should be able to carry out this planning by itself, using information about the objective and the environment as the operation goes on (Sørensen, 2018). For a ship to operate autonomously, the mission and guidance and optimization layers must be automated, which will add new requirements to the system architecture on-board vessels.

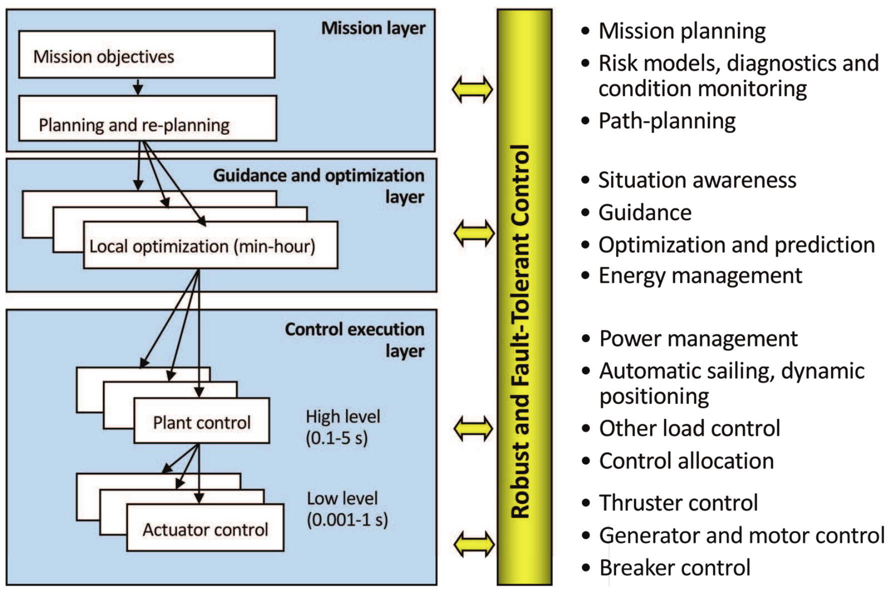


Figure 2.1: Control structure. Courtesy: Sørensen (2018).

2.1.2 System architecture

Advanced Autonomous Waterborne Applications (AAWA) was a finnish funded project that in 2015 sought to, among other things, map the technical specifications needed to realize the "next generation of advanced ship solutions" (AAWA, 2016). It was a collaboration between universities and large players in the maritime cluster, like DNV GL and Rolls-Royce. They looked at already existing system architecture for other autonomous vehicles, like self-driving cars, to investigate how they could be applied to autonomous marine navigation. The proposed system architecture is shown in Fig. 2.2.

Since a fully autonomous vessel should be capable of operating without any crewing, it has to have in place a system that can control the vessel's position and attitude based on the decision-making modules. The proposed system architecture therefore contains a module for dynamic positioning. In Sørensen (2018), a dynamic positioning (DP) vessel is defined as a vessel "that maintains its position and heading (fixed location or pre-determined track) exclusively by means of active thrusters". In other words the actuators (thrusters) are used to control the ship movement in in the degrees of freedom one wants to control. To per-

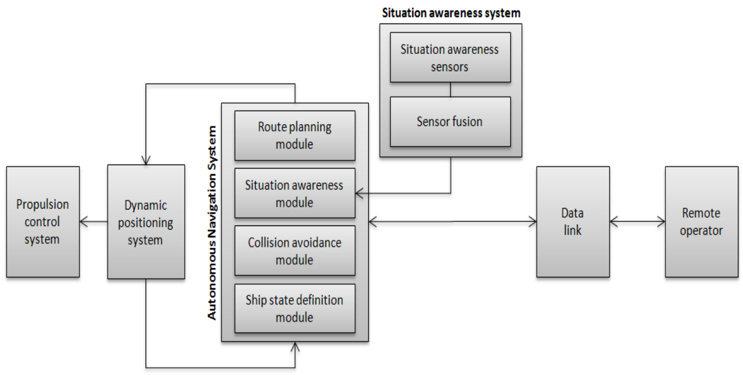

Situation awareness system
Situation awareness sensors
Sensor fusion
Autonomous Navigation System
Route planning module
Situation awareness module
Collision avoidance module
Ship state definition module
Propulsion control system
Dynamic positioning system
Data link
Remote operator

Figure 2.2: Autonomous system architecture. Courtesy: AAWA (2016).

form the positioning the DP system is dependant on information from various sensors and a global positioning reference like the Global Navigation Satellite System (GNSS).

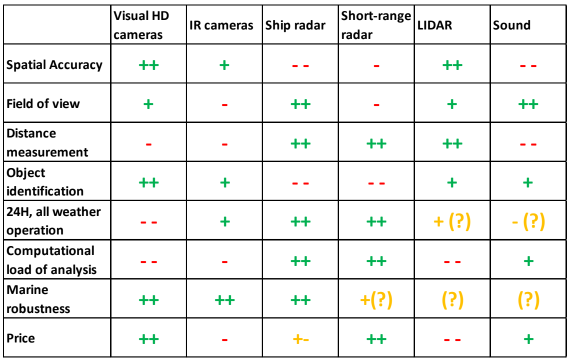
According to Sørensen (2018), a minimum configuration for a DP system is using a position reference system together with one gyro compass, one vertical reference unit (VRU) and one wind sensor. The gyro compass measures the heading of the vessel, the VRU measures the vessel heave, roll and pitch motions and the wind sensor measures the velocity and direction of the wind. However, because of a decrease in cost the trend today is towards the use of Inertial Measurement Units (IMUs) integrated with a satellite navigation system (Sørensen, 2018). According to Fossen (2011), the measurements available from a typical IMU are three-axes rate gyros, accelerometers and magnetometers. In theory these could be integrated to give attitude and position, but in practice the measurements will drift due to sensure bias, misalignments and temperature variations (Fossen, 2011). The drift is removed by integration with GNSS as a state observer.

When the DP module has produced a desired thrust in surge, sway and yaw, the Propulsion Control (PC) module calculates the corresponding force and direction of the thrusters, which the low-level thruster controllers uses to controll the propeller speed, pitch, torque and power (Sørensen, 2018).

The Route Planning (RP) module is a software module that uses available electronic navigational charts to a plan the route from start position to goal position. The route consists of waypoints, headings and speed for the ship (AAWA, 2016). It does not plan the route in real-time, as maneuvers to avoid non-static obstacles are carried out by the collision avoidance (CA) module.

The CA module has two responsibilities, assessing the risk of a situation, and if needed, make the ship deviate from the planned course to navigate safely past obstacles. From the DP module, the CA module can get the area in which the ship is actually able to maneuver to, setting boundaries on how the vessel can act to avoid collision (AAWA, 2016). The CA module receives its local real-time map from the Situational Awareness (SA) module.

The performance of the autonomous navigation system depends heavily on how accurate the system is able to represent the vessel surroundings. It is evident that the equipment used to map the surroundings must be very reliable, because deciding where to go next based on a faulty representation of the environment puts both human life and equipment at risk. The system should also be robust enough to handle demanding weather conditions and be able to function at night and daytime. In order to achieve this, the SA module should fuse the signals of different sensor types, as there are strengths and weaknesses associated with each type. These are summarized in Fig. 2.3.



	Visual HD cameras	IR cameras	Ship radar	Short-range radar	LIDAR	Sound
Spatial Accuracy	++	+	- -	-	++	- -
Field of view	+	-	++	-	+	++
Distance measurement	-	-	++	++	++	- -
Object identification	++	+	- -	- -	+	+
24H, all weather operation	- -	+	++	++	+ (?)	- (?)
Computational load of analysis	- -	-	++	++	- -	+
Marine robustness	++	++	++	+(?)	(?)	(?)
Price	++	-	+-	++	- -	+

Figure 2.3: Comparison of SA sensor types. Courtesy: AAWA (2016).

It is suggested in (AAWA, 2016) that a feasible solution for marine SA is a combination of radar technology and visual sensors, including infrared cameras. Visual spectrum cameras provide high spatial resolution for use in object detection. They can however, not be used in the dark and will struggle under demanding weather conditions. Long-Wave Infrared (LWIR) cameras detect IR radiation, and can therefore be used to detect objects in total darkness. The performance of LWIR is however also degraded by weather conditions. The camera sensors are therefore combined with radar technology, which copes better in bad weather and can provide accurate information about distance to objects. Lidars are also mentioned as a valuable addition to SA, but the technology is seen as to expensive to be included in the proposed system architecture.

2.2 Partitioning methods and path-planning methods for global path

Global path-planning is the problem of finding a collision-free path from point of departure to the desired destination, in a static environment where the obstacles are known prior to departure. In robotics, this has been a focus of research for decades. Šeda (2007) presents a comparison of two of the basic types of motion planning, the cell decomposition method and the roadmap method.

Cell decomposition is the process of dividing the obstacle-free space into a set of connected regions called cells. In its most basic form, cell decomposition divides the operation area into square cells. Finding the path from start to goal is then a matter of finding cells that share borders and searching for a path. Cellular decomposition does however suffer from the drawbacks of limited granularity and combinatorial explosion (Šeda, 2007). If the size of the cells in the decomposition is small, then the number of possible paths increases very fast with the size of the operation area, making it time consuming to find the optimal solution. In large environments, it can therefore be difficult to create a cell decomposition with the required resolution for path-planning.

Roadmap methods do not suffer from these drawbacks (Šeda, 2007). If we denote the part of the operation area that is not inside any obstacles C_{free}, the roadmap method captures the connectivity of this free space by forming a network of one-dimensional curves (Latombe, 1991). Edges are added between pairs of vertices in C_{free} that can be connected by a straight line which does not intersect any obstacles, which results in a roadmap of collision-free paths that can be used in path-planning. Shortest-path algorithms, like Dijkstra's algorithm or A* (A-Star) search can then be used to retrieve the shortest path through the roadmap.

Several methods exist to construct the initial roadmap, one being Voronoi diagrams. In Voronoi diagrams, the contour points of the obstacles in the environment are used to produce paths that are of maximum distance to the obstacles. The vertices of the roadmap are the points where three of these paths intersect. Fig. 2.4 shows a simple Voronoi diagram.

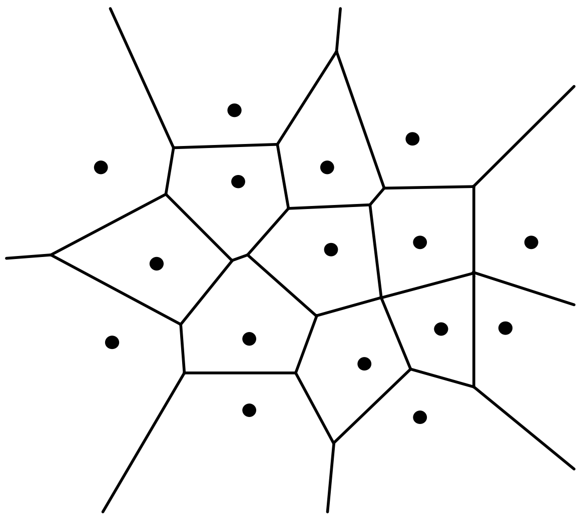


Figure 2.4: Voronoi diagram. Courtesy: Šeda (2007).

A more formal introduction to Voronoi diagrams is given in Chapter 5.1.1.

Other roadmap methods include the Probabilistic Road Map (PRM) and the Rapidly-exploring Random Trees (RRT) method. To produce a PRM, random nodes are generated in the operation area. They are then tested to see if they belong to C_{free}, and if they do, a local planner is applied to connect it to neighbouring nodes. This process is continued until the roadmap has the required resolution. RRTs are also produced by generating random nodes in the operation area, and then connecting the new node to the closest existing node, making it biased towards exploration of parts of the operation area with a lower density of nodes. Hvamb (2015) investigated the use of all three mentioned roadmap methods for path-planning for marine vehicles, where all three methods yielded similar results in terms of path length.

2.3 The Bio-Inspired Neural Network approach to path-planning

The use of a bio-inspired neural network approach for dynamic collision-free trajectory generation was first proposed in Yang and Meng (1998). The neural network architecture is a discrete topologically organized map, where the state space can be the Cartesian workspace, letting the location of the ith neuron at the grid represent a position in the workspace. The activity of each neuron is characterized by the shunting equation Eq. 6.1. This translates environment information about the operation area into a dynamic activity landscape, where the peak of the landscape represents the target destination. The activity of the target(s) is propagated through the landscape through lateral connections between neurons. When in a position p_{n-1}, corresponding to the position of a neuron, the next position p_n is in (Yang and Meng, 2001) chosen as

$$p_n \Leftarrow x_{p_n} = max\{x_j, j = 1, 2, ..., k\} \tag{2.1}$$

where x is the neuron activity level and k is the number of neighbouring neurons. This is repeated until the target neuron is reached.

In Scibilia et al. (2012), a similar approach is used for transit operation and complete area coverage for an AUV. A rectangular operation area is divided into circles with a radius adjusted according to the AUV speed, which are distributed optimally on the workspace to achieve minimum repeated coverage. The circles are then chosen as neurons for the neural network architecture. Paths are computed using Dubins theory to ensure that the kinematic constraints of the AUV are taken into account. To attenuate heading changes, the next position when AUV is in position p_q with heading angle θ_q is chosen as

$$p_n \Leftarrow_{x_j : p_j \in neig_{r_0}(p_q)} \left\{ \left(1 - \frac{diff(\theta_q, \theta_j)}{\pi} \right) \lambda x_j + (1 - \lambda) x_j \right\} \tag{2.2}$$

where $0 \leq \lambda \leq 1$ is a weighting parameter used to tune the weighting mechanism. For transit operation, the turning radius from which the operation area is partitioned, is adjusted at each step of the algorithm according to the distance to the closest obstacle or target, to allow for faster transit speeds and smoother trajectories (Scibilia et al., 2012).

Ni et al. (2017) proposed a dynamic BINN approach, where the AUV is considered the core of a three-dimensional BINN, where the size of the BINN is set based on the sensor range. The BINN is then set to move together with the AUV. This is done to reduce the computation in large environments. Since the final target can be farther away from the AUV than the sensor detection range, Ni et al. (2017) also propose a virtual target concept. The virtual target is located at the edge of the BINN, as close to the real target as possible, without any obstacles on the straight path from the AUV to the virtual target.

In Huang et al. (2016), a BINN algorithm is used for real-time path planning of a group of hunter AUVs, which objective is to surround and trap a moving target (evader).

2.4 Generation of feasible path segments

The path-planning module produces a route given as a series of waypoints. The next problem is then to generate a feasible path, that is, a path that respects the vessel's dynamic constraints. It is therefore desirable to generate a smooth path, to avoid jumps in the control loop and achieve bumpless transfer between waypoints. Lekkas (2014) distinguishes between two categories of connecting waypoints to generate a path:

- Combining straight lines and arc segments
- Using splines

Dubins (1957) showed that the shortest path from an initial position and heading to a terminal position and heading is given by joining circular arcs with straight line segments. This method is often used in path generation due to its simplicity. However, the drawback of this method is a jump in the desired yaw rate r_d when transitioning from a straight line ($r_d = 0$) to a circular arc ($r_d = constant$) (Fossen, 2011). In Fraichard and Scheuer (2004), it is suggested to solve this problem by using clothoids as transition curves between the straight lines and arc segments. Lekkas et al. (2013) show that also Fermat's spirals can be used as transition curves, while being less computational intensive than clothoids. In Candeloro et al. (2017), Fermat's spirals are used to connect straight line segments in a Voronoi partitioning.

The second category involves constructing a curve through a set of waypoints, using interpolation methods. Many methods exist to solve this problem. Cubic Hermite spline interpolation can be used to parametrize the curve, but gives discontinuous curvature at the waypoint locations (Lekkas, 2014). Cubic spline interpolation gives continuous curvature, but the generated path will have more oscillations than the cubic Hermite spline for unsmooth data (Fossen, 2011). In Fossen (2011), it is shown how the path generation can be solved as a nonlinear constrained optimization problem, where the object function based on time and energy consumption is minimized under the constraints of speed and acceleration limits of the vessel. The drawback of this method is that it is harder to solve numerically than the previously mentioned methods (Fossen, 2011).

2.5 Graph theory

In order for a path-planner to be able to find a path through the partitioning, it is necessary to define for every vertex, which vertices are reachable. This requires a flexible way of describing the vertices of the partitioning and the relationship between them. A suitable data structure to represent the neural network model and the roadmap partitioning are graphs.

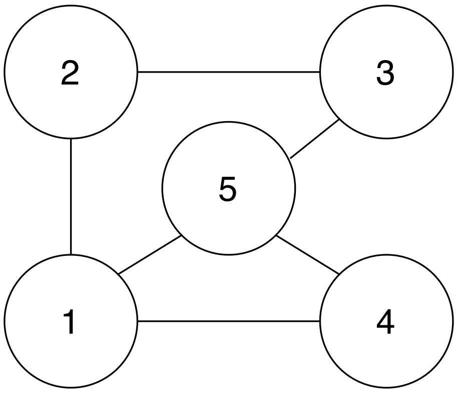
Formally, a graph G is an ordered pair

$$G = (V, E), \tag{2.3}$$

where V is a set of vertices (or nodes), and E is a set of edges. The edges connect the vertices and represent the relationship between them.

The *order* N of a graph G is the number of vertices V, and its *size* is the number of edges E. Two vertices, V_i and V_j, are said to be *connected*, if there exists a path of edges between them. If only one edge is needed to get from one vertex to another, then the two vertices are said to be *adjacent*. The *neighbourhood* of a vertex is the set of *adjacent* vertices.

The properties of a graph can be edited to suit the application. Fig. 2.5 shows an undirected and unweighted graph, meaning that edges have no specific orientation and that the "cost" of every edge is equal. If desired however, we can define weights for the edges. This is useful when one wants to calculate the costs of traversing the graph using different paths.



$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Figure 2.5: Example of a graph and its adjacency matrix

A graph can be represented using an adjacency matrix. Fig. 2.5 shows an example of an adjacency matrix set to represent the undirected graph to the left. Element (i, j) in the adjacency matrix is set to 1 if vertex V_i and V_j are adjacent, and 0 if they are not. It follows that for undirected graphs, the adjacency matrix will be symmetric.

Chapter 3

Problem formulation

This chapter starts with presenting a 6 degree of freedom (DOF) dynamic model used to simulate a marine craft, before showing a simplified 3 DOF DP model that is used for controller design. The problem statement of the thesis is then given by dividing the main objectives into three (interrelated) problems: the path-planning problem, the path generation problem and finally the control problem, which is formulated as a maneuvering problem.

3.1 Dynamic models

3.1.1 Simulation model

Using the notation of SNAME (1950), a marine craft operating in 6 DOFs can be described by six motion variables. The first mode is (x, y, z), which is referred to as *surge*, *sway* and *heave* and describe the vessel position in a three-dimensional space. The second mode is (ϕ, θ, ψ), referred to as *roll*, *pitch* and *yaw* which describe the vessel orientation.

To express the kinematics of a marine craft we introduce two reference frames, the North-East-Down (NED) frame and the body-fixed reference frame.

- The NED frame, denoted {n}, is usually defined as the tangent plane on the earth surface moving with the craft, with its x axis pointing towards true north, its y axis pointing east and its z-axis pointing downwards (Fossen, 2011). By assuming that the vessel operates at approximately constant longitude and latitude, {n} can be approximated as an inertial frame, where Newton's laws apply.

- The body-fixed reference frame, denoted {b}, is a moving reference frame fixed to the craft. This is the reference frame in which the linear and angular velocities of the craft is expressed in. It has its x-axis pointing from aft to fore, its y-axis directed starboards and its z-axis directed downwards.

The linear and angular velocities in the {b} frame are related to the six motion variables through

$$\dot{\eta} = J_{\Theta}(\eta)\nu \tag{3.1}$$

where $\eta = [x, y, z, \phi, \theta, \psi]^T$ and $\nu = [u, v, w, p, q, r]^T$ are the body-fixed velocities as illustrated in Fig. 3.1.

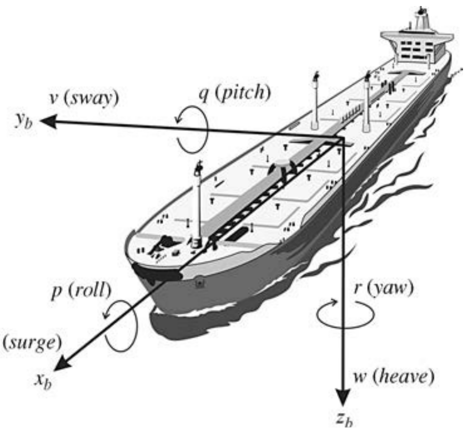


Figure 3.1: 6 DOF velocities in the body-fixed reference frame. Courtesy: Fossen (2011).

The transformation matrix $J_{\Theta}(\eta)$ is expressed as

$$J_{\Theta}(\eta) = \begin{bmatrix} R_b^n(\Theta_{nb}) & 0_{3x3} \\ 0_{3x3} & T_{\Theta}(\Theta_{nb}) \end{bmatrix} \tag{3.2}$$

where the linear velocity transformation matrix $R_b^n(\Theta_n b)$ is given by

$$R_b^n(\Theta_{nb}) = R_{x,\phi}R_{y,\theta}R_{z,\psi} = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{3.3}$$

and the angular velocity transformation $T_\Theta(\Theta_{nb})$ is given by

$$T_\Theta(\Theta_{nb}) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \tag{3.4}$$

where $s = sin(\cdot)$, $c = cos(\cdot)$ and $t = tan(\cdot)$.

The kinetic model is given by (Fossen, 2005)

$$M\dot{\nu}_r + C_{RB}(\nu)\nu + C_A(\nu_r)\nu_r + D(\nu_r)\nu_r + \mu + G\eta = \tau_{waves} + \tau_{wind} + symol\tau \tag{3.5}$$

where M is the system inertia matrix, $C_{RB}(\nu)\nu$ and $C_A(\nu_r)$ are the coriolis and centripetal matrices, $\dot{\nu}_r$ is the relative velocity between the craft and the water, $D(\nu_r)$ is the damping matrix, μ is the term that captures the fluid memory effects, G is the linearized restoring forces and moments matrix, τ_{waves}, $\tau_{current}$ and τ_{wind} is the environmental forces and τ is the propulsion forces. The model is derived using nonlinear unified theory for maneuvering and seakeeping. For a complete derivation of the dynamic model, the reader is referred to Fossen (2005).

3.1.2 Control design model

For control design a simplified representation of the vessel dynamics is sufficient. We consider a fully actuated low speed DP vessel. By assuming that the ship is longitudinally and laterally metacentrically stable for small amplitudes of $\phi = \theta = \dot{\phi} = \dot{\theta} \approx 0$, and that the vessel floats with $z \approx 0$ in mean, we can neglect the dynamics associated with heave, roll and pitch. For low speed applications, a linearization about $\nu = 0$ then gives the 3 DOF control design model (Skjetne, 2019)

$$\dot{\eta} = R(\psi)\nu \tag{3.6}$$

$$M\dot{\nu} + D\nu = \tau + R(\psi)^T b \tag{3.7}$$

where $\eta = (p, \psi) \in \mathbb{R}^2 \times \mathcal{S}_1$ is the ship pose, $\nu = (u, v, r) \in \mathbb{R}^3$ is the body-fixed linear and angular velocity vector,

$$R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, R(\psi)^T R(\psi) = R(\psi)R(\psi)^T = I \tag{3.8}$$

is the matrix that gives the rotation about the z axis,

$$M = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix} = M^T > 0 \tag{3.9}$$

is the system inertia matrix,

$$D = \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & d_{23} \\ 0 & d_{32} & d_{33} \end{bmatrix} = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} > 0 \tag{3.10}$$

is the linear damping matrix and b is a slowly varying bias term capturing unmodeled dynamics.

3.2 Problem statement

3.2.1 Path-planning

The path-planning function is to determine a set of waypoints from point of departure, p_0, to the final destination p_t, so that the resulting path is efficient and includes sufficient clearance to obstacles.

The path-planning problem is divided into two parts, one global and one local. The global path-planner takes as input obstacle information from a pre-existing map, to produce a coarse route from p_0 to p_t, given as a series of waypoints. In the local path-planner, it assumed that information about previously unknown obstacles, static and dynamic, are made available by on-board sensors. The local path-planner uses this information to produce the intermediate waypoints between the global waypoints. These waypoints are produced online in a stepwise manner, to allow for reactive obstacle avoidance using the newest available environment information.

3.2.2 Path generation

The path-planning module provides the path generation with waypoints. The path generation problem, is then to generate a path through these waypoints.

In order to ensure bumpless transfer between waypoints, we require that the path be $\mathcal{C}^3$. The objective is then to construct a sufficiently differentiable curve through all the waypoints from p_0 to p_t, so that the resulting path is a smooth curve with continuous path derivatives in the connection points. Letting $s \in \mathbb{R}$ be a continuous path variable, this desired path can be expressed as

$$p_d(s) = col(x_d(s), y_d(s)) \tag{3.11}$$

3.2.3 Control objective

The control objective is formulated as a maneuvering problem, which is comprised of two tasks as defined Skjetne (2005):

1. The geometric task: For any continuous function $s(t)$, force the output η to converge to the desired path $\eta_d(s)$,

$$\lim_{t\to\infty} |\eta(t) - \eta_d(s(t))| = 0 \tag{3.12}$$

2. The dynamic task: We also want to satisfy a dynamic behaviour along the path, in the form of a speed assignment: Force the path speed $\dot{s}$ to a desired speed $v_s(s,t)$,
$$\lim_{t\to\infty} |\dot{s}(t) - v_s(s(t),t)| = 0 \tag{3.13}$$

From Skjetne (2005) we have that a speed assignment is given by

$$v_s(t,s) = \frac{u_d(t)}{|p_d^s(s)|} \tag{3.14}$$

From the path generation module we have the desired point on the path, $p_d : \mathbb{R} \to \mathbb{R}^2$, parametrized by the continuous path variable $s \in \mathbb{R}$. Additionally, we propose a desired heading tangent to the desired path (Skjetne, 2005)

$$\psi_d(s) = \angle p_d^s(s) = atan\left(\frac{y_d^s(s)}{x_d^s(s)}\right) \tag{3.15}$$

The desired output pose now makes out a curve in the output space, that is, $\eta_d : \mathbb{R} \mapsto \mathbb{R}^2 \times \mathcal{S}_1$ where

$$\eta_d(s) := \begin{bmatrix} p_d(s) \\ \psi_d(s) \end{bmatrix},\ s \in \mathbb{R} \tag{3.16}$$

The maneuvering control objective is then to design control τ such that

$$\left.\begin{aligned} \eta(t) \to \eta_d(s(t)), \\ \dot{s}(t) \to v_s(t,s), \end{aligned}\right\} \qquad \text{as } t \to \infty \tag{3.17}$$

Chapter 4

Experimental setup

4.1 The marine cybernetics laboratory

Model scale experiments are carried out at the *Marine cybernetics laboratory* (MCLab) at the Norwegian University of Science and Technology (NTNU). MCLab is a small wave basin with dimensions L x B x D = 40 m x 6,5 m x 1,5 m. The advanced instrumentation package makes it suitable for testing of motion control systems for model-scale vessels, and it can also be used for more specialized hydrodynamic tests as it is equipped with an advanced towing carriage with precise movements. It is shown in Fig. 4.1.

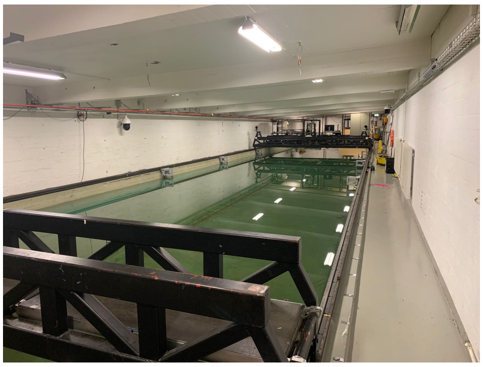


Figure 4.1: The basin at MCLab

Position measurements of the model-scale vessels are obtained using the installed

Qualisys motion capture system. Vessels are fitted with three or four reflector spheres, so that accurate position and orientation measurements can be obtained using the three Oqus infrared cameras mounted on the towing carriage (seen at the far end of the basin in Fig. 4.1). The measurements are transmitted over a P2P network to a computer running Qualisys Track Manager (QTM) software, as shown in Fig. 4.2.

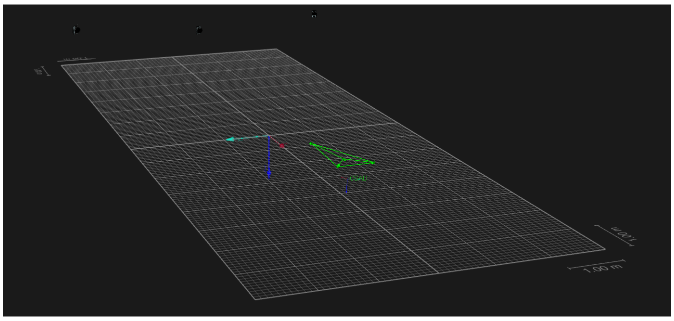


Figure 4.2: Qualisys Track Manager

4.2 C/S Artic Drillship

The target vessel used in this thesis is the C/S Artic Drillship (CSAD), shown in Fig. 4.3. It is a 1:90 scale model of the Inocean Cat I Drillship. It was initially developed in Bjørnø (2016) for use in research on thruster-assisted position mooring, and is now a platform for testing of motion control systems at MCLab.

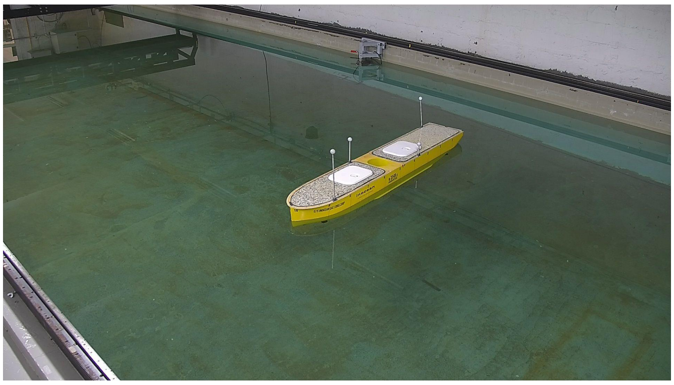


Figure 4.3: C/S Artic Drillship

The vessel is equipped with 6 azimuth thrusters (3 fore and 3 aft). The model runs real-time control systems programmed in Matlab/Simulink using NI VeriStand and a National Instrument CompactRIO (cRIO) embedded controller. Experimental results are made available in real-time using Labview on a host computer at MCLab. The hull is constructed by carbon fiber and a casted frame stiffens the hull (Bjørnø, 2016). The model dimensions are given in Table 4.1.

Table 4.1: CSAD dimensions

L	2.578m
B	0.440m
D	0.211m
T	0.133m
m	127.92kg

Simulations are carried out in Matlab/SimulinkR2018b, using the fixed thrust allocation developed in Lyngstadaas (2018). The system parameters needed to use the dynamic models presented in Chapter 2 are the ones identified in Bjørnø (2016).

Chapter 5

Global path-planning

The objective of the global path-planner is to use information about the environment that is known prior to departure, to plan a coarse route from the point of departure to the final destination, that is efficient and includes sufficient clearance to obstacles. This chapter describes how the operation area is partitioned and how the optimal path through the resulting partitioning is found.

5.1 Path-planning algorithm

Using Voronoi diagrams to partition operation areas is a well proven method that has been applied both to marine and aerial operations (Candeloro et al., 2017). Therefore many implementations exist in the literature, and the procedure used in this thesis is inspired by the ones presented in Lekkas (2014), Candeloro et al. (2017) and Ørjan Grefstad (2018). First, Voronoi diagrams are used to partition the operation area and produce an obstacle-free roadmap. Then, paths that cross map borders or obstacles are removed, and the point of departure and point of arrival is connected to the roadmap. A* search is then used to search the roadmap for the shortest path. Collinear and almost collinear waypoints are then removed to shorten the path and reduce heading changes. The resulting path is then checked to see if it respects clearance constraints. If it does not, the A* search is repeated to find an alternative path. Finally, waypoints that can be omitted from the path without violating the clearance constraints are removed. The process is illustrated in Fig. 5.1.

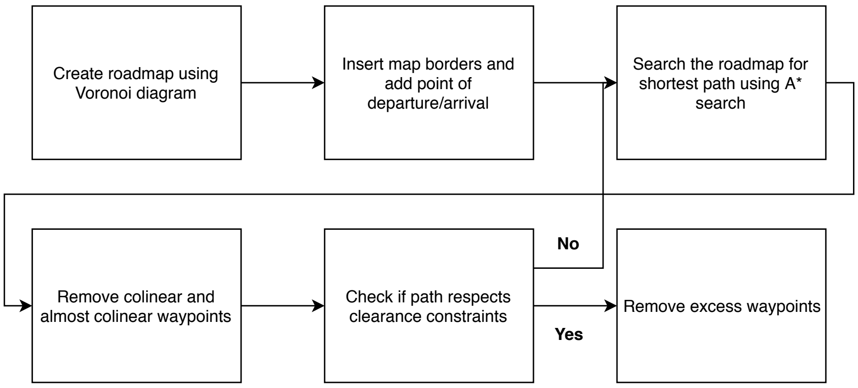

Create roadmap using Voronoi diagram
Insert map borders and add point of departure/arrival
Search the roadmap for shortest path using A* search
Remove colinear and almost colinear waypoints
Check if path respects clearance constraints
No
Yes
Remove excess waypoints

Figure 5.1: Global path-planner

Each step of the algorithm is explained in the following subchapters.

5.1.1 Partitioning the operation area using Voronoi diagrams

For large operation areas, grid partitioning with a high enough resolution to include small obstacles would be very computationally demanding. Therefore, the area is partitioned using Voronoi diagrams. Its computational cost is low, and the generated paths are of maximum distance to the obstacles in the environment.

5.1.1.1 Theory

The Voronoi partitioning method divides a two-dimensional space X into a set of regions R_k, using a set of generator points $P = \{p_1, ..., p_k\}$ and a metric function $d(\cdot)$. The result is a partitioning in which every point x contained inside a Voronoi region R_k is closer to the generator point p associated with that Voronoi region than any other generator point.

For $x \subset \mathbb{R}^2$ the metric function can be chosen as the Euclidean distance:

$$d(x,p) = \sqrt{(x_x - p_{i_x})^2 + (x_y - p_{i_y})^2} \tag{5.1}$$

The Voronoi regions are then defined as (Lekkas, 2014):

$$R_k = \{x \in X | d(x, P_k) \leq d(x, P_j) \forall j \neq k\} \tag{5.2}$$

5.1.1.2 Implementation

For use in path planning, the vertices of the obstacles in the environment and the coordinates of the map borders are used as generator points for the Voronoi diagram. The result is a roadmap of possible paths that the vessel can travel along to navigate the environment. The point of departure and point of arrival is then connected to the roadmap. Fig. 5.2 shows an example, where three simple obstacles are approximated as polygons.

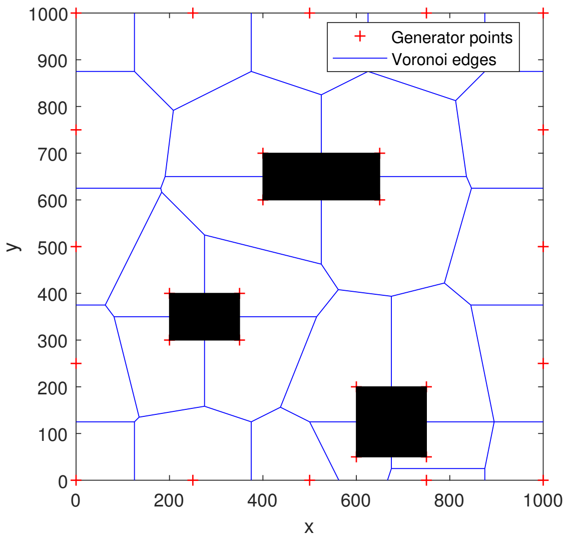


Figure 5.2: Example of a Voronoi partitioning

As can be seen from Fig. 5.2, the roadmap needs to be processed so that paths passing through obstacles or map borders are removed.

5.1.2 Finding the shortest path using A* search

5.1.2.1 Theory

A* search is an informed search strategy used to find the order in which the nodes of a graph should be traversed to produce the path has the smallest total cost from start to goal node. It evaluates nodes by using a cost function

$$f(n) = g(n) + h(n) \tag{5.3}$$

where $g(n)$ is the path cost from start node to node n, and $h(n)$ is the estimated cost of the cheapest path from node n to the goal node.

In order for A* to be guaranteed to return an optimal path, the heuristic used has to be *admissible*, meaning that it must never over-estimate the cost of getting from node n to the goal node. By using distance travelled as cost, the Euclidean distance

$$h(n) = \sqrt{(n_x - goal_x)^2 + (n_y - goal_y)^2} \tag{5.4}$$

can be used as the heuristic in path planning, since the shortest path between two points is the straight-line distance.

5.1.2.2 Implementation

A* maintains two lists, the Open list which contains nodes that need to be examined, and the Closed list which contains nodes that have been examined. At each step of the algorithm, the algorithm takes from the Open list the node with the smallest f value, which is equivalent to choosing the path that is most promising based on the heuristic h. The node is then added to the Closed list and its successors is generated. The process is repeated until the goal node is found or every node has been searched without finding a solution. The algorithm is shown in Algorithm 1 (Caltech, 2019):

Algorithm 1: Pseudo-code for A* search

```
Input  : Start node, goal node and graph adjacency matrix
Output: Optimal path from start to goal node
Initialize Open and Closed list as empty
Add start node to Open list
while Open list is not empty do
    Take node n from Open list with lowest f(n)
    Add n to Closed list
    if n is the goal node then
        return solution
    else
        Generate successors of n
        for each successor n' of n do
            Calculate g(n'), h(n') and f(n')
            if n' is on the Open list and the existing entry is as good or better then
                Discard n'
                Continue
            end
            if n' is on the Closed list and the existing entry is as good or better
              then
                Discard n'
                Continue
            end
            Remove n' from Open and Closed list
            Add n' to Open list
        end
    end
end
```

The A* algorithm can then find the shortest path from start to goal node in the Voronoi partitioning, using the coordinates of the vertices in the partitioning and the associated adjacency matrix. The path is then refined by removing collinear and almost collinear waypoints. The resulting path is given as a series of waypoints $[p_0, ..., p_t]$.

5.1.3 Clearence constraints

The path is then checked to see if has enough clearance to obstacles, adopting the approach used in Ørjan Grefstad (2018). The path and the obstacles are drawn on two separate binary images. The path is drawn as a line with a thickness equal to the width of the vessel plus the required safety margin. A test can then be done for each path segment, to see if there is any overlap between the two images. If the path segment is found to violate the clearance constraints, it is removed from the roadmap so that the A* algorithm does not consider it a possibility when searching for a new path.

Algorithm 2: Pseudo-code for testing if path respects clearance constraints

```
Input  : wayPointList, safetyMargin, obstacleContours
Output: isLegal, 1 if path respects clearance constraints
isLegal = 1
Set pathSegments to lines connecting waypoints in wayPointList
Set pathWidth equal to width of vessel plus safetyMargin
Create binaryImageObstacles from obstacleContours
for each pathSegment p of pathSegments do
    Create binaryImagePath of pathSegment p with pathWidth
    if any overlap between binaryImagePath && binaryImageObstacles then
        isLegal = 0
        return
    end
end
```

5.1.4 Waypoint reduction

The path can be further improved by omitting waypoints that can be removed without violating clearance constraints. Fig. 5.3 illustrates this: there is no point travelling via the points B and C, if the path directly from A to D respects the clearance constraints.

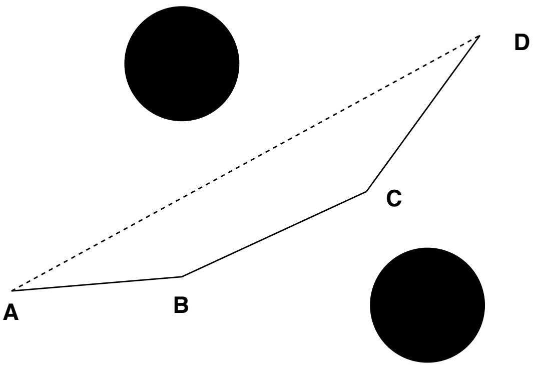


Figure 5.3: Example of how waypoint reduction can improve the path

Removing these will reduce the heading changes and shorten the path length, which in turn will reduce the fuel consumption (Candeloro et al., 2017). This is done as in Lekkas (2014) by iterating through the waypoints comprising the path, and testing if the path remains legal after removing the different waypoints, as shown in Algorithm 3.

Algorithm 3: Pseudo-code waypoint reduction

```
Input  : path in terms of waypoints
Output: path with excess waypoints removed
i = 1
for each waypoint of path do
    Q1 = path(i)
    P = path(i+1)
    Q2 = path(i+2)
    isLegal = isPathLegal(Q1, Q2) % Using Algorithm 2
    if isLegal == 1 then
        remove P from path
    else
        i = i + 1
    end
end
```

Chapter 6

Local path-planning

The local path-planner aims to move the vessel along the path given by the global path planner, whilst handling reactive obstacle avoidance. The output of the local path-planner is all the intermediate waypoints $[q_0, ..., q_t]$ between each two consecutive waypoints p_k and p_{k+1} from the global path-planner. This chapter shows how the BINN approach is used for local path-planning.

6.1 Creating the dynamic activity landscape

The operation area is partitioned into a rectangular grid. Let the position of the center of each cell be denoted q_i, then each cell at the grid is associated with a neuron and its activity level x_i. The dynamics of the *i*th neuron, where the excitatory and inhibitory inputs $[I]^+$ and $[I]^-$ arise from target and obstacle lateral connections, is expressed as a shunting differential equation

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+\right) - (D + x_i)[I_i]^- \tag{6.1}$$

where *k* is the number of neighboring neurons (neurons with lateral connections to the *i*th neuron) of the *i*th neuron, A represents the passive decay rate, B is the upper bound on neural activity, D is the lower bound on neural activity and $[I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+$ and $[I_i]^-$ is the excitatory and inhibitory inputs respectively.

The external input I_i to the *i*th neuron is defined as

$$I_i = \begin{cases} E, & \text{if there is a target} \\ -E, & \text{if there is an obstacle} \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

where $E \gg B$ is a very large positive constant. The functions $[a]^+$ and $[a]^-$ are defined as $max\{a, 0\}$ and $max\{-a, 0\}$ respectively. The connection weight w_{ij} from the *i*th to the *j*th neuron is defined as

$$w_{ij} = f(d_{ij}) \tag{6.3}$$

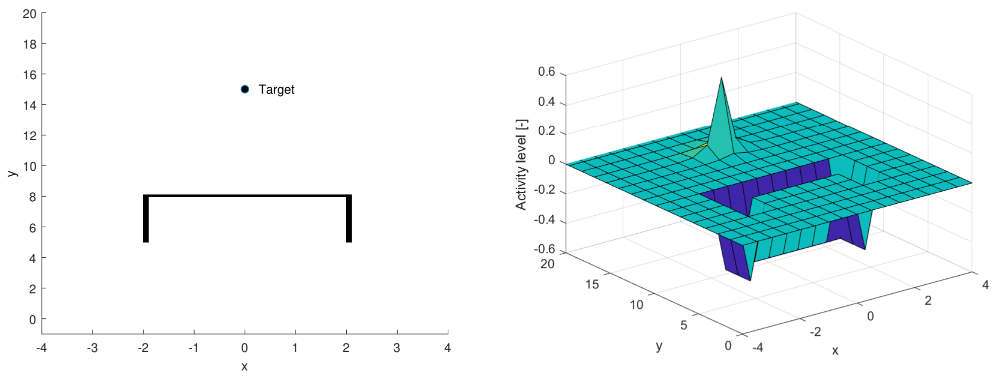
where d_{ij} is the Euclidean distance between two positions q_i and q_j in the state space. The connection weight function $f(a)$ is a monotonically decreasing function defined as

$$f(a) = \begin{cases} \mu/a & \text{if } 0 < a < r_0 \\ 0, & \text{if } a \geq r_0 \end{cases} \tag{6.4}$$

where μ and r_0 are positive constants.

The neural network characterized by 6.1 guarantees that the neural activity from the target is able to propagate through the whole state space, while negative activity from obstacle stays local only (Yang and Meng, 2001). This has the effect of targets attracting globally, while the obstacles only have effect in a small range to avoid collision. The optimal path to the target neuron is then found by climbing the activity landscape, following a steepest gradient ascent rule.

Fig. 6.1 shows an example where the environment information has been translated into an activity landscape. As can be seen, the target location becomes a peak and the obstacles become valleys in the generated landscape.



(a) Target location and obstacle **(b)** Resulting dynamic activity landscape

Figure 6.1: Dynamic activity landscape representation of the environment

6.2 Choosing the next waypoint

Let $p(t) = (x(t), y(t))$ be the current position of the vessel position and $q_n = (x_n, y_n)$ be the position of the current waypoint. When the vessel is found to be inside a *circle of acceptance* with radius R defined by

$$[x_n - x(t)]^2 + [y_n - y(t)]^2 \leq R^2, \tag{6.5}$$

of the current waypoint q_n, the waypoint is considered reached and the next waypoint needs to be chosen.

The next waypoint q_{n+1} is chosen among the neighboring nodes, as in (Scibilia et al., 2012):

$$q_{n+1} \Leftarrow max\left\{ \left(1 - \frac{diff(\theta_n, \theta_j)}{\pi}\right) \lambda x_j + (1-\lambda) x_j \right\} \tag{6.6}$$

where j is the number of neighbouring neurons, θ_n is the current vessel heading, θ_j is the heading angle of the line from q_n to q_j used as an indication of the corresponding change of direction. The weighting is introduced to attenuate heading changes and λ is used to tune the weighting mechanism.

Using the graph data structure presented in Chapter 2.5, the neighboring nodes j of q_n are defined as shown in Fig. 6.2, so that the vessel may move diagonally.

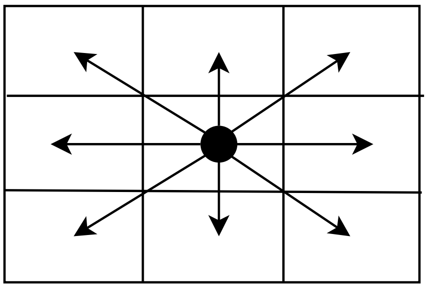


Figure 6.2: Choosing the next waypoint

6.2.1 Integration with the global path-planner

We already now describe how the local path-planner is integrated with the global-path planner, as it is relevant for how the operation area is partitioned. Between each two consecutive waypoints p_k and p_{k+1} provided by the global path-planner, a local reference frame is defined, with its y axis pointing from p_k to p_{k+1}. A rectangular operation area with length Y_w and width X_w is then defined between the waypoints as illustrated in Fig. 6.3. The transformation from coordinates in the local reference frame to $\{n\}$ coordinates are given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = p_k + R(\psi)^\top \begin{bmatrix} x_{local} \\ y_{local} \end{bmatrix} \tag{6.7}$$

where $R(\psi)$ is the two-dimensional rotation matrix

$$R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{bmatrix} \tag{6.8}$$

and ψ is defined as shown in Fig. 6.3.

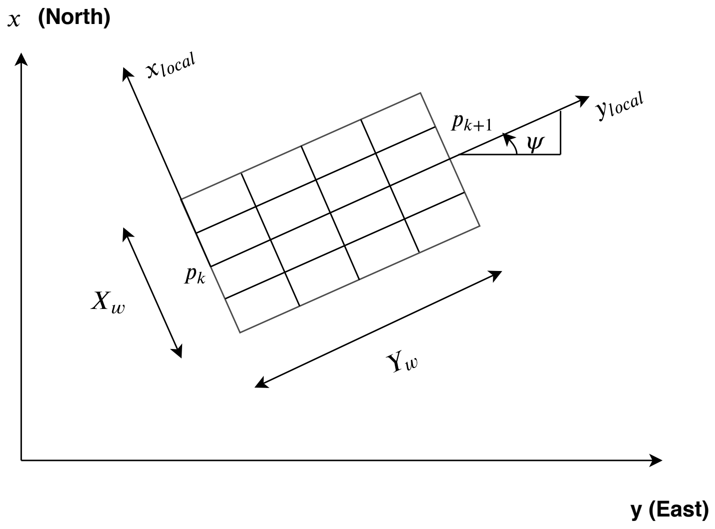


Figure 6.3: Local reference frame between p_k and p_{k+1}

6.2.2 Determining the resolution of the grid partitioning

Since the obstacles only have local effect in the BINN network, the size of each cell, or neuron, in the grid decomposition is directly related to the clearance the path will have to obstacles. Large grid sizes can produce paths with smaller curvature that are easy to track and gives good safety margins with respect to static obstacles. To large however, one can get inefficient routes and possibly miss entire solutions to the path planning problem. Fig. 6.4 shows a scenario where the two circular obstacles are seen to intersect with a neuron position on the path from the vessel position to the target. In this case, the BINN algorithm would have to find an alternative route around the obstacles, which is inefficient, assuming that the route actually was safe. On the other end, having the grid size set to small would produce paths with little clearance to obstacles.

Again, since the obstacles only have a local effect in the dynamic activity landscape, one would need a very coarse partitioning to stay clear of dynamic obstacles. It is therefore proposed to partition the operation area only with respect to clearance of static obstacles, so that the resulting paths are efficient. To stay clear of dynamic obstacles, it is suggested to instead increase the size of the representation of dynamic obstacles in the BINN, so that they are registered as occupying

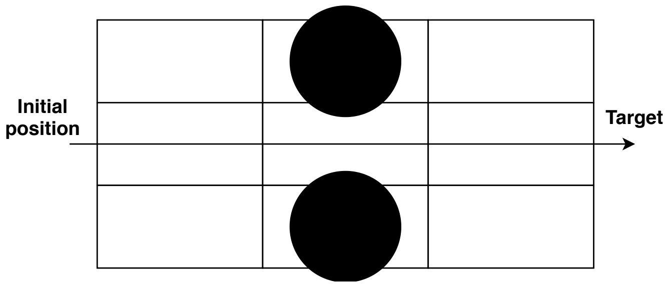


Figure 6.4: Example showing the effect of a too coarse partitioning

more cells at the grid than they actually do. This will allow the vessel to respond earlier to dynamic obstacles.

Another important consideration is to take into account that the local frame between p_k and p_{k+1}, in which the operation area is defined, is already oriented directly towards p_{k+1}. It is therefore reasonable to assume that, for most of the time, the vessel will want to be moving relatively aligned with the y_{local} axis of the local frame. For this reason, the operation area is divided into rectangular cells. This is also illustrated in Fig. 6.3. Partitioning such that the cells are longer in the direction towards the next global waypoint, will make it easier for the path generation module to produce paths that are feasible for the vessel.

Simulations showed that partitioning the grid into cells of size L x B = 4 m x 1 m gave good results.

6.3 Tuning the neural network parameters

The neural network model was found to be robust and worked well for many variations of the parameters, as stated in Yang and Meng (1998), where a thorough discussion on the parameter sensitivity of the model is given. The most important parameter in the model is the passive decay rate A. To low values for this parameter can make the neural activity saturate, so that the next waypoint can not be determined. This is illustrated in Fig. 6.5.

The chosen neural neural network parameters are shown together with the heading weight λ in Table 6.1.

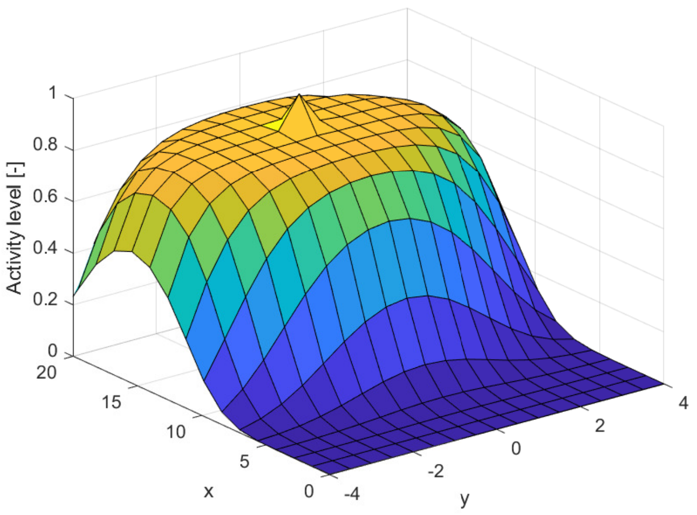


Figure 6.5: Saturation of the activity landscape.

Table 6.1: Neural network parameters

A	B	D	E	μ	λ
50	1	-1	70	1	0,1

Chapter 7

Path generation

The objective of the path generation module is to construct the curve $p_d(s) = col(x_d(s), y_d(s))$ through the waypoints $[q_0, ..., q_t]$ given by the path-planning module. This can be achieved using splines and interpolation techniques. The methods used and presented here are adopted from Skjetne (2005) and Skjetne (2019).

7.1 Generating a $\mathcal{C}^r$ curve through a set of waypoints

A desired path $p_d(s)$ can be divided into n subpaths $p_{d,i}(s)$, $i = 1, ..., n$ between the waypoints, each of which is expressed as a polynomial in s of a certain order. The $\mathcal{C}^r$ requirement means that at every connection point between two subpaths we must have

$$\begin{aligned}
\lim_{s \nearrow i-1} x_{d,i-1}(s) &= \lim_{s \searrow i-1} x_{d,i-1}(s) \quad & \lim_{s \nearrow i-1} y_{d,i-1}(s) &= \lim_{s \searrow i-1} y_{d,i-1}(s) \\
\lim_{s \nearrow i-1} x^s_{d,i-1}(s) &= \lim_{s \searrow i-1} x^s_{d,i-1}(s) \quad & \lim_{s \nearrow i-1} y^s_{d,i-1}(s) &= \lim_{s \searrow i-1} y^s_{d,i-1}(s) \\
&\vdots \\
\lim_{s \nearrow i-1} x^{s^r}_{d,i-1}(s) &= \lim_{s \searrow i-1} x^{s^r}_{d,i-1}(s) \quad & \lim_{s \nearrow i-1} y^{s^r}_{d,i-1}(s) &= \lim_{s \searrow i-1} y^{s^r}_{d,i-1}(s)
\end{aligned}$$

for $i \in \mathcal{I} \setminus \{1\}$. We consider polynomials of order k:

$$\begin{aligned} x_{d,i}(s) &= a_{k,i}s^k + ... + a_{1,i}s + a_{0,i} \\ y_{d,i}(s) &= b_{k,i}s^k + ... + b_{1,i}s + b_{0,i} \end{aligned} \tag{7.1}$$

where the coefficients $\{a_{j,i}, b_{j,i}\}$ must be determined. Since there are $(k+1) \cdot 2$ unknowns for each subpath, there are a total of $(k+1) \cdot 2n$ of coefficients that need to be determined. This can be solved as a set of linear equations for the full path or by calculating the coefficients for each subpath independently. Continuity is then ensured at the connection points by assigning numerical values which are common for the neighboring subpaths.

7.2 Stepwise $\mathcal{C}^3$ path generation

The path-planning module can only decide one waypoint ahead in time, which is decided first when the current waypoint is reached. We therefore require an online path generation strategy. The method is adopted from Skjetne (2019).

Each subpath is parametrized individually, corresponding to a hybrid parametrization. Letting $i = \lfloor s \rfloor + 1 \in \mathcal{I}$ identify the active subpath and $\theta = s - \lfloor s \rfloor \in [0, 1)$ map the point along the path segment, a continuous parametrization is achieved by using the mapping

$$s \mapsto p_d(s) := \bar{p}(i(s), \theta(s)) \tag{7.2}$$

For $q_{0,i}$ given as a general departure waypoint, and $q_{t,i}$ as a destination waypoint, and by letting the subpath i connect $q_{0,i}$ and $q_{t,i}$, the equations to calculate the coefficients $\{a_{j,i}, b_{j,i}\}$ become

$\mathcal{C}^0$: Continuity at the waypoints $p_{0,i}$ gives for segment i:

$$\begin{aligned} x_{d,i}(0) &= x_{0,i} & \qquad x_{d,i}(1) &= x_{t,i} \\ y_{d,i}(0) &= y_{0,i} & \qquad y_{d,i}(1) &= y_{t,i} \end{aligned} \tag{7.3}$$

$\mathcal{C}^1$: The slope at the first waypoint is set to point against the second waypoint

$$x^{\theta}_{d,1}(0) = \frac{(x_{t,1} - x_{0,1})}{|p_{t,1} - p_{0,1}|} \qquad y^{\theta}_{d,1}(1) = \frac{(x_{t,1} - x_{0,1})}{|p_{t,1} - p_{0,1}|} \tag{7.4}$$

The slopes at the intermediate waypoints are chosen as:

$$x^{\theta}_{d,i}(0) = \lambda T_{0x,i} \qquad x^{\theta}_{d,i}(1) = \lambda \frac{(x_{t,i} - x_{0,i})}{|p_{t,i} - p_{0,i}|} \tag{7.5}$$
$$y^{\theta}_{d,i}(0) = \lambda T_{0y,i} \qquad y^{\theta}_{d,i}(1) = \lambda \frac{(y_{t,i} - y_{0,i})}{|p_{t,i} - p_{0,i}|}$$

where $T_{0,i} \in \mathbb{R}^2$ is the unit tangent vector at $p_{0,i}$ and $\lambda > 0$ is a design constant used to tune the curvature of the path.

$\mathcal{C}^j$: Setting derivatives of order $j \geq 2$ to zero gives for $i \in \mathcal{I}$:

$$x^{\theta^j}_{d,i}(0) = 0 \qquad y^{\theta^j}_{d,i}(0) = 0 \tag{7.6}$$
$$x^{\theta^j}_{d,i}(1) = 0 \qquad y^{\theta^j}_{d,i}(1) = 0 \tag{7.7}$$

Since the differentiability requirement of the path is $\mathcal{C}^3$, the above equations up to $j = 3$ gives $16n$ equations to solve for $(k+1)\cdot 2n$ unknowns, meaning that the polynomial in θ must be of order $k = 7$.

Given that the waypoints are provided we can now produce the desired path segments

$$\bar{p_d}(i,\theta) = \begin{bmatrix} x_{d,i}(\theta) \\ y_{d,i}(\theta) \end{bmatrix} \tag{7.8}$$

and the path derivatives

$$\bar{p_d}^{\theta^j}(i,\theta) = \begin{bmatrix} x^{\theta^j}_{d,i}(\theta) \\ y^{\theta^j}_{d,i}(\theta) \end{bmatrix}, j \geq 1 \tag{7.9}$$

Since the desired heading is set using the path tangent vector we generate the path derivatives for each path segment i up to $j = 3$, and the heading curve $\bar{\psi}_d$ and derivatives:

$$\bar{\psi}_d(i,\theta) = atan\left(\frac{\bar{y}^{\theta}_d(i,\theta)}{\bar{x}^{\theta}_d(i,\theta)}\right) \tag{7.10}$$

$$\bar{\psi}_d^{\theta}(i,\theta) = \frac{\bar{x}_d^{\theta}(i,\theta)\bar{y}_d^{\theta^2}(i,\theta) - \bar{x}_d^{\theta^2}(i,\theta)\bar{y}_d^{\theta}(i,\theta)}{\bar{x}_d^{\theta}(i,\theta)^2 + \bar{y}_d^{\theta}(i,\theta)^2} \tag{7.11}$$

$$\bar{\psi}_d^{\theta^2} = \frac{\bar{x}_d^{\theta}\bar{y}_d^{\theta^3} - \bar{x}_d^{\theta^3}\bar{y}_d^{\theta}}{\bar{x}_d^{\theta 2} + \bar{y}_d^{\theta 2}} - 2\frac{\left(\bar{x}_d^{\theta}\bar{y}_d^{\theta^2} - \bar{x}_d^{\theta^2}\bar{y}_d^{\theta}\right)\left(\bar{x}_d^{\theta}\bar{x}_d^{\theta^2} + \bar{y}_d^{\theta}\bar{y}_d^{\theta^2}\right)}{\left[\bar{x}_d^{\theta 2} + \bar{y}_d^{\theta 2}\right]^2} \tag{7.12}$$

and the speed profile $v_{s,i}(t,\theta)$ and its derivatives

$$v_{s,i}(t,\theta) = \frac{u_d(t)}{|\bar{p}_d^{\theta}(i,\theta)|} \tag{7.13}$$

$$v_{s,i}^{t}(t,\theta) = \frac{\dot{u}_d(t)}{|\bar{p}_d^{\theta}(i,\theta)|} \tag{7.14}$$

$$v_{s,i}^{\theta}(t,\theta) = -\frac{\bar{p}_d^{\theta}(i,\theta)^{\top}\bar{p}_d^{\theta^2}(i,\theta)}{|\bar{p}_d^{\theta}(i,\theta)|^3}u_d(t) \tag{7.15}$$

where u_d is the desired speed along the path.

Fig. 7.1 shows an example of a path generated using the hybrid path parametrization, with 8 different waypoints, with the tuning parameter $\lambda = 0, 5$.

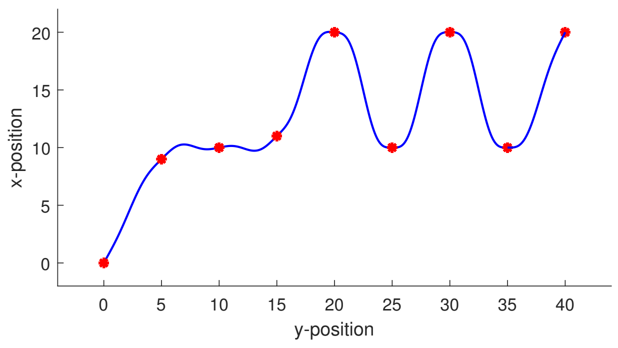


Figure 7.1: Path generated using hybrid path parametrization

Chapter 8

Guidance, navigation and control

Chapter 3, 4 and 5 has presented the path planning and path generation functions. This chapter presents the

- Guidance law, which will use information from the path generation module to provide the control system with a desired position, velocity and acceleration
- Control law, which determines the necessary control forces and moments needed to satisfy the control objective (Fossen, 2011)
- Observer design, which is used to estimate positions, velocities and unmodelled dynamics

The signal flow of the overall system is shown in Figure 8.1, where it is depicted as a *Guidance, navigation and control* (GNC) system, as defined in Fossen (2011).

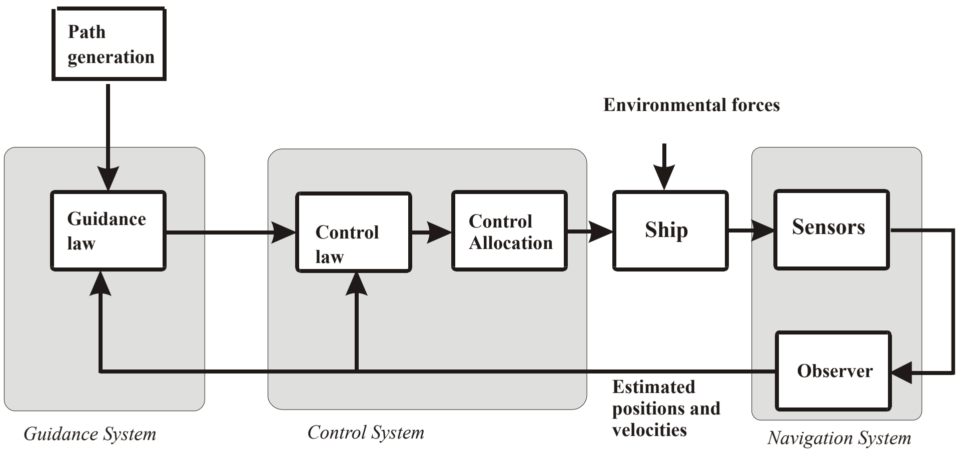


Figure 8.1: Interaction between guidance, control and navigation systems. Adapted from: Fossen (2005).

8.1 Observer design

The purpose of an observer, or a state estimator, is to Sørensen (2018):

- Filter measurement noise: signals can be contaminated by sensor noise or external disturbances, which need to be removed.
- Reconstruct unmeasured states: Due to cost reasons or lack of appropriate sensors, not all the states of a system can be measured. Observers are used to estimate these states so that they can be used in a feedback control loop.
- Dead reckoning: If the sensor signals fall out due to some kind of error, the observer should be able to replace the measured signal, at least for some period of time.

There are several observer designs to choose from. The implemented observer is a modified version of nonlinear passive observer design presented in Fossen and Strand (1999). It has few parameters to tune (compared to e.g. the Kalman filter) and it guarantees global convergence of estimation errors to zero.

The 3 DOF observer equations are

$$\dot{\hat{\eta}} = R(y_3)\hat{\nu} + K_2\tilde{y} \tag{8.1}$$

$$\dot{\hat{b}} = -T^{-1}\hat{b} + K_3\tilde{y} \tag{8.2}$$

$$M\dot{\hat{\nu}} = -D\hat{\nu} + R(y_3)^\top \hat{b} + \tau + R(y_3)^\top K_4 \tilde{y} \tag{8.3}$$

$$\hat{y} = \hat{\eta} \tag{8.4}$$

where the wave filtering equations have been removed as there are no waves present in simulations or experiments at the MCLab. $\hat{\eta}$, $\hat{\nu}$, $\hat{b}$ are the state estimates, $\tilde{y} = y - \hat{y}$ is the measurement estimation error and $T \in \mathbb{R}^{3x3}$ is a diagonal matrix of bias time constants. $K_2 \in \mathbb{R}^{3x3}$, $K_3 \in \mathbb{R}^{3x3}$ and $K_4 \in \mathbb{R}^{3x3}$ are the observer gain matrices. The bias term $\hat{b}$ is assumed to account for slowly varying loads like currents, and other unmodelled dynamics.

8.2 Backstepping maneuvering control design

This section presents a backstepping maneuvering control design, which is used to satisfy the maneuvering control objective in Chapter 3.2.3, given the low speed control design model presented in Chapter 3.1.2. The control design is adopted from Skjetne (2019).

We choose the LgV backstepping design and define

$$z_1 := \left[R(\psi)^\top [\eta - \eta_d(s)]\right], \quad z_2 = \nu - \alpha_1, \quad \omega = \dot{s} - v_s(t) \tag{8.5}$$

8.2.1 Step 1

The design follows these steps:

$$\dot{z_1} = \dot{R}^\top [\eta - \eta_d] + R(\psi)^\top [\dot{\eta} - \eta_d^s \dot{s}] = -rSz_1 + z_2 + \alpha_1 - R(\psi)^\top \eta_d^s(\omega + v_s) \tag{8.6}$$

where S is the skew-symmetric matrix

$$S = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{8.7}$$

The first control Lyapunov function (CLF) is chosen as

$$V_1 = \frac{1}{2} z_1^\top z_1 \tag{8.8}$$

which time differentiated gives

$$\dot{V}_1 = -rz_1^\top S z_1 + z_1^\top z_2 + z_1^\top \left[\alpha_1 - R(\psi)^\top \eta_d^s(\omega + v_s)\right] \tag{8.9}$$

The first virtual control is

$$\alpha_1 = -K_1 z_1 + R(\psi)^\top \eta_d^s v_s + \alpha_{10}, \quad K_1 = K_1^\top > 0 \tag{8.10}$$

and the first tuning function

$$\rho_1 = -z_1^\top R(\psi)^\top \eta_d^s \tag{8.11}$$

Using Young's inequality we get

$$\dot{V}_1 \leq -z_1^\top K_1 z_1 + \rho_1 \omega + \kappa_1 z_1^\top z_1 + \frac{1}{4\kappa_1} z_2^\top z_2 + z_1^\top \alpha_{10} \tag{8.12}$$

where $\kappa_1 > 0$. Choosing

$$\alpha_{10} = -\kappa_1 z_1 \tag{8.13}$$

gives

$$\dot{V}_1 \leq -z_1^\top K_1 z_1 + \rho_1 \omega + \frac{1}{4\kappa_1} z_2^\top z_2 \tag{8.14}$$

$$\alpha_1(t, s, \eta) = -(K_1 + \kappa_1 I) R(\psi)^\top [\eta - \eta_d(s)] + R(\psi)^\top \eta_d^s(s) v_s(t, s) \tag{8.15}$$

We now choose the maneuvering update law, in order for it to only act in the output space of η. First, we have:

$$\rho_1 = -z_1^\top R(\psi)^\top \eta_d^s = V_1^s(\eta, s) \tag{8.16}$$

$$\begin{aligned} \dot{V}_1 &\leq -z_1^\top K_1 z_1 + \rho_1 \omega + \frac{1}{4\kappa_1} z_2^\top z_2 && (8.17) \\ &= -z_1^\top K_1 z_1 - \omega \eta_d^s(s)^\top R(\psi) z_1 + \frac{1}{4\kappa_1} z_2^\top z_2 && (8.18) \end{aligned}$$

We choose the unit-tangent gradient update law:

$$\omega = \mu \frac{\eta_d^s(s)^\top}{|\eta_d^s(s)|} R(\psi) z_1, \quad \mu \geq 0 \tag{8.19}$$

$$\Rightarrow \dot{s} = v_s(t, s) + \mu \frac{\eta_d^s(s)^\top}{|\eta_d^s(s)|} R(\psi) z_1 \tag{8.20}$$

which gives $\rho_1\omega \leq 0$. Concluding Step 1, let $\tilde{K}_1 = K_1 + \kappa_1 I$ which gives

$$\alpha_1(t,s,\eta) = \tilde{K}_1 R(\psi)^\top[\eta - \eta_d(s)] + R(\psi)^\top \eta_d^s(s) v_s(t,s) \tag{8.21}$$

$$\dot{z_1} = -\left(\tilde{K}_1 + rS\right) z_1 + z_2 - R(\psi)^\top \eta_d^s(s)\omega \tag{8.22}$$

$$\dot{s} = v_s(t,s) + \omega \tag{8.23}$$

$$\dot{V}_1 \leq -z_1^\top K_1 z_1 + \frac{1}{4\kappa_1} z_2^\top z_2 \tag{8.24}$$

Since the update law is already chosen, $\dot{\alpha_1}$ must be cancelled directly in the next step:

$$\begin{aligned}\dot{\alpha}_1 &= \sigma_1(t,s,\eta,\nu) + \alpha_1^s(t,s,\eta)\dot{s} &(8.25)\\ &= r\tilde{K}_1 S z_1 - \tilde{K}_1\nu - rSR(\psi)^\top \eta_d^s(s) v_s(t,s) + R(\psi)^\top \eta_d^s(s) v_s^t(t,s) &(8.26)\\ &+ \left[\tilde{K}_1 R(\psi)^\top \eta_d^s(s) + R(\psi)^\top \eta_d^{s^2}(s) v_s(t,s) + R(\psi)^\top \eta_d^s(s) v_s^s(t,s)\right]\dot{s} &(8.27)\end{aligned}$$

so we have

$$\sigma_1(t,s,\eta,\nu) = r\tilde{K}_1 S z_1 - \tilde{K}_1\nu - rSR(\psi)^\top \eta_d^s(s) v_s(t,s) + R(\psi)^\top \eta_d^s(s) v_s^t(t,s) \tag{8.28}$$

$$\alpha_1^s(t,s,\eta) = \tilde{K}_1 R(\psi)^\top \eta_d^s(s) + R(\psi)^\top \eta_d^{s^2}(s) v_s(t,s) + R(\psi)^\top \eta_d^s(s) v_s^s(t,s) \tag{8.29}$$

8.2.2 Step 2

$$M\dot{z}_2 = M\dot{\nu} - M\dot{\alpha}_1 = -D\nu + \tau + R(\psi)^\top b - M[\sigma_1 + \alpha_1^s \dot{s}] \tag{8.30}$$

The second CLF:

$$V_2 = V_1 + \frac{1}{2} z_2^\top M z_2 \tag{8.31}$$

$$\begin{aligned}\dot{V}_2 &= \dot{V}_1 + z_2 M \dot{z}_2 &(8.32)\\ &\leq -z_1 K_1 z_1 + \frac{1}{4\kappa_1} z_2^\top z_2 + z_2^\top\left[-D(z_2 + \alpha_1) + \tau + R(\psi)^\top b - M(\sigma_1 + \alpha_1^s \dot{s})\right] &(8.33)\end{aligned}$$

$$\tau = -K_2 z_2 + D\alpha_1 - R(\psi)^\top b + M(\sigma_1 + \alpha_1^s \dot{s}), \quad K_2 = K_2^\top > 0 \tag{8.34}$$

$$\dot{V}_2 \leq -z_1^\top K_1 z_1 - z_2^\top \left(K_2 - \frac{1}{4\kappa_1} I \right) z_2 \tag{8.35}$$

The final control law and closed-loop system becomes

$$\dot{s} = v_s + \omega \tag{8.36}$$

$$\tau = -K_2 z_2 + D\alpha_1 - R(\psi)^\top b + M(\sigma_1 + \alpha_1^s \dot{s}) \tag{8.37}$$

$$\dot{z}_1 = -\Big((K_1 + \kappa_1 I) + rS \Big) z_1 + z_2 - R(\psi)^\top \eta_d^s(s)\omega \tag{8.38}$$

$$M\dot{z}_2 = -(D + K_2) z_2 \tag{8.39}$$

where $K_1 = K_1^\top > 0$, $K_2 = K_2^\top > 0$ and $\kappa_1 > 0$ are gain matrices to be tuned and ω is the unit-tangent gradient update law in Eq. 8.19.

8.3 Guidance law

Given measurements of η and the gain $\mu \geq 0$ the guidance law will implement the path-parameter dynamics

$$\dot{s} = v_s(t, s) + \mu \frac{(\eta_d^s)^\top}{|\eta_d^s|} (\eta - \eta_d) \tag{8.40}$$

which is used together with the signals from the path generation module to provide the controller with the desired states. These are found as follows (Skjetne, 2019):

$$i = \lfloor s \rfloor + 1, \quad \theta = s - \lfloor s \rfloor \tag{8.41}$$

$$v_s = v_{s,i}(t, \theta), \quad v_s^t = v_{s,i}^t(t, \theta), \quad v_s^s = v_{s,i}^\theta(t, \theta) \tag{8.42}$$

$$\psi_d = \bar{\psi}_d(i, \theta), \quad \psi_d^s = \bar{\psi}_d^\theta(i, \theta), \quad \psi_d^{s^2} = \bar{\psi}_d^{\theta^2}(i, \theta) \tag{8.43}$$

$$p_d = \bar{p}_d(i, \theta), \quad p_d^s = \bar{p}_d^\theta(i, \theta), \quad p_d^{s^2} = \bar{p}_d^{\theta^2}(i, \theta) \tag{8.44}$$

$$\eta_d = \begin{bmatrix} p_d \\ \psi_d \end{bmatrix}, \quad \eta_d^s = \begin{bmatrix} p_d^s \\ \psi_d^s \end{bmatrix}, \quad \eta_d^{s^2} = \begin{bmatrix} p_d^{s^2} \\ \psi_d^{s^2} \end{bmatrix} \tag{8.45}$$

8.4 Tracking control design

The backstepping controller presented in the previous section can be difficult to tune. This was the case during experiments at MCLab. Tuning parameters that had worked well in simulations did not produce good results. A proportional–integral–derivative (PID) tracking controller, which allows for a more intuitive tuning process, was therefore implemented for use at MCLab. The design is adopted from Lindegaard and Fossen (2003), based on the same control design model from Chapter 3.1.2.

Letting $\eta_e = \eta - \eta_d$, and $\nu_e = \nu - \nu_d$ the tracking control law is

$$\dot{\xi} = \eta_e \tag{8.46}$$

$$\tau = -M(K_i R(\psi)^\top \xi + K_p R(\psi)^\top \eta_e + K_d \nu_e) + D\nu_d \tag{8.47}$$

where $K_i > 0$, $K_p > 0$ and $K_d > 0$ are gain matrices to be tuned. Note that in this controller, the bias is compensated for through the integral state ξ, instead of using the bias estimate from the observer, as was the case in the maneuvering control design. In this case, we choose $\omega = 0$, so that the path parameter dynamics become

$$\dot{s} = v_s(t, s) \tag{8.48}$$

Chapter 9

Results

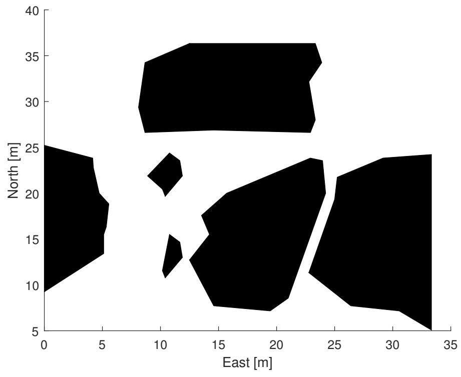
This chapter starts with presenting results for the global and local path-planners on maps with some simple topographical obstructions. Movement in the BINN is then simulated with CSAD, using the hybrid path parametrization. Finally, results for the whole system (AutoVoyage) are presented through simulations and experimental results from MCLab.

The neural network parameters used in simulations are the one presented in Table 6.1, and the size of the grid decomposition is set as explained in Chapter 6. The tuning parameter λ used in the path generation is set to $\lambda = 0,5$. μ in the unit-tangent gradient update law is set to $\mu = 0,2$. The gains of the backstepping controller is set to $K_1 = diag([0.050.050.035])$, $K_2 = diag([99.699.624.9])$ and $\kappa_1 = 0$. The radius of acceptance is chosen as $R = 0.3$ m.

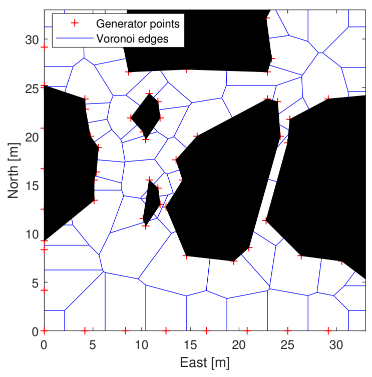
9.1 Global path-planning

9.1.1 Scenario 1

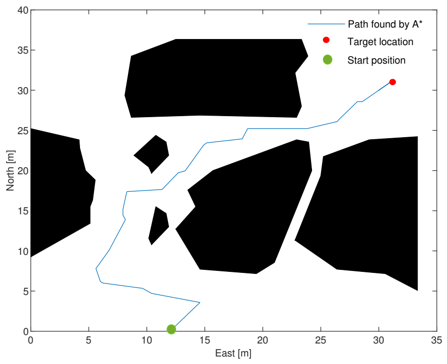
Fig. 9.1 shows the global path-planning algorithm solving a path-planning problem from $p_0 = (0, 12)$ to $p_t = (30, 30)$. As can be seen in Fig. 9.1)d), it would be possible to omit more waypoints, and still have a path that does not intersect any obstacles. However, the algorithm chooses not to do so, because the resulting path would be to close to obstacles.



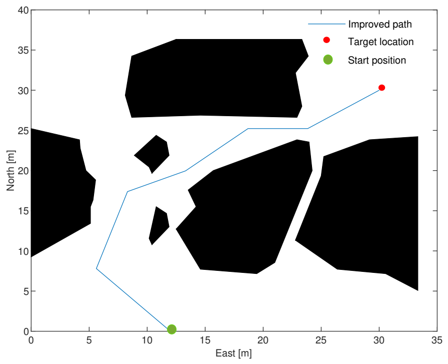
(a) Operation area



(b) Voronoi partitioning



(c) Path found by A* algorithm

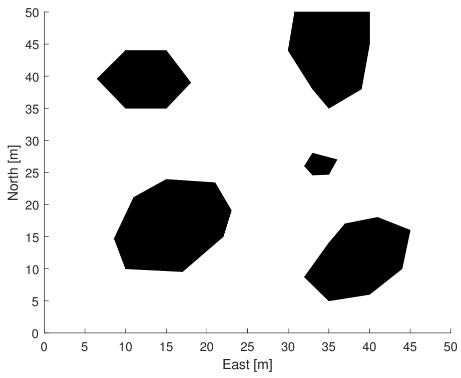


(d) Refined path

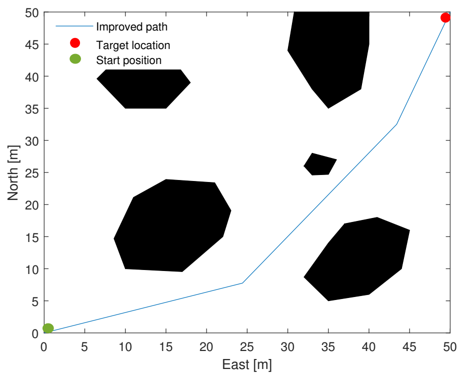
Figure 9.1: Scenario 1, global path-planning

9.1.2 Scenario 2

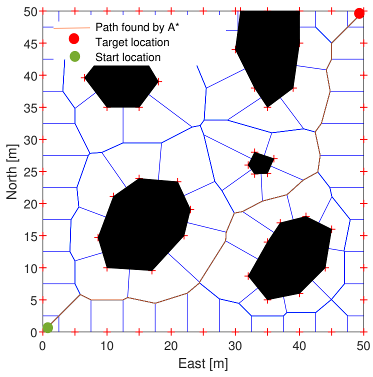
Fig. 9.2 shows the global path-planning algorithm finding the path from $p_0 = (0,0)$ to $p_t = (50,50)$. Fig. 9.2)c) shows how the initial path found by the A* search is taken directly from the edges of the Voronoi diagram. The final path with all uncesessary waypoints removed is seen to be safe and efficient.



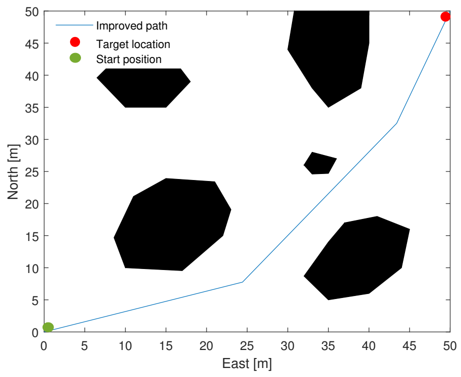
(a) Operation area



(b) Voronoi partitioning



(c) Path found by A* algorithm

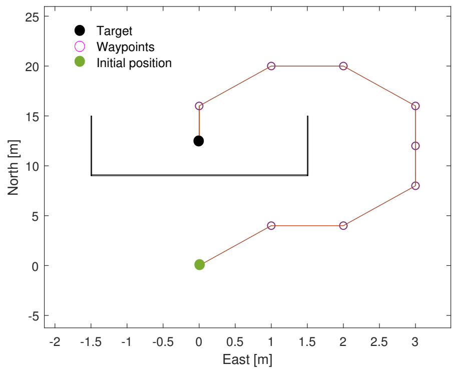


(d) Refined path

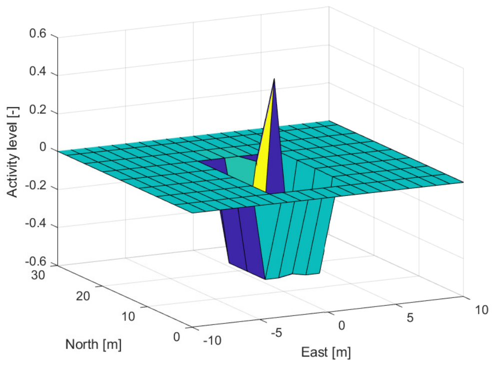
Figure 9.2: Scenario 2, global path-planning

9.2 Path-planning using BINN

This section shows how the BINN algorithm solved two path-planning scenarios. The scenarios are not necesarilly realistic for a ship, but they are used to illustrate the path-finding capabilities of the BINN algorithm. Fig. 9.3 shows the algorithm finding the path to a target inside a U-shaped obstacle. Since the activity is not able to propagate through the obstacles, the algorithm immediately knows to start a turn. The path is seen to keep good distance to obstacles. The activity level appears to be very close to zero at neurons not occupied by a target or obstacles. This is due to the large passive decay rate. Still, the algorithm is able to separate the values and find the optimal path.



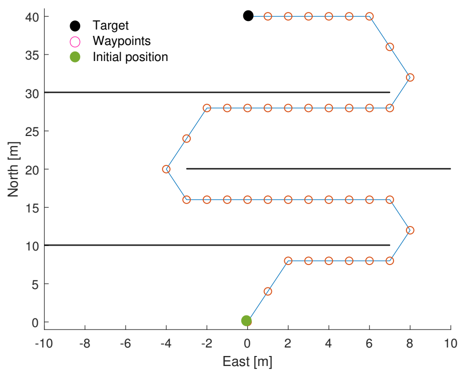
(a) Operation area



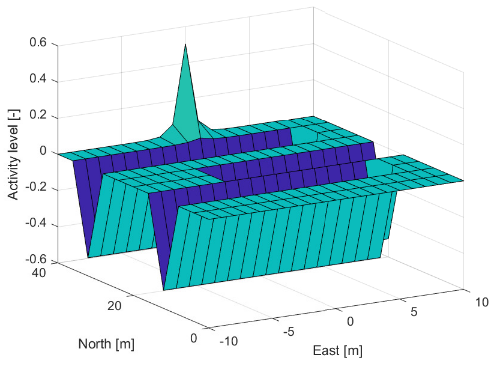
(b) Activity landscape

Figure 9.3: Scenario 1, BINN path-planning

Fig. 9.4 shows a scenario type where the BINN algorithm is particularly efficient. As can be seen, there are 39 waypoints between the initial and target position. While the BINN algorithm is able to find the optimal path simply by following the maximum activity at each step, a shortest-path graph search algorithm could end up having to search a much larger portion of the state space to find the same path.



(a) Operation area

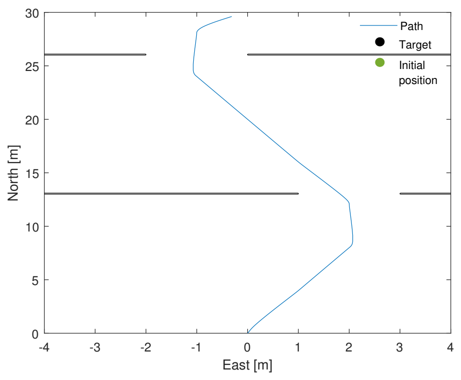


(b) Activity landscape

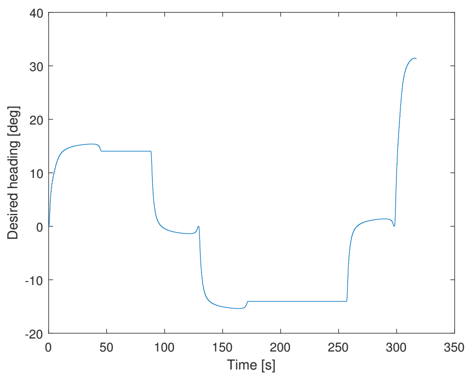
Figure 9.4: Scenario 2, BINN path-planning

9.3 Movement in BINN with $\mathcal{C}^3$ path generation

The rest of the simulations are carried out using the simulation model of CSAD presented in Chapter 3.1.1 and the maneuvering control design in Chapter 8.2. Fig. 9.5)a) shows the path travelled by the vessel through an obstacle course. It is seen to keep good distance to obstacles. Fig. 9.5)b) shows that the desired heading curve, which is tangent to the path, is smooth, but with some undesirable fast wiggles.



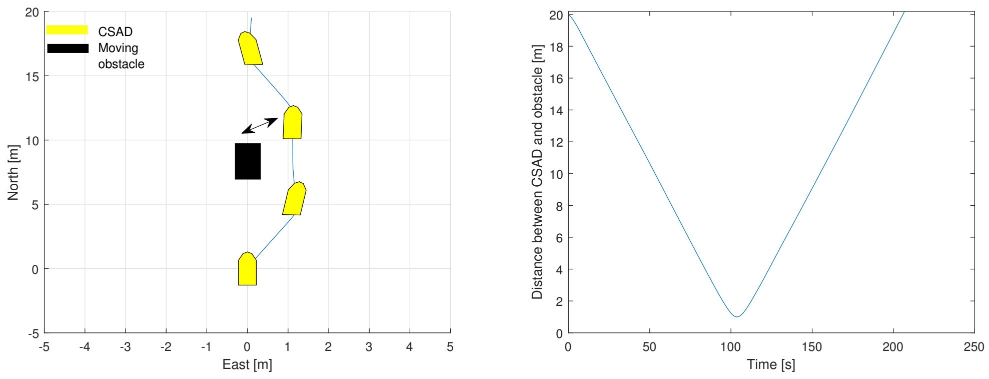
(a) North-east plot of travelled path



(b) Desired heading, tangent to the path

Figure 9.5: Scenario 1, obstacle course

In the next simulation a dynamic obstacle is introduced. The obstacle is a rectangle of similar size to CSAD, L x B = 2,5 m x 0,4 m, but it is represented in the BINN network as twice it size. CSAD has initial position $\eta_0 = [0, 0, 0]^\top$ and the obstacle has initial position $\eta_0 = [20, 0, \pi]^\top$. The vessels are then set to move straight forward so that they are on collision course, with $u_d = 0, 1m/s$ for both vessels.



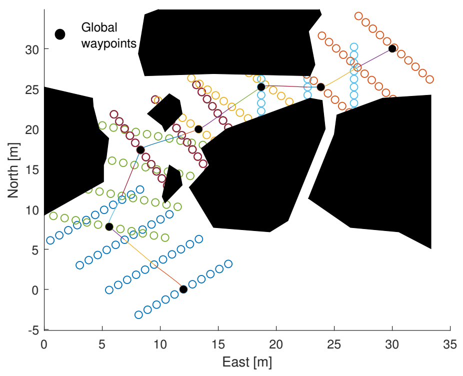
(a) North-east plot of desired path **(b)** Distance between CSAD and obstacle

Figure 9.6: Scenario 2, head-on situation

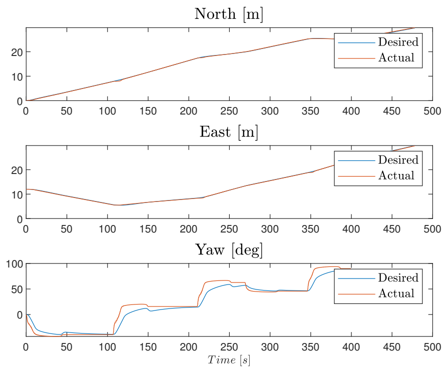
Fig. 9.6)a) shows that CSAD was able to safely perform obstacle avoidance, and then continue on its path. The black two-sided arrow show where the vessels were after approximately 120 seconds. Fig. 9.6)b) shows the distance between the two vessels as a function of time. It is smallest after approximately 104 seconds, which is when they are side-by-side.

9.4 Testing of the complete system

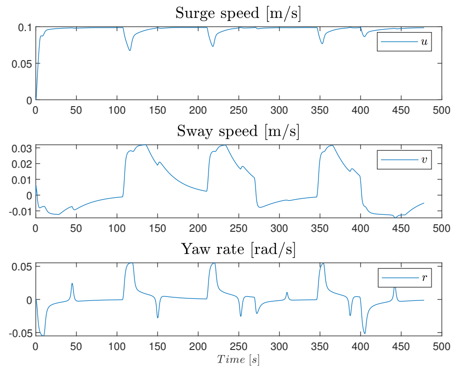
This section presents results for two scenarios, using the complete AutoVoyage system. In both scenarios, a constant current of $v_c = [0.01 \text{ m/s}, 0.01 \text{ m/s}, 0]^\top$ is present. CSAD was commanded to move along the path with a surge speed $u_d = 0,1$ m/s. Fig. 9.7)a) shows the path through the partitioning, which is seen to go through the center of the neurons in the BINN. Fig. 9.7)b) and c) show that both the geometric and the dynamic part of the maneuvering control objective is satisfied well, but the vessel struggles to maintain the desired heading. Fig. 9.7)d) shows that the commanded thrust in sway and yaw during turns are larger than feasible. These are however saturated by the thrust allocation and the vessel is still able to follow the path well.



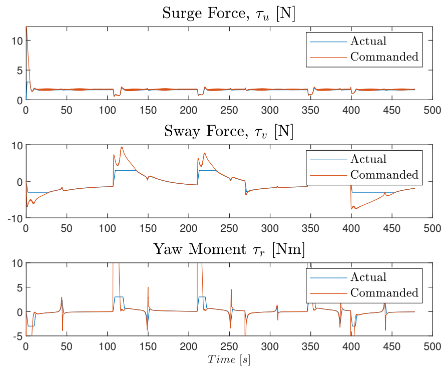
(a) Operation area



(b) Positions



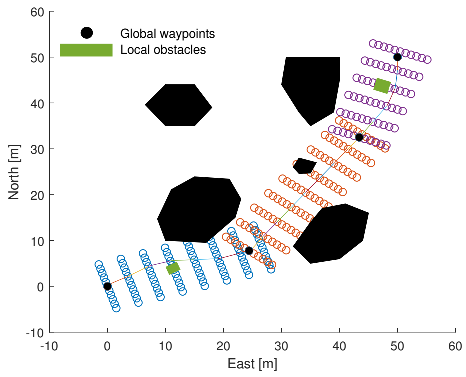
(c) Body-fixed velocities



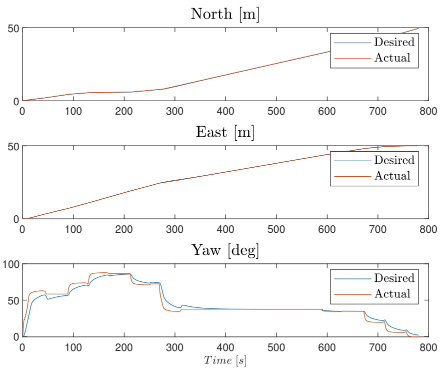
(d) Control forces

Figure 9.7: Scenario 1, AutoVoyage

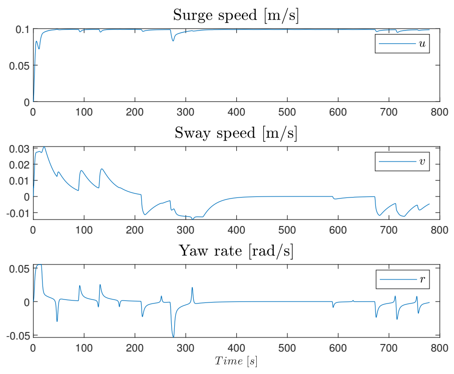
In scenario 2, two previously unknown "local obstacles" are introduced to be dealt with by the local path-planner. Fig. 9.8)a) shows that these are navigated past safely. Fig. 9.8)b) and c) shows that the geometric and dynamic tasks are satisfied similarly as for scenario 1.



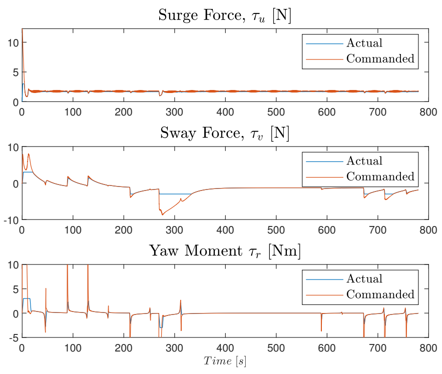
(a) Operation area



(b) Positions



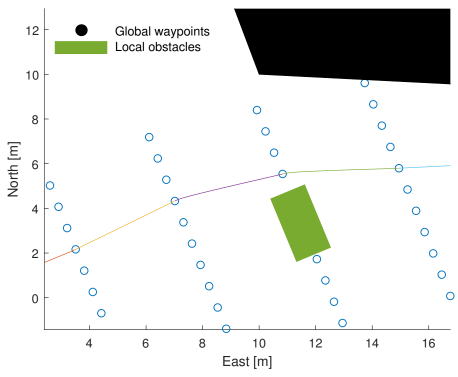
(c) Body-fixed velocities



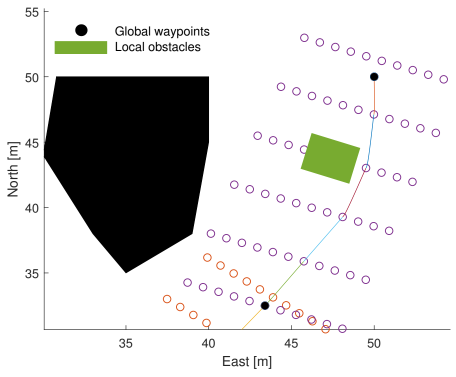
(d) Control forces

Figure 9.8: Scenario 2, AutoVoyage

Fig. 9.9 shows a zoomed in view of scenario 2. It illustrates how the distance between neurons are directly related to the size of the grid partitioning.



(a) First local obstacle

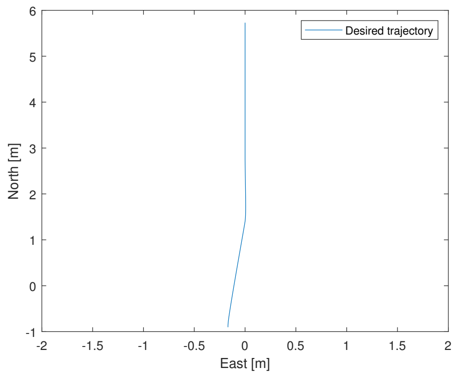


(b) Second local obstacle

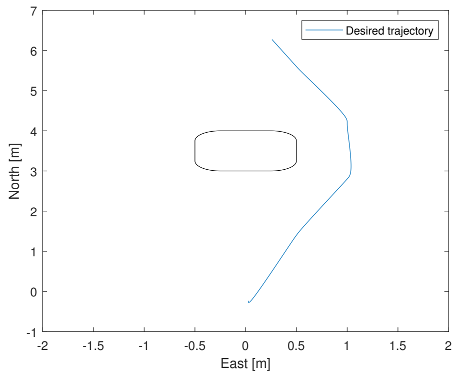
Figure 9.9: Zoomed in view, scenario 2

9.5 Experimental results

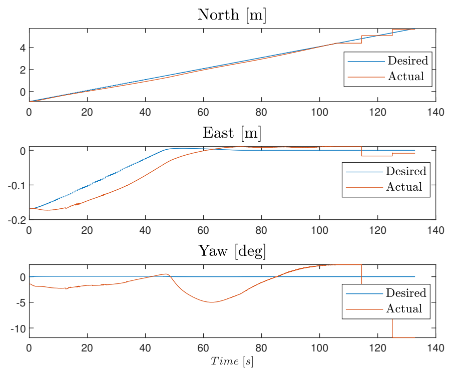
As shown in the attached log files, the position measurements from QTM would vanish whenever the ship moved past approximately $4,5m$ in positive north direction from the origin in the MCLab basin. This left very little space left for maneuvering. Still, two simple tests were carried out to show that the online path-planning and path generation works well. As previously mentioned, a PID tracking controller was used for experiments at MCLab, for simplicity in tuning. Due to the limited space available for maneuvering a tighter partitioning of the BINN, $1,4$ m x $0,5$ m, was used. In both scenarios, the objective is to get from the current position of the vessel (which is made to be close to the origin) to $p_t = (7,0)$, with the speed assignment $u_d = 0,05$ m/s. Fig. 9.10 shows the result from the experiments. Results show decent performance of the tracking controller, though it is seen to fall behind at the beginning, especially for scenario 1. The worst performance is seen in tracking of the desired heading. The controller is also seen to struggle with satisfying the speed assignment, though this could also be due to poor estimates from the observer.



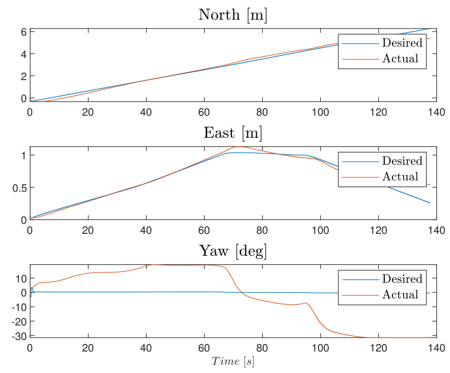
(a) Scenario 1: Operation area and desired trajectory



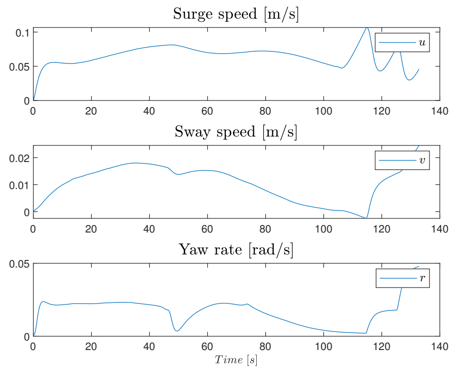
(b) Scenario 2: Operation area and desired trajectory



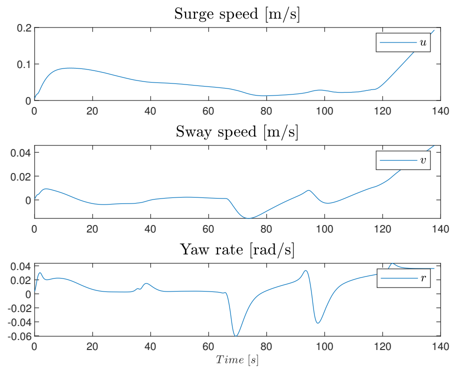
(c) Scenario 1: Positions



(d) Scenario 2: Positions



(e) Scenario 1: Estimated velocities



(f) Scenario 2: Estimated velocities

Figure 9.10: Scenario 1 and 2, MCLab experiments

9.6 Discussion

In summary, simulations showed that the vessel was able to autonomously complete a voyage in many different scenarios. Both the global and the local path-planner produces paths that are safe and efficient in their operation areas, and integration of the two worked well. The vessel is able to accurately follow the parametrized path all the way from point of departure to point of arrival.

It was however seen that the vessel was not always able to achieve the desired heading. This could be due to sub-optimal tuning, or it could mean that some of the turns are to sharp for the vessel. The paths could be made easier to follow by increasing the size of the cells in the grid partitioning, though the resulting paths would be less efficient in terms of path length. As long as the path is constrained to move through the centers of each cell, there will be a weighting between the resolution of the BINN, which must be high enough for obstacle avoidance, and the feasibility of the path.

Increasing the size of the representation of the dynamic obstacles in the BINN, is an easy solution to a complicated problem. In one sense, it is very robust. It forces the vessel to respond early to the presence of a dynamic obstacle, as the BINN algorithm will never choose to go to a neuron position which is registered as occupied by an obstacle. In that sense, the system is guaranteed to perform obstacle avoidance. The weakness lies in the assumption that accurate information about the size of dynamic obstacles can be obtained. Furthermore, if the frequency of dynamic obstacle encounters are high the total path can become inefficient.

It should be mentioned that simulations with a stronger current were attempted, which were not succesful. The controller was not able to successfully compensate for the stronger current, which is believed to be because of poor bias estimation from the observer when the vessel changes heading.

Experimental results from MCLab were obtained from much simpler scenarios, but they show that the online path-planning and path generation strategy works well. It also demonstrates that the system is computationally efficient enough to respect the hardware limitations on-board.

Chapter 10

Conclusions and suggestions for further work

10.1 Conclusion

This thesis has presented an approach to autonomous path-planning, path generation and maneuvering control, for the purpose of autonomous transit operation for a marine surface vessel. The system has been validated through simulations and lab experiments.

The global path-planner worked well, which is to be expected as it is largely based on established methods. Partitioning using Voronoi diagrams gave the A* search enough possible paths to find an efficient route through the partitioning. Waypoint reduction made sure that the path was as efficient as possible, given the clearance constraints.

Integration with the BINN approach for local path-planning using hybrid path parametrization also yielded good results. The paths are in general satisfactory, staying clear of obstacles and avoiding sharp turns. It was also shown that the vessel was able to perform dynamic obstacle avoidance in a head-on collision situation.

The system is shown to be robust, and can easily be adapted. Adjusting the clearance of the path with respect to obstacles is simply a matter of changing the size of the partitioning of the grid that the BINN is organized on. This flexibility opens up for many possible areas of use.

10.2 Recommendations for further work

As of now, the system is only concerned with maneuvering in a safe manner. A natural next step for the guidance concept is to make the system comply with the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs). These regulations dictate how one should perform collision avoidance maneuvers at sea. For example, in the head-on situation simulated in Chapter 9.3, COLREGs state that both ships shall alter course to starboard. The fact that CSAD performed a starboard maneuver was coincidental, as turning port-side would have yielded a path with similar length.

As the vessel was not always able to keep the desired heading, it is suggested to loosen the constraint of forcing the path to go through the center of the cells in the grid decomposition. This is employed in (Scibilia et al., 2012), and gives more flexibility in path generation.

Another very interesting idea is to change strategy with respect to currents. In this work, it was always attempted to counteract the effect of currents in the controller through the bias estimation. It is necessary to have this function in the motion control system. In some cases however, it is possible to use the energy in the current by letting the current influence the path. By incorporating this in the decision-making on which waypoint to choose next, the energy consumption of the system could be reduced. The control system still has to be able to compensate for the effect of currents, which will require either a better bias estimate or to instead include integral action in the controller.

For further testing of the system at MCLab, it is recommended to implement the system on a vessel smaller than CSAD, such that the space available for maneuvering is larger compared to the size of the ship. This would allow for testing of more advanced scenarios.

Bibliography

AAWA, 2016. Remote and Autonomous Ships: The next steps. `https://www.rolls-royce.com/~/media/Files/R/Rolls-Royce/documents/customers/marine/ship-intel/aawa-whitepaper-210616.pdf` [Accessed: 15.10.2018].

Bjørnø, J., 2016. Thruster-Assisted Position Mooring of C/S Inocean Cat I Drillship. Master thesis, NTNU.

Caltech, 2019. Summary of the A* Algorithm. `http://robotics.caltech.edu/wiki/images/e/e0/Astar.pdf` [Accessed: 12.05.2019].

Candeloro, M., Lekkas, A. M., Sørensen, A., 2017. A Voronoi-Diagram-Based Dynamic Path-Planning System for Underactuated Marine Vessels. Control Engineering Practice.

Dubins, L., 1957. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. American Journal of Mathematics, 497–516.

Fossen, T. I., 2005. A nonlinear unified state-space model for ship maneuvering and control in a seaway. Journal of Bifurcation and Chaos.

Fossen, T. I., 2011. Handbook of Marine Craft Hydrodynamics and Motion Control. John Wiley Sons, Ltd.

Fossen, T. I., Strand, J. P., 1999. Passive nonlinear observer design for ships using Lyapunov methods: full-scale experiments with a supply vessel. Automatica 35, p 3-16.

Fraichard, T., Scheuer, A., 2004. From Reeds and Shepp's to continuous-curvature paths. IEEE Transactions on Robotics.

Huang, Z., Zhu, D., Sun, B., 2016. A multi-auv cooperative hunting method in 3-d underwater environment with obstacle. Engineering Applications of Artificial Intelligence.

Hvamb, K., 2015. Motion planning algorithms for marine vehicles. Master thesis, NTNU.

Latombe, J.-C., 1991. Robot Motion Planning. Kluwer Academic Publishers Group.

Lekkas, A. M., 2014. Guidance and path-planning systems for autonomous vehicles. PhD thesis, NTNU.

Lekkas, A. M., Dahl, A. R., Breivik, M., Fossen, T. I., 2013. Continuous-Curvature Path Generation Using Fermat's Spiral. Modeling, Identification and Control, Vol. 34, No. 4, 2013, pp. 183–198, ISSN 1890–1328.

Lindegaard, K.-P., Fossen, T. I., 2003. Fuel-efficient rudder and propeller control allocation for marine craft: Experiments with a model ship. IEEE Transactions on Control Systems Technology.

Lyngstadaas, O. N., 2018. Ship Motion Control Concepts Considering Actuator Constraints. Master thesis, NTNU.

Ni, J., Wu, L., Shi, P., Yang, S., 2017. A Dynamic Bioinspired Neural Network Based Real-Time Path Planning Method for Autonomous Underwater Vehicles. Computational Intelligence and Neuroscience Volume 2017, Article ID 9269742.

Scibilia, F., Jørgensen, U., Skjetne, R., 2012. AUV Guidance System for Subsurface Ice Intelligence. Proceedings of the ASME 2012 31st International Conference on Ocean, Offshore and Arctic Engineering.

Skjetne, R., 2005. The Maneuvering Problem. PhD thesis, NTNU.

Skjetne, R., 2019. Notes on: Maneuvering control design of a low-speed fully-actuated vessel with stepwise path generation. Unpublished work, NTNU.

SNAME, 1950. Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. Technical and Research Bulletin No. 1–5.

Sørensen, A., 2018. Marine Cybernetics, Towards Autonomous Marine Operations and Systems. Department of Marine Technology, NTNU.

Yang, S. X., Meng, M., 1998. A neural network approach to real-time trajectory generation. IEEE Proc. Int. Conf. Robot. Automat, 1725–1730.

Yang, S. X., Meng, M., 2001. Neural network approaches to dynamic collision-free trajectory generation. IEEE Transactions ON SYSTEMS, MAN, AND CYBERNETICS-PART B: CYBERNETICS, VOL. 31.

Ørjan Grefstad, 2018. Development of an obstacle detection and avoidance system for ROV. Master thesis, NTNU.

Šeda, M., 2007. Roadmap Methods vs Cell Decomposition in Robot Motion Planning. Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation.

Appendix

A - Attached files in the delivery file

- Digital version of the thesis
- Digital version of the master poster
- Log files from MCLab
- Veristand project
- Matlab and Simulink code