UNIVERSIDADE Đ COIMBRA

André Filipe Almeida Santos

AUTONOMY FOR UNMANNED SURFACE VEHICILE

OBSTACLE DETECTION AND AVOIDANCE

Autonomia para Veículos de Superfície Aquática

Deteção e Desvio de Obstáculos

Dissertação no âmbito do curso Mestrado Integrado em Engenharia Mecânica no ramo de Produção e Projeto, orientada pelo Professor Doutor Pedro Mariano Simões Neto e apresentada ao Departamento de Engenharia Mecânica.

Setembro de 2018

Autonomy for Unmanned Surface Vehicle

Autonomia para Veículos de Superfície Aquática

André Filipe Almeida Santos

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisor(s): Prof. Pedro Mariano Simões Neto

Examination Committee

Chairperson: Prof. Cristóvão Silva
Supervisor: Prof. Pedro Mariano Simões Neto
Member of the Committee: Prof. Nuno Alberto Marques Mendes

September 2018

Acknowledgments

I would first like to thank my thesis advisor, professor Pedro Neto of the Department of Mechanical Engineering at University of Coimbra. Prof. Pedro Neto was the first person to incite me into the robotics path and throughout the years has helped me in the pursuit of my career.

I would also like to thank CEiiA, in which I developed myself through a summer internship and as the place for my thesis. Thanks to CEiiA continuous guidance, this thesis was possible and developed.

I would also like to thank my CEiiA advisor, Ivan Sinkarenko. It was due to his insights that many adversities were overcome.

I would also like to thank my CEiiA thesis colleagues. It was thanks to them that some points in the journey became more bearable.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Resumo

Navegação autónoma tem sido desenvolvida e apresentada tanto para a terra como ar. No entanto, autonomia para veículos de superfície marítima tem provado algumas dificuldades. Para desenvolver um veículo autónomo de superfície aquático (ASV), é necessário um sistema de orientação, navegação e controlo (GNC) totalmente operacional. Esta tese apresenta uma parte deste sistema GNC para o ASV com um enfase em algoritmos para evitar colisão, com o maior foco na deteção das regras do mar a seguir, COLREGS, e da sua aplicação.

Para os algoritmos para evitar colisão, dois sub-módulos foram implementados, sendo o Dynamic Window Approach (DWA) para obstáculos estáticos, e um algoritmo que segue as regras COLREGS para situações de cruzamento de veículos, sendo definidos com perímetros de segurança. Algumas modificações foram introduzidas relativamente ao algoritmo original de DWA, de modo a melhor responder às condições apresentadas, como um reset da variável de obstáculos, quando estes se encontram fora do perímetro interno, de modo a reduzir o esforço computacional requerido. O algoritmo que segue as regras COLREGS implementado, é capaz de detetar qual a regra a seguir e gera o novo conjunto de orientações e velocidades de modo a evitar colisão e cumprir as regras do mar. Sensores, como um IMU e um sistema GNSS, são responsáveis pela representação das posições e estados do veículo, em que por outro lado, o Lidar e um sistema AIS, são responsáveis pela representação dos obstáculos e seus atributos. ROS foi introduzido como o middleware para o ASV, o que evita a necessidade de desenvolver código para muitos aspetos do controlo do veículo. As simulações realizadas no Gazebo demonstraram que o sistema de evitar colisão consegue dar resposta a obstáculos estáticos e dinâmicos, na condição que os obstáculos dinâmicos sejam veículos com sistemas AIS integrados. Foi demonstrado que este algoritmo é capaz de detetar a respetiva regra a seguir e proceder com uma nova trajetória de modo a evitar colisão.

Palavras-chave: ASV, Autonomia, COLREGS, DWA, Evitar Colisão, Gazebo, ROS.

Abstract

Autonomous navigation has been successfully presented and developed both on ground and air. Though, autonomy on maritime surface vehicles has proven to be more challenging. In order to have an Autonomous Surface Vehicle (ASV), it ought to have a completely functional Guidance, Navigation and Control (GNC) system. This work presents part of the GNC system for the ASV giving special emphasis on collision avoidance and as the main focus the COLlision REGulationS (COLREGS) mode detection and it's application.

For collision avoidance, two submodules were implemented, being the Dynamic Window Approach (DWA) for static obstacles, and a COLREGS compliant algorithm for multi-vehicle encounters, which are set with perimeters of safety. Slights modifications to the original DWA algorithm were introduced in order to better suit the conditions presented, such as the reset of the array of obstacles, when thus are outside of the inner perimeter, in order to reduce computational effort. The COLREGS algorithm implemented is able to detect the COLREGS rule to follow and to generate a new combination of heading and velocity to both, avoid collision and to correspond to the rules requirements. Sensors, such as an IMU and a GNSS system, were responsible to provide total knowledge of the vehicle's position and state, whereas a Lidar and an AIS system, were responsible for the representation of the obstacles and it's attributes. ROS has been employed as the middleware for the ASV, which avoids the need to develop your own code for many aspects of robot control. The computer simulations with Gazebo showed that the collision avoidance system could handle static and dynamic obstacles, with the condition that such dynamic obstacles were vessels with an AIS system integrated. It was demonstrated that the algorithm was capable of detecting the respective rule to follow and thus proceed with a new course which would lead to avoid collision.

Keywords: ASV, Autonomy, Collision Avoidance, COLREGS, DWA, Gazebo, ROS.

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	Relative bearing from contact to ownship.
β	Relative bearing from ownship to contact.
δ	Deviation.
$\dot{\beta}$	Bearing rate.
ψ	Yaw.
θ	Heading.

Roman symbols

bng	Bearing.
h	Heuristic.
$\dot{r}$	Range rate.
r	Range.
$relbng$	Relative bearing.
t	Time.

Subscripts

b	Body-Fixed non-inertial frame.
cn	Contact (of/to).
os	Ownship (of/to).
tn	Tangent heading.
ref	Reference condition.

Superscripts

cn	Contact (from).
os	Ownship (from).

Acronyms

AI Artificial Intelligence.

AIS Automatic Identification System.

ASV Autonomous Surface Vehicle.

CPA Closest Point of Approach.

COG Course Over Ground.

COLREGS COLision REGulationS.

DOF Degreees of Freedom.

DWA Dynamic Window Approach.

GNC Guidance, Navigation and Control.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

IMO International Maritime Organization.

IMU Inertial Measurement Unit.

MMSI Maritime Mobile Service Identity

NOAA National Oceanic and Atmosferic Administration.

ODA Obstacle Detection and Avoidance.

ROS Robot Operating System.

SLAM Simultaneous Localization And Mapping.

SNAME Society of Naval Architects and Marine Engineers.

SOG Speed Over Ground.

USV Unmanned Surface Vehicles.

Chapter 1

Introduction

1.1 Motivation

Throughout the years, the deep sea has been an area unknown to humans. High pressures, pitch black darkness and extreme temperatures are some of the obstacles that result in a near-impossible exploitation so far. Even though the ocean covers more than 70% of the planet's surface, NOAA (National Oceanic and Atmosferic Admnistration (NOAA), 2000) estimates that up to 95% of the world's oceans are unexplored and unseen by human eyes. In 2009, (Ministry of defence of Portugal, 2009) there was a submission for a proposal to extend the Portuguese continental shelf, which being accepted, will result in Portugal to become 3% of land and 97% of opportunities (figure A.1 in appendix A).

Automation greatly contributed to the huge leaps in the evolution of modern society. While the first automative processes were developed to reduce working load and save time, nowadays automation is crucial for safety, cost-effective solutions and quality. Even though humans have the ability to adapt and to apply judgment, they are inconsistent and are subjected to human errors. This necessity led to the implementation of autonomy on surfaced vehicles.

Autonomous Surface Vehicle (ASV), also known as Unmanned Surface Vehicles (USV), are vessels which operate on the water surface without any operator and crew. In (Liu et al., 2016) are presented the main ASV developments all around the world from 1985 to 2016. However, most ASVs presented could be named as merely “semi-autonomou” ASVs. Largely due to the challenges in automated and reliable guidance, navigation and control (GNC) functions which were not capable of withstanding all the diverse operating conditions, in face of sophisticated and hazardous environments, and due to the sensors, actuators and communication systems presented at the time. With further developments of these systems and hardware, it is now conceivable the development of “fully-autonomous” ASVs which is a prerequisite in order to have more effective, safe and reliable solutions. Collision avoidance is an important part of such system and compromises a vital component of the self-navigation of the unmanned surface vehicle.

The different uses for ASVs are scientific research, environmental missions, exploitation of ocean resources, military uses and other applications. (Liu et al., 2016) presents a specific description of each with examples of some prototypes developed. The purpose of the present thesis is for a specific application of scientific research, the bathymetric survey, which is the study of the underwater depth of lakes and ocean floors. Fully autonomous ASVs are the best choice for most missions, as it was compared in (Liu et al., 2016), where the performance of ASVs against other vehicles[1] [2] demonstrated a clear advantage in most parameters overall, table 1.1. However, when presented with a semi-autonomous ASV, where an operator must control the course and prevent collisions throughout the mission, a more rudimentary technique can be more cost effective, thus there is the need to confer autonomy to the ASVs.

Table 1.1: Comparison of ASV/USV performance with other vehicles. Adapted from (Liu et al., 2016).

Clear advantage of ASVs 🟢 Near parity ⚪ Clear disadvantage of ASVs 🔴

Attributes	UUVs	Float Platforms	Satellites	Manned Ships	UAVs	Manned Aircrafts
Endurance	🟢	🔴	🔴	⚪	🟢	🟢
Payload Capacity	🟢	⚪	🟢	⚪	🟢	⚪
Cost	⚪	🔴	🟢	⚪	🟢	⚪
Maneuverability	🟢	🟢	🟢	🟢	🔴	🟢
Deployability	🟢	🟢	🟢	🟢	🔴	🟢
Water depth measurement	🔴	⚪	🟢	⚪	🟢	⚪
Autonomy requirement	⚪	🟢	🟢	🟢	⚪	🟢

1.2 Obstacles in Collision Avoidance

Obstacle prevention is the major problem facing the deployment of ASV missions, thus the necessity of implementing a fully autonomous system. Collision challenges for ASVs can be differentiated in both dynamic and static obstacles.

In (Tan, 2006), collision avoidance is defined as “the ability of a vehicle to detect and avoid colliding with both static and dynamic obstacles, while still attempting to accomplish the current mission objective.”

As presented in (Byrne et al., 2012), obstacle prevention involves more than just other vessels. An ASV has the potential to be caught in static obstacles such as buoys, lobster traps, fishing nets, ice and natural debris. Keeping an operational ASV out of all possible entanglements while conducting coastal environmental monitoring or task performance can prove to be challenging.

When referring to dynamic obstacles, the grand majority are other vessels. The main issue with other vessels is not only to generate a safe route, which would prevent collision, but to also comply with the maritime rules. As there are rules for the roads on land, rules for the sea also exist and must be complied. Hence the main focus of this thesis, a collision avoidance structure that follows the laws of COLREGS (COLision REGulationS) (chapter 2.1.1) and ensures safety for both our ASV and others.

[1]UUV - Unmanned Underwater Vehicle.
[2]UAV - Unmanned Aerial Vehicle.

1.3 Autonomy

ASVs are tasked with more and more complex missions and are expected to operate in diverse weather conditions. In order to not increase the operator's workload and maintain high levels of safety, a higher level of autonomy is therefore required. Providing such autonomy to a vehicle means being capable of sensing its environment and being able to navigate without human input, thus the need for a completely functional GNC. The Guidance framework is responsible to continuously generate and update smooth, feasible, and optimal trajectory commands to the control system, according to the information provided by the navigation system, assigned missions, vehicle capability and environmental conditions. The Navigation framework concentrates on identifying the ASVs current and future states (such as position, orientation, velocity, and acceleration) and its surrounding environment, based on the past and current states of the ASV, as well as environmental information including the ocean currents and wind speed obtained from its on-board sensors. The Control framework focuses on determining the proper control forces and moments to be generated in conjunction with instructions provided by the guidance and navigation systems, while at the same time satisfying desired control objectives.

1.4 Objectives

The overall objective of this thesis is to present an autonomous system for an ASV to be able to navigate safely in both static and dynamic situations. To successfully achieve this main objective, the following goals were set:

- Background literature review on Path Planning and Collision Avoidance algorithms, autonomy layers and proximity sensors.
- Background literature review on technologies for measuring and mapping and present the most adequate solution.
- Take advantage of AIS system for an augmented safety for Collision Avoidance.
- Create a system which comprises reactive methods for obstacle detection and avoidance, following COLREGS rules for dynamic obstacles, and non-compliant for static obstacles.
- Integrate this system in the open source Robot Operating System and make full use of the libraries and packages available.
- Take advantage of Gazebo software to test the vehicle and implement the algorithms.

1.5 Scope and Limitations

1.5.1 Thesis Structure

The outline of the thesis is organized as follows:

Chapter 1: Present the motivation which led to the pursuit of this thesis and delineate the objectives, the structure and the limitations.

Chapter 2: Present a literature review on the algorithms and sensors needed to provide autonomy.

Chapter 3: Present and explains the functions for the COLREGS compliant algorithm.

Chapter 4: Present the middleware and virtual simulator proposed. Explanation of some of the technical procedures on the work with an overview of the system architecture.

Chapter 5: Present the results.

Chapter 6: Conclusion and proposed future work.

1.5.2 Limitations

In order to allow an initial conceptualization of the collision avoidance system for an ASV, some assumptions and thus simplifications were applied in this thesis:

- It is assumed that all vehicles are Power-Driven Vessels, disregarding vessels of different types such as sailing vessels.
- It is also assumed that all vehicles have an AIS system fully operational, in order to permit message exchange for COLREGS collision avoidance manoeuvres. No faulty communications occur.
- No state uncertainty is considered. Both vehicles share each position, heading and pose assuming no error with proprioceptive sensors.
- All ROS simulations were done in a computer Intel$^{@}$Core$^{\mathrm{TM}}$i7-3540M CPU @ 3.00GHz $\times$ 4.

Chapter 2

Literature Review

2.1 Obstacle Detection and Avoidance

The Obstacle Detection and Avoidance (ODA) module, which integrates the guidance framework, has the function of generating the route for the vehicle. An ODA module is usually presented with two sub-modules, one for path generation, and another for reactive collision avoidance.

The path generation can be approached with search algorithms for path planning like the ones presented in (González et al., 2016), which describe the different types of algorithms, mentioning each with its advantages and disadvantages. Path planning algorithms are needed to generate a sequence of actions from a start position to a pre-specified goal position. (Campbell et al., 2012) describe the group of algorithms which belong collectively to a family of grid-searching techniques with each associated heuristic cost functions. Most search algorithms are best first search, which means if there exists one path, it will find the best possible path. However, this demands heavy computation power and, depending on the map, may take several seconds to compute. The purpose of the current work contemplates ASVs for missions where the corresponding paths are defined through waypoints. The path generation can be set up with waypoint following or path following guidance laws (Breivik and Fossen, 2009, chapter 4). After evaluating the alternatives, it can be deemed that the Path following solution may be the best when compared with waypoint following, taking into account the purpose of the mission, thus ensuring a better following autopilot (Ceccarelli and Rasmussen, 2010) is essential for bathymetry missions. Therefore, this needs to be implemented in the ASV, however it does not contemplate the main focus of the thesis. For more reference, (Fossen, 2011) should be read.

The other sub-module is the reactive collision avoidance which is the main purpose of this work. A reactive ODA, also known as a local method and as collision avoidance, considers only the immediate environment of a vehicle and does not take into account the environment currently not covered by the sensors. This property makes it excellent for dynamic environments, allowing the system to adapt readily to changes. (Kunchev et al., 2006) presents a review on the most renowned collision avoidance algorithms. A reactive algorithm has the function of modifying the trajectory of the ASV in real time when a collision is detected, and thus avoided. In this reactive sub-module, a distinction of COLREGS collision avoidance for vessels and non-COLREGS collision avoidance for static obstacles is defined through perimeters as explained in section 4.6.

2.1.1 COLREGS

The International Rules were formalized in the Convention on the International Regulations for Preventing Collisions at Sea, in 1972, and became effective on July 15, 1977, including 41 rules divided into five sections and four annexes, covering the conduct that vessels must have in order to reduce the risk of collision.

The rules which are the most correlated to the work done throughout this thesis are the rules from part B (Steering & Sailing Rules), with special emphasis in section II - Conduct of vessels in sight of one another, as presented in (Ventura, 2005):

- Rule 13 (*Overtaking*): (a) "Notwithstanding anything contained in the Rules of Part B, Sections I and II, any vessel overtaking any other shall keep out of the way of the vessel being overtaken." (b) "A vessel shall be deemed to be overtaking when coming up with another vessel from a direction more than 22.5 degrees abaft her beam, that is, in such a position with reference to the vessel she is overtaking, that at night she would be able to see only the stern light of that vessel but neither of her sidelights." (c) "When a vessel is in any doubt as to whether she is overtaking another, she shall assume that this is the case and act accordingly." (d) "Any subsequent alteration of the bearing between the two vessels shall not make the overtaking vessel a crossing vessel within the meaning of these Rules or relieve her of the duty of keeping clear of the overtaken vessel until she is finally past and clear."

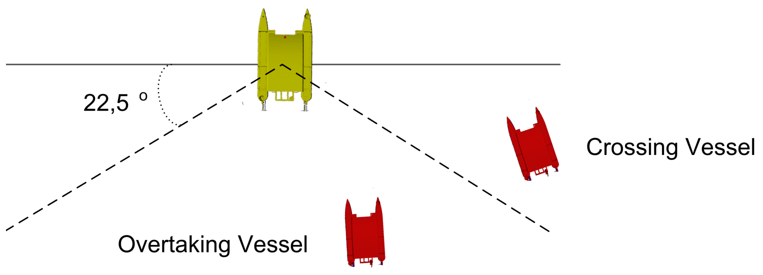


Figure 2.1: Crossing and Overtaking situations.

- Rule 14 (*Head-on situation*): (a) "When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other." (b) "Such a situation shall be deemed to exist when a vessel sees the other ahead or nearly ahead and by night she would see the mast head lights of the other in a line or nearly in a line and or both sidelights and by day she observes the corresponding aspect of the other vessel." (c) "When a vessel is in any doubt as to whether such a situation exists she shall assume that it does exist and act accordingly."

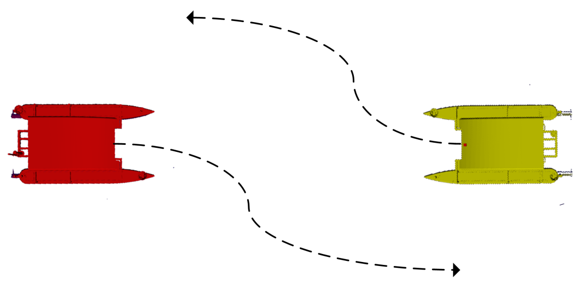


Figure 2.2: Head On representation.

- Rule 15 (*Crossing situation*):

  "When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel."

- Rule 16 (*Action by give-way vessel*):

  "Every vessel which is directed to keep out of the way of another vessel shall, so far as possible, take early and substantial action to keep well clear."

- Rule 17 (*Action by stand-on vessel*): (a) (i) "Where one of two vessels is to keep out of the way the other shall keep her course and speed." (ii) "The latter vessel may however take action to avoid collision by her manoeuvre alone, as soon as it becomes apparent to her that the vessel required to keep out of the way is not taking appropriate action in compliance with these Rules." (b) "When, from any cause, the vessel required to keep her course and speed finds herself so close that collision cannot be avoided by the action of the give-way vessel alone, she shall take such action as will best aid to avoid collision." (c) "A power-driven vessel which takes action in a crossing situation in accordance with sub-paragraph (a)(ii) of this Rule to avoid collision with another power-driven vessel shall, if the circumstances of the case admit, not alter course to port for a vessel on her own port side." (d) "This Rule does not relieve the give-way vessel of her obligation to keep out of the way."

2.1.2 Dynamic Window Approach

For ASVs, the dynamic window approach (DWA) (Fox et al., 1997) is the most commonly used, as it takes into account the dynamics of the vehicle by reducing the search space to the velocities which are reachable under the dynamic constraints. In addition to this restriction, only the velocities that are deemed safe with respect to the obstacles are considered. This correction of the search space is done in the first step of the algorithm. In the second step, the velocity which maximizes the objective function from the remaining velocities is chosen.

A simplified presentation of the different parts of the dynamic window approach is presented below. For a complete explanation of the algorithm, (Fox et al., 1997) should be read.

Different parts of the dynamic window approach

Search space: The search space of the possible velocities is reduced in three steps.

$$V_r = V_s \cap V_a \cap V_d \tag{2.1}$$

(a) Circular trajectories: The dynamic window approach considers only circular trajectories (curvatures) uniquely determined by pairs (v, w) of translational and rotational velocities. This results in a two-dimensional velocity search space.

(b) Admissible velocities: The restriction to admissible velocities ensures that only safe trajectories are considered. A pair (v, w) is considered admissible if the robot is able to stop before it reaches the closest obstacle on the corresponding curvature.

$$V_a = \{(v, w) | v \leq \sqrt{2 \times dist(v, w) \times \dot{v}_b} \wedge w \leq \sqrt{2 \times dist(v, w) \times \dot{w}_b}\} \tag{2.2}$$

(c) Dynamic window: The dynamic window restricts the admissible velocities to those that can be reached within a short time interval given the limited accelerations of the robot.

$$V_d = \{(v, w) | v \epsilon [v_a - \dot{v} \times t, v_a + \dot{v} \times t] \wedge w \epsilon [w_a - \dot{w} \times t, w_a + \dot{w} \times t]\} \tag{2.3}$$

Optimization: The objective function is maximized.

$$G(v, w) = \sigma(\alpha \times heading(v, w) + \beta \times dist(v, w) + \gamma \times vel(v, w)) \tag{2.4}$$

With respect to the current position and orientation of the robot this function trades of the following aspects:

(a) Target heading: *heading* is a measure of progress towards the goal location. It is maximal if the robot moves directly towards the target.

(b) Clearance: *dist* is the distance to the closest obstacle on the trajectory. The smaller the distance to an obstacle the higher is the robot's desire to move around it.

(c) Velocity: *vel* is the forward velocity of the robot and supports fast movements.

The function σ smooths the weighted sum of the three components and results in more side-clearance from obstacles.

2.2 Sensors

A vital part of an ASV are the sensors implemented in order to be able to navigate fully autonomous through waypoints and obstacles, which can only be achieved by continuous measurements. The sensors can be defined as proprioceptive, which measure values internally to the system, and as exteroceptive, which are used for the observation of the environment.

2.2.1 Proprioceptive Sensors

The proprioceptive sensors implemented in the ASV are an Inertial Measurement Unit (IMU) and a global navigation satellite system (GNSS). The IMU is an electronic device which provides the data that enables an automated driving system to not only know where it is, but also how it is moving to all requesting devices. It uses a combination of accelerometers and gyroscopes to measure both acceleration and angular rates of the ASV. The GNSS collects position reports from orbiting satellites and measures the time delay of the updates to estimate its own global position. The combination of the GNSS system and the IMU, provides an accurate and reliable odometry estimation of the ASV's position, attitude, heading and speed over time.

2.2.2 Exteroceptive Sensors

The exteroceptive sensors implemented in the ASV are a ranging and measurement device and an Automatic Identification System (AIS).

Technologies for measuring and mapping

The three types of technology commonly used to measure and map are: Radar, Sonar, and Lidar. The name of these technologies were given to imply the usefulness of each technology:

- Radar: ra(dio) d(etection) a(nd) r(anging)
- Sonar: so(und) na(vigation) (and) r(anging)
- Lidar: li(ght) d(etection) a(nd) r(anging)

For these sensors, the same principle applies: a wave of its type is emitted and, when encountering an object, it is reflected back. Knowing the speed of the wave type and the time it took to be bounced backwards, the distance of the object can be estimated.

Due to the properties of each technology, some are more suitable in some environments than others. Table 2.1 presents some advantages and disadvantages.

Even though many applications with Lidar in autonomous vehicles have been developed and documented, some examples of ASVs with radar and sonar based collision avoidance have been pursued, (Almeida et al., 2009) and (Heidarsson and Sukhatme, 2011) respectively.

Table 2.1: Advantages and limitations of the detection and ranging sensors with application to ASVs (Liu et al., 2016).

Sensors	Advantages	Limitations
Radar	1) Long detecting range; 2) Provides nearly all-weather and broad-area imagery; 3) High depth resolution and accuracy.	1) Skewed data from fast turning maneuvers; 2) Limited small and dynamic targets detection capability; 3) Susceptible to high waves and water reflectivity.
Lidar	1) Good at near range obstacle detection; 2) High depth resolution and accuracy.	1) There exists sensor noise and calibration errors; 2) Sensitive to environment and USV motion.
Sonar	1) No visual restrictions; 2) High depth resolution and accuracy.	1) Limited detecting range in each scanning; 2) Impressionable to the noise from near-surface.

Surface

When surfaced, Lidar has been increasingly chosen compared to others, with the main reasons being that the measurements are more precise, and the speed and accuracy of the laser pulses from Lidar sensors, in which data can be collected faster and with utmost accuracy. These reasons make it preferred in high capacity and data intensive applications. In this work, the technology for measuring and ranging is applied for near range collision avoidance, (as described in section 4.6), and thus the choice for the Lidar. The greatest flaw for Lidar is that it is adversely affected by smoke, rain and fog. When using a Lidar, we must have this flaw in mind as it might restrict the operation.

Radar on the other hand, can operate in cloudy weather. Another characteristic of Radar technology is that it has a longer operating distance, although it takes a longer time to return data regarding the distance of the object. The main flaws of Radar are the fact that it does not allow the detection of smaller objects due to longer wavelength, it cannot distinguish multiple targets within a surface that are closely entangled together, and since it is not a beam, but a wave, it is susceptible to water reflectivity.

The use of Radar in a ASV should be used as a complement of the Lidar for adverse weather situations and to further expand the map/grid.

Underwater

In the underwater domain, radio waves and vision suffer from inherent limitations. Radio waves are virtually useless underwater due to their high attenuation, while vision effectiveness is restricted to a range of a few meters, and highly dependent on the turbidity of the water. This is caused by the scattering effect of light by suspended matter. One method could be employing a higher intensity light source to offset the light attenuation, but this only results in a massive power drain, thus both *Radar* and *Lidar* technologies are discarded for this domain.

Unlike radio-waves and optical energy, sound transmission is the single most effective means of directing energy transfer over long distances in sea-water. Consequently, *Sonar* is largely employed underwater.

The Automatic Identification System

The Automatic Identification System (AIS) is a tracing and tracking system that allows a vessel to post information about itself and receive information about other vessels in the area. The typical range of the AIS depends on the altitude of the external antenna, being a reference to have a range of $15 - 20$ nautical miles[1] for an altitude of 15 meters. The international Maritime Organization (IMO) defined that commercial ships above 300 tons under international navigation, all passenger ships and fishing ships above 16 meters in length are all obliged to have an AIS integrated (United States Coast Guard, 2017). Due to the AIS's droppage in price and aptitude for collision avoidance, there has been an increase in its usage. The AIS transmits both dynamic and static information, with some of the information consisting (MarineTraffic, 2017b):

- Dynamic information:
  - Navigation status – "at anchor", "under way using engine(s)", "not under command", etc.
  - Rate of turn
  - Speed over ground
  - Positional accuracy
    - Longitude
    - Latitude
  - Course over ground
  - Heading
- Static information:
  - Name – 20 characters to represent the name of the vessel
  - Navigational status
  - Type of ship/cargo
  - Dimensions of ship – to the nearest meter

An example of a typical AIS Class A position report could be:

$$!AIVDM, 1, 1, , A, 13u?etPv2; 0n : dDPwUM1U1Cb069D, 0 * 24$$

The data string contains different fields of data segregated by commas. A complete description of each field is presented in (Raymond, 2016), and for a detailed description on the AIS system and the types of data that are transmitted (MarineTraffic, 2017a) should be read.

Several online AIS decoders exist[2], providing tools to interpret AIS messages, where the user can paste a raw AIS string and receive the underlying data in a readable format.

In this work, for simulation and testing, ROS topics and messages were used with the corresponding AIS information with a message exchange of $1Hz$ which is the typical frequency for dynamic AIS messages exchange.

[1] $1\ nautical\ mile = 1.85200\ kilometers$

[2] An example: `http://www.maritec.co.za/tools/aisvdmvdodecoding/`

Chapter 3

COLREGS Collision Avoidance

The AIS system presented in subsubsection 2.2.2 provides information about the position, pose and speed, allowing the sharing of information between vehicles equipped with AIS. The three data messages mentioned are the most important for the collision detection modes as will be presented.

Having a representation of both our vehicle, which will be, from now on, addressed as *ownship*, and the obstacle vehicle, which will be addressed as *contact*, it is possible to detect whether a collision is in risk of occurring and which COLREGS rule to follow. The terms and algorithms presented through this chapter were adapted from (Raymond, 2016).

The AIS position data shared are in latitude and longitude, which then need to be converted to a local x-y coordinate in meters, where $< x, y > \epsilon \mathbb{R}^2$. The heading is given in degrees, where $\theta \, \epsilon \, [0, 360)$ in the Earth-fixed reference frame (subsection D.3.1), with the reference being North. The speed is given in meters per second, with $v \, \epsilon \, \mathbb{R}$.

Figure 3.1 represents both *ownship* and *contact* with each local coordinates, heading and velocity.

The heading for each case is the direction that the vehicle is pointing and not the direction the vehicle is moving, which could be different due to external forces such as in wind and current situations. As pointed out in subsubsection 2.2.2, the dynamic information exchanged with an AIS system is roughly $1Hz$, which guarantees reliable instantaneous data from both vessels.

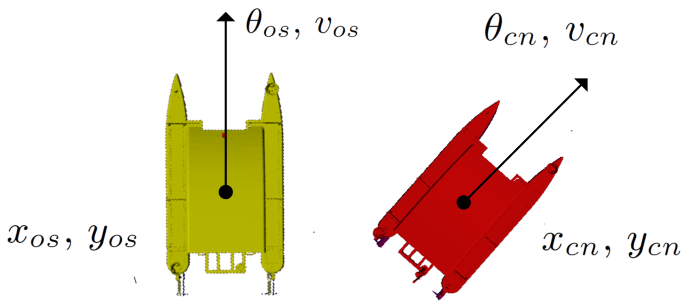


Figure 3.1: *Ownship* and *contact* with each local coordinates, heading and velocity, represented with yellow and red color respectively. Adapted from (Raymond, 2016).

3.1 Useful Operators on Heading Values

The heading values for each vehicle can be presented in the interval range of $[0, 360)$ or $[-180, 180)$. However sometimes they are presented outside these domains. The following conditions guarantee the necessary conversion to the desired domain and limit, using the floor function condition "$\lfloor\ \rfloor$" [1].

$$[\theta]^{360} = \begin{cases} \theta - \lfloor \frac{\theta}{360} \rfloor \times 360 & (\theta \geq 0) \\ \theta + (\lfloor \frac{-\theta}{360} \rfloor + 1) \times 360 & \text{otherwise} \end{cases} \tag{3.1}$$

$$[\theta]^{180} = \begin{cases} \theta - \lfloor \frac{\theta+180}{360} \rfloor \times 360 & (\theta \geq 0) \\ \theta + (\lfloor \frac{-\theta+180}{360} \rfloor + 1) \times 360 & \text{otherwise} \end{cases} \tag{3.2}$$

Another definition which helps to simplify some work is the *heading deviation*, which gives the absolute value of the difference between two directions.

$$\delta(\theta_1, \theta_2) = |[\theta_1 - \theta_2]^{180}| \tag{3.3}$$

3.2 Range, Absolute Bearing and Relative Bearing

In order to define the distance between the vehicles, the bearing of *ownship* to *contact*, and of *contact* to *ownship*, the following terms are essential:

- *range* between *ownship* and *contact*: r^{os}_{cn}
- *absolute bearing* from *ownship* to *contact*: bng^{os}_{cn}
- *relative bearing* from *ownship* to *contact*: $relbng^{os}_{cn}$, or simply β
- *absolute bearing* from *contact* to *ownship*: bng^{cn}_{os}
- *relative bearing* from *contact* to *ownship*: $relbng^{cn}_{os}$, or simply α

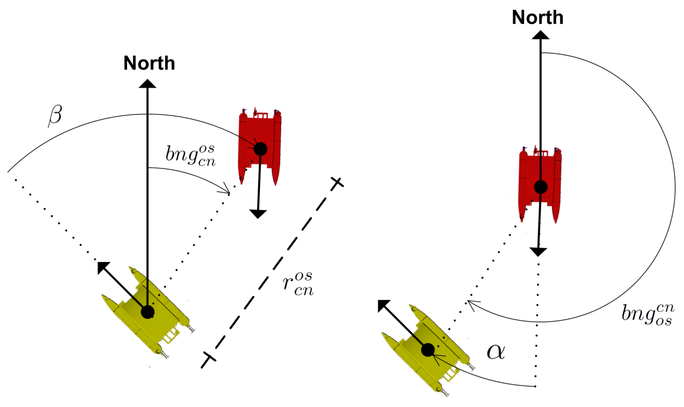


Figure 3.2: Bearing relations for *ownship* and *contact* represented with yellow and red color respectively. Adapted from (Raymond, 2016).

[1] Floor function - function that takes as input a real number θ and gives as output the greatest integer less than or equal to θ

Range is the linear distance between the vehicles at the current instance of time, and is thus defined as the the *Euclidean* distance between both vehicles position, which is directly obtained from *Pythagorean theorem*:

$$range = r_{cn}^{os} = \sqrt{(x_{os} - x_{cn})^2 + (y_{os} - y_{cn})^2} \tag{3.4}$$

The absolute bearing is the angle of one vehicle to the other's position, regardless of the reference vehicle orientation, with the North reference being 0 degrees and East 90 degrees. The absolute bearing from *ownship* to *contact* is obtained as follows:

$$bng_{cn}^{os} = \begin{cases} 0 & (x_{os} = x_{cn}) \text{ and } (y_{os} \leq y_{cn}) \\ 180 & (x_{os} = x_{cn}) \text{ and } (y_{os} > y_{cn}) \\ 90 - tan^{-1}(\frac{|y_{os} - y_{cn}|}{|x_{os} - x_{cn}|}) * \frac{180}{\pi} & (x_{os} < x_{cn}) \text{ and } (y_{os} \leq y_{cn}) \\ tan^{-1}(\frac{|y_{os} - y_{cn}|}{|x_{os} - x_{cn}|}) * \frac{180}{\pi} + 90 & (x_{os} < x_{cn}) \text{ and } (y_{os} > y_{cn}) \\ 270 - tan^{-1}(\frac{|y_{os} - y_{cn}|}{|x_{os} - x_{cn}|}) * \frac{180}{\pi} & (x_{os} > x_{cn}) \text{ and } (y_{os} > y_{cn}) \\ tan^{-1}(\frac{|y_{os} - y_{cn}|}{|x_{os} - x_{cn}|}) * \frac{180}{\pi} + 270 & (x_{os} > x_{cn}) \text{ and } (y_{os} \leq y_{cn}) \end{cases} \tag{3.5}$$

The relative bearing is similar to the absolute bearing, however the reference is now the vehicle's orientation. The relative bearing from *ownship* to *contact* can then be calculated:

$$relbng_{cn}^{os} = \beta = [bng_{cn}^{os} - \theta_{os}]^{360} \tag{3.6}$$

Both equations 3.5 and 3.6 are similarly used to calculate from *contact* to *ownship*, inverting the local coordinates respectively.

3.3 Range Rate and Bearing Rate

Other terms of utmost importance are the range rate, which represents how quickly the two vehicles are moving towards or away from one another, and the bearing rate, which is the rate of change in the relative bearing, from *ownship* to *contact*, in degrees per second. Both are given by the following equations:

$$range\, rate = \dot{r} = v_{cn}^{os} + v_{os}^{cn} \tag{3.7}$$

Being v_{cn}^{os} the velocity of *ownship* in the direction of *contact's* current position and v_{os}^{cn} the velocity of the *contact* in direction of *ownship's* current position.

$$v_{os}^{cn} = \cos(\alpha) \times v_{os} \tag{3.8a}$$

$$v_{cn}^{os} = \cos(\beta) \times v_{cn} \tag{3.8b}$$

$$bearing\, rate = \dot{\beta} = -(v_{tn}^{os} + v_{tn}^{cn}) \times \frac{360}{2r\pi} \tag{3.9}$$

Being v_{tn}^{os} and v_{tn}^{cn} the *ownship* and *contact* speed in the direction of the *tangent heading* consequently.

$$v_{tn}^{os} = \cos(\theta_{tn}^{os}) \times v_{os} \tag{3.10a}$$

$$v_{tn}^{cn} = \cos(\theta_{tn}^{cn}) \times v_{cn} \tag{3.10b}$$

The *tangent heading* is the heading given the current *ownship* to *contact* absolute bearing, plus 90 degrees:

$$\theta_{tn} = [bng^{os}_{cn} + 90]^{360} \tag{3.11}$$

Therefore, the speed of *ownship* and *contact* in the tangent heading is given by:

$$\theta^{os}_{tn} = \delta(\theta_{os}, \theta_{tn}) \tag{3.12a}$$

$$\theta^{cn}_{tn} = \delta(\theta_{cn}, \theta_{tn}) \tag{3.12b}$$

3.4 Relative Boolean Positions

3.4.1 Fore and Aft

The boolean variables define whether the vehicles are either in front or behind the other.

$$fore^{os}_{cn} = \begin{cases} \text{false} & 90 < \alpha < 270 \\ \text{true} & \text{otherwise} \end{cases} \tag{3.13a}$$

$$aft^{os}_{cn} = \begin{cases} \text{true} & 90 \leq \alpha \leq 270 \\ \text{false} & \text{otherwise} \end{cases} \tag{3.13b}$$

$$fore^{cn}_{os} = \begin{cases} \text{false} & 90 < \beta < 270 \\ \text{true} & \text{otherwise} \end{cases} \tag{3.13c}$$

$$aft^{cn}_{os} = \begin{cases} \text{true} & 90 \leq \beta \leq 270 \\ \text{false} & \text{otherwise} \end{cases} \tag{3.13d}$$

3.4.2 Port and Starboard

The boolean variables define whether the vehicles are in port side or starboard side from one another.

$$port^{os}_{cn} = \begin{cases} \text{false} & 0 \leq \alpha \leq 180 \\ \text{true} & \text{otherwise} \end{cases} \tag{3.14a}$$

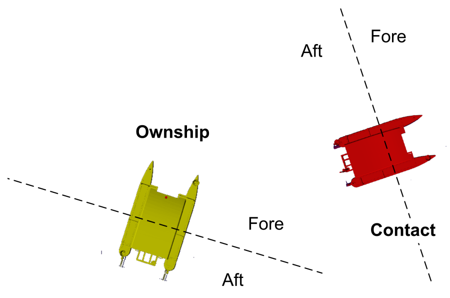
$$star^{os}_{cn} = \begin{cases} \text{true} & 0 \leq \alpha \leq 180 \\ \text{false} & \text{otherwise} \end{cases} \tag{3.14b}$$

$$port^{cn}_{os} = \begin{cases} \text{false} & 0 \leq \beta \leq 180 \\ \text{true} & \text{otherwise} \end{cases} \tag{3.14c}$$

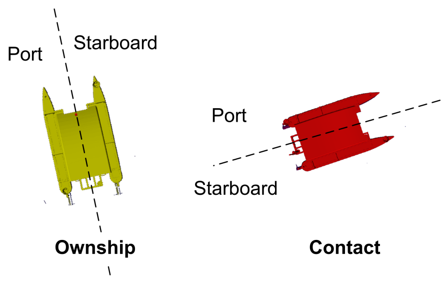
$$star^{cn}_{os} = \begin{cases} \text{true} & 0 \leq \beta \leq 180 \\ \text{false} & \text{otherwise} \end{cases} \tag{3.14d}$$

3.4.3 Representations

For better understanding of the boolean conditions, two representations are given in figures 3.3(a) and 3.3(b).



(a) Fore and Aft



(b) Port and Starboard

Figure 3.3: Boolean examples of *ownship* and *contact*, represented with yellow and red color respectively. Adapted from (Raymond, 2016).

In figure 3.3(a), it is represented a situation where *ownship* is aft of *contact*, $aft^{os}_{cn} = true$, and *contact* is fore of *ownship*, $fore^{cn}_{os} = True$. In figure 3.3(b), is represented a situation where *ownship* is port of *contact*, $port^{os}_{cn} = true$, and *contact* is in starboard of *ownship*, $star^{cn}_{os} = true$.

3.5 Crossing Relationships

To decide whether a crossing maneuver is to be executed, three boolean relationships are necessary, thus the following terms are presented.

The ownship gamma heading θ_{γ}^{os} is the direction from *ownship* making a 90 degree angle to the *contact's* bow-stern line.

$$\theta_{\gamma}^{os} = \begin{cases} [\theta_{cn} + 90]^{360} & port_{cn}^{os} \\ [\theta_{cn} - 90]^{360} & \text{otherwise} \end{cases} \tag{3.15}$$

The *ownship* gamma velocity v_{γ}^{os} is the velocity at which *ownship* closes on *contact's* bow-stern line.

$$v_{\gamma}^{os} = \cos(\theta_{os} - \theta_{\gamma}^{os}) \times v_{os} \tag{3.16}$$

The *ownship* gamma range r_{γ} is the current range between *ownship* and the *contact's* bow-stern line.

$$r_{\gamma} = r \times \cos(\theta_{\gamma}^{os} - relbng_{cn}^{os}) \tag{3.17}$$

3.5.1 Boolean Relationships

The three boolean relationships can thus be defined.

The first is $cross_{xcn}^{os}$, which is true only when the *ownship* crosses the *contact's* bow-stern line, regardless of whether it is aft or fore, given the current position and linear trajectory of both vehicles.

$$cross_{xcn}^{os} = \begin{cases} \text{true} & (\alpha = 0) \text{ or } (\alpha = 180) \\ \text{true} & (v_{\gamma}^{os} > 0) \\ \text{false} & \text{otherwise} \end{cases} \tag{3.18}$$

Another boolean relationship is $cross_{xcnb}^{os}$, which is true only when the *ownship* crosses the *contact's* bow-stern line, and is fore of the *contact* at the moment of crossing, given the current position and linear trajectory of both vehicles.

$$cross_{xcnb}^{os} = \begin{cases} \text{true} & (\alpha = 0) \\ \text{true} & (v_{\gamma}^{os} > 0) \text{ and } (port_{cn}^{os}) \text{ and } (\dot{\beta} > 0) \\ \text{true} & (v_{\gamma}^{os} > 0) \text{ and } (star_{cn}^{os}) \text{ and } (\dot{\beta} < 0) \\ \text{false} & \text{otherwise} \end{cases} \tag{3.19}$$

The last is $cross_{xcns}^{os}$, which is true only when the *ownship* crosses the *contact's* bow-stern line, and is aft of the *contact* at the moment of crossing, given the current position and linear trajectory of both vehicles.

$$cross_{xcns}^{os} = \begin{cases} \text{true} & (\alpha = 180) \\ \text{true} & (v_{\gamma}^{os} > 0) \text{ and } (port_{cn}^{os}) \text{ and } (\dot{\beta} < 0) \\ \text{true} & (v_{\gamma}^{os} > 0) \text{ and } (star_{cn}^{os}) \text{ and } (\dot{\beta} > 0) \\ \text{false} & \text{otherwise} \end{cases} \tag{3.20}$$

3.5.2 Numerical Relationships

The crossing numerical relations are necessary for the collision mode algorithms. It comprises of range and time values.

The *ownship* crossing time t_{γ}^{os} is the time in seconds in which the *ownship* will cross the *contact*.

$$t_{\gamma}^{os} = \frac{r_{\gamma}}{v_{\gamma}^{os}} \tag{3.21}$$

Knowing the *ownship's* crossing time, it is possible to calculate the distance that the *contact* travels, in the time between now and up to the time that the *ownship* crosses the *contact's* bow-stern line, r_{ϵ}^{xcn}, and also the distance that *ownship* travels in this same time, r_{ϵ}^{xos}, in direction of the *contact's* heading.

$$r_{\epsilon}^{xcn} = t_{\gamma}^{os} \times v_{cn} \tag{3.22}$$

$$r_{\epsilon}^{xos} = t_{\gamma}^{os} \times v_{cnh}^{os} \tag{3.23}$$

Depending on the active crossing boolean relation, the range of crossing the *contact's* bow or stern is calculated.

The *ownship* crossing bow range r_{xcnb}^{os} is the range in which the *ownship* crosses the *contact's* bow.

$$r_{xcnb}^{os} = \begin{cases} r_{\epsilon}^{xos} - (r_{\epsilon} - r_{\epsilon}^{xcn}) & cross_{xcnb}^{os} \\ -1 & \text{otherwise} \end{cases} \tag{3.24}$$

The *ownship* crossing stern range r_{xcns}^{os} is the range in which the *ownship* crosses the *contact's* stern.

$$r_{xcnb}^{os} = \begin{cases} (r_{\epsilon} - r_{\epsilon}^{xcn}) - r_{\epsilon}^{xos} & cross_{xcns}^{os} \\ -1 & \text{otherwise} \end{cases} \tag{3.25}$$

3.6 Passing Relationships

Just as the crossing relationships defined above, it is necessary to define passing relationships, thus the following terms are required.

The *ownship* epsilon heading θ_{ϵ}^{os} is the *ownship* heading perpendicular and toward the *contact's* beam.

$$\theta_{\epsilon}^{os} = \begin{cases} [\theta_{cn} + 180]^{360} & fore_{cn}^{os} \\ \theta_{cn} & \text{otherwise} \end{cases} \tag{3.26}$$

The *ownship contact* velocity v_{cnh}^{os} is the speed of *ownship* in the direction of *contact's* heading.

$$v_{cnh}^{os} = \cos(\theta_{os} - \theta_{\epsilon}^{os}) \times v_{os} \tag{3.27}$$

The *ownship* epsilon velocity v_{ϵ}^{os} is the speed at which *ownship* is closing on the *contact's* beam.

$$v_{\epsilon}^{os} = \begin{cases} v_{cn} + v_{cnh}^{os} & fore_{cn}^{os} \\ v_{cnh}^{os} - v_{cn} & \text{otherwise} \end{cases} \tag{3.28}$$

The *ownship* epsilon range r_{ϵ} is the current range between both vehicle's beam.

$$r_{\epsilon} = r \times \cos(\theta_{\epsilon}^{os} - relbng_{cn}^{os}) \tag{3.29}$$

3.6.1 Boolean Relationships

The three boolean relationships can thus be defined.

The first is $pass_{cn}^{os}$, which is true only when the *ownship* crosses the *contact's* beam, regardless of port or starboard, given the current linear trajectory of both vehicles.

$$pass_{cn}^{os} = \begin{cases} \text{true} & (v_{\epsilon}^{os} > 0) \\ \text{false} & \text{otherwise} \end{cases} \tag{3.30}$$

Another boolean relationship is $pass_{cnp}^{os}$, which is true only when the *ownship* crosses the *contact's* port beam, given the current position and linear trajectory of both vehicles.

$$pass_{cnp}^{os} = \begin{cases} \text{true} & aft_{cn}^{os} \text{ and } pass_{cn}^{os} \text{ and } (\dot{\beta} > 0) \\ \text{true} & fore_{cn}^{os} \text{ and } pass_{cn}^{os} \text{ and } (\dot{\beta} < 0) \\ \text{false} & \text{otherwise} \end{cases} \tag{3.31}$$

The last is $pass_{cns}^{os}$, which is true only when the *ownship* passes the *contact's* starboard beam, given the current position and linear trajectory of both vehicles.

$$cross_{cns}^{os} = \begin{cases} \text{true} & aft_{cn}^{os} \text{ and } pass_{cn}^{os} \text{ and } (\dot{\beta} < 0) \\ \text{true} & fore_{cn}^{os} \text{ and } pass_{cn}^{os} \text{ and } (\dot{\beta} > 0) \\ \text{false} & \text{otherwise} \end{cases} \tag{3.32}$$

3.6.2 Numerical Relationships

Likewise the crossing numerical relations, the passing numerical relations are necessary for the collision mode algorithms.

The *ownship* passing time t_{ϵ}^{os} is the time in seconds in which the *ownship* will pass the *contact*.

$$t_{\epsilon}^{os} = \frac{r_{\epsilon}}{v_{\epsilon}^{os}} \tag{3.33}$$

The distance travelled by the *ownship* in the direction of θ_{γ}^{os}, perpendicular to the *contact's* bow-stern line, is represented with r_{γ}^{pos} which is useful in calculating the range between *ownship* and *contact* at the moment of passing.

$$r_{\gamma}^{pos} = t_{\epsilon}^{os} \times v_{\epsilon}^{os} \tag{3.34}$$

Depending on the active passing boolean relation, the range of passing the *contact's* port or starboard is calculated.

The *ownship* passing port range r_{pcnp}^{os} is the range at which the *ownship* passes the *contact's* port beam.

$$r_{pcnp}^{os} = \begin{cases} r_{\gamma} - r_{\gamma}^{pos} & port_{cn}^{os} \text{ and } (r_{\gamma}^{pos} \leq r_{\gamma}) \\ r_{\gamma}^{pos} - r_{\gamma} & star_{cn}^{os} \text{ and } (r_{\gamma} > r_{\gamma}^{pos}) \\ -1 & \text{otherwise} \end{cases} \tag{3.35}$$

The *ownship* passing starboard range r^{os}_{pcns} is the range at which the *ownship* passes the *contact's* starboard beam.

$$r^{os}_{pcns} = \begin{cases} r_\gamma - r^{pos}_\gamma & star^{os}_{cn} \text{ and } (r^{pos}_\gamma \leq r_\gamma) \\ r^{pos}_\gamma - r_\gamma & port^{os}_{cn} \text{ and } (r_\gamma > r^{pos}_\gamma) \\ -1 & \text{otherwise} \end{cases} \tag{3.36}$$

3.7 Closest Point of Approach

The closest point of approach (CPA) is the point where *ownship* and *contact* have minimal range during the course of a maneuver. On ships, CPA is often reported with range, time, and bearing components, but here it is used with the terms CPA and r_{CPA} as the range at the time of CPA. This value is directly calculated once the time of CPA (t_{CPA}) is known, by the following:

$$CPA = r(t_{CPA}) = \sqrt{k_2 \times {t_{CPA}}^2 + k_1 \times t_{CPA} + k_0} \tag{3.37}$$

$$t_{CPA} = \begin{cases} 0 & \dot{r} \geq 0 \\ \frac{-k_1}{2 \times k_2} & \text{otherwise} \end{cases} \tag{3.38}$$

Where k_0, k_1 and k_2 are given by the following equations:

$$\begin{aligned} k_2 = \; & \cos^2(\theta_{os}) \times v_{os}^2 - 2 \times \cos(\theta_{os}) \times v_{os} \times \cos(\theta_{cn}) \times v_{cn} \\ & + \cos^2(\theta_{cn}) \times v_{cn}^2 + \sin^2(\theta_{os}) \times v_{os}^2 - \\ & 2 \times \sin^2(\theta_{os}) \times v_{os} \times \sin(\theta_{cn}) \times v_{cn} + \sin^2(\theta_{cn}) \times v_{cn}^2 \end{aligned} \tag{3.39}$$

$$\begin{aligned} k_1 = \; & 2 \times \cos(\theta_{os}) \times v_{os} \times y_{os} - 2 \times \cos(\theta_{os}) \times v_{os} \times y_{cn} - 2 \times y_{os} \times \cos(\theta_{cn}) \times v_{cn} + \\ & 2 \times \cos(\theta_{cn}) \times v_{cn} \times y_{cn} + 2 \times \sin(\theta_{os}) \times v_{os} \times x_{os} - \\ & 2 \times \sin(\theta_{os}) \times v_{os} \times x_{cn} - 2 \times x_{os} \times \sin(\theta_{cn}) \times v_{cn} + 2 \times \sin(\theta_{cn}) \times v_{cn} \times x_{cn} \end{aligned} \tag{3.40}$$

$$k_0 = y_{os}^2 - 2 \times y_{os} \times y_{cn} + y_{cn}^2 + x_{os}^2 - 2 \times x_{os} \times x_{cn} + x_{cn}^2 \tag{3.41}$$

3.8 Risk of Collision

Other vital definitions are $\check{r}_{cpa}$ and $\hat{r}_{cpa}$, which represent the thresholds in which beyond it is considered to be or not to be in risk of collision respectively.

In (Vujicic et al., 2017), it is presented a research with the aim of determining the CPA for the ideal distance to initialize the safety maneuver and the minimum distance the vessels should approach one another. This research included only ship officers and captains with, at least, one year of navigation, and thus led to some conclusions. $M_{cpa,min}$ should be between $[1.6 - 2.5]$ and for implementation it will be used as $M_{cpa,min} = 2$. M_{cpa} should be between $[5.1 - 8.0]$, and for implementation it will be used as $M_{cpa} = 6.5$.

With the AIS data transferred from the *contact*, we obtain the vehicle length. The $\check{r}_{cpa}$ and $\hat{r}_{cpa}$ can then be obtained as follows:

$$\check{r}_{cpa} = r_{cpa,min} = M_{cpa,min} \times Length_{contact} \tag{3.42}$$

$$\hat{r}_{cpa} = r_{cpa,max} = M_{cpa} \times Length_{contact} \tag{3.43}$$

3.9 Collision Avoidance Modes

Once all the parameters are established, it is possible to distinguish the encounter type. The possible modes and the respective COLREGS rules being followed (subsubsection 2.1.1) are:

- GiveWayOT, or GiveWay Overtaking Mode (Rules 13 and 16)
- HeadOn Mode (Rule 14)
- StandOnOT, or StandOn Overtaken Mode (Rules 13 and 17)
- GiveWayX, or GiveWay Crossing Mode (Rules 15 and 16)
- StandOnX, or StandOn Crossing Mode (Rules 15 and 17)
- CPA, or Closest Point Approach Mode
- Null

The corresponding algorithms for each were adapted from (Raymond, 2016) and presented in appendix B.

3.10 Collision Avoidance Objective Functions

After having the COLREGS mode defined, an objective function must be applied in order to determine the most suitable combination of heading and velocity. For the Stand On modes, the heading and velocity should be maintained. As for the others, an objective function is applied.

The objective function has two parameters: the risk function and the heuristic function. The target is to generate the ideal combination of heading and velocity by maximizing the function's value.

$$f(\theta, v) = g(cpa(\theta, v)) \times h(\theta, v) \tag{3.44}$$

For all the combination of admissible headings and velocities, the CPA function defines the r(CPA) which being greater than $\hat{r}_{cpa}$, the risk function outputs the max value (100). Being less than $\check{r}_{cpa}$, the output is 0, meaning it is a combination of heading and velocity that must not be pursued. When in between $\check{r}_{cpa}$ and $\hat{r}_{cpa}$, the output value is now a linear interpolation from these limits.

$$g(x) = \begin{cases} 100 & x \geq \hat{r}_{cpa} \\ 0 & x \leq \check{r}_{cpa} \\ \dfrac{x - \check{r}_{cpa}}{(\hat{r}_{cpa} - \check{r}_{cpa})} & \text{otherwise} \end{cases} \tag{3.45}$$

In case that the objective function only contains the risk function, the output combination of heading and velocity would present the least risk. However, it could not be complying with COLREGS rules. When not following the rules, both vehicles could give away to the same direction and then collide, thus the heuristic function, which constrains the possible headings to be only mode compliant.

The heuristic function depends on the COLREGS mode defined. If the COLREGS mode is CPA mode, the heuristic presents all values in the array with value 1, as it is the critical situation and disregards the heuristic component. On the other hand, Crossing mode, Head On mode and Passing mode have each its unique function.

3.10.1 The GiveWay Overtaking Heuristic Function

The heuristic function for GiveWay Overtaking mode also has submodes. For the combination of heading and velocities, which have the boolean conditions of $pass^{os}_{cnp}$ and in port submode, or $pass^{os}_{cns}$ and in starboard submode, the heuristic outputs the true condition of 1. Otherwise, it would be disrespecting the COLREGS rules, and thus the false condition of 0.

$$h(\theta, v) = \begin{cases} 1 & \text{(submode = port) and } pass^{os}_{cnp} \\ 1 & \text{(submode = starboard) and } pass^{os}_{cns} \\ 0 & \text{otherwise} \end{cases} \tag{3.46}$$

3.10.2 The Head-On Heuristic Function

The Head-On Heuristic function only has the condition of whether the *ownship* is crossing the *contact's* port beam. For the combination of heading and velocities making the boolean condition true, the heuristic also defines as true with output 1.

$$h(\theta, v) = \begin{cases} 1 & pass^{os}_{cnp} \\ 0 & \text{otherwise} \end{cases} \tag{3.47}$$

3.10.3 The GiveWay Crossing Heuristic Function

The GiveWay Crossing Heuristic function, like the Passing heuristic function, depends on two submodes, the Bow submode and Stern submode.

Bow Submode

$$h(\theta, v) = \begin{cases} 0 & turn^{os}_{star} \text{ and } !cross^{os}_{xcnb} \\ 1 & \text{otherwise} \end{cases} \tag{3.48}$$

Stern Submode

$$h(\theta, v) = \begin{cases} 0 & turn^{os}_{port} \\ 0 & cross^{os}_{xcnb} \\ 1 & \text{otherwise} \end{cases} \tag{3.49}$$

Chapter 4

Implementation

The solution presented, takes advantage of the power of a middleware and a virtual simulator in order to apply the control module and to implement the collision avoidance algorithm. Autonomous robots, such as ASVs, are complex systems that require an interaction between numerous heterogeneous components (software and hardware). The increase in the complexity of robotic applications and the diverse range of hardware available, has led to the choice of a robotic middleware, which has the advantages such as: it is usually designed to manage the complexity and heterogeneity of the hardware and applications; promote the integration of new technologies; simplify software design; hide the complexity of low-level communication; make use of different heterogeneity of sensors; improve software quality; reuse robotic software infrastructure across multiple research efforts; and reduce production costs. (Magyar and Sincak, 2015) presents the main robotics middleware available, comparing each perks and flaws. Robot Operating System (ROS) can be said to be the most popular robotic middleware, having more components and supporting more robots than other middlewares. Some weaknesses of ROS are the fact that Windows is not among the supported operating systems, and the fact that it does not have a graphical IDE. Such factors, led to the choice of using ROS with the operating system being *Linux*, an *Ubuntu* version 16.04.

4.1 Robot Operating System

The Robot Operating System (ROS) is an open-source middleware used for robotics, which provides services, tools, and libraries for obtaining, building, writing and running code across multiple computers. With an increasing community, several packages which aid the implementation of autonomy to the vehicle such as *tf* and *navigation* stack are implemented in this work. Some concepts and language are needed, in order to understand some of the technical descriptions used throughout the work presented on this thesis are:

- **Nodes:** *Nodes* are independent processes that perform computation. A robot/vehicle has several *nodes* which communicate with each other over *topics*. A sensor becomes a *node*, which is independent, and communicates with whoever *subscribes* to the *topics* that the sensor is *publishing*.

- **Messages:** *Nodes* communicate with each other by passing *messages*. A *message* is simply a data structure, comprised of typed fields, such as integer, floating point, boolean, etc..

- **Topics:** *Messages* which are routed via a transport system with a *publish/subscribe* semantics. *Topics* is the name which is used to identify the content of the *message*.

- **Services:** A *Request/Reply* structure used by *nodes* with a defined pair of *messages* structures.

- **Master:** The ROS *Master* provides name registration and look-up to the rest of the Computation Graph. Without the *Master*, *nodes* would not be able to find each other, exchange *messages*, or invoke *services*. Example on figure 4.1.

- **Package:** Software in ROS is organized in packages. A package might contain ROS nodes, a ROS-independent library, a dataset, configuration files, a third-party piece of software, or anything else that logically, constitutes a useful module. The goal of these packages is to provide this useful functionality in an easy-to-consume manner, so that software can be easily reused (Open Source Robotics Foundation, 2015).

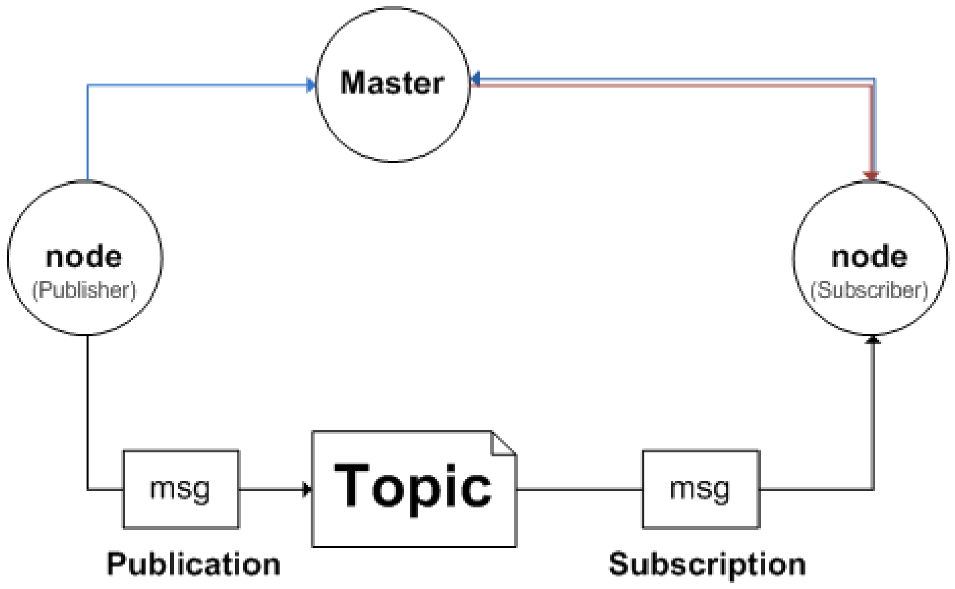


Figure 4.1: ROS technical representation. The node on the left publishes messages to a determined topic and advertises to the Master. The node on the right asks the Master the path to the desired topic and, knowing the path, it is thus subscribed.

Programming in the ROS framework can be said to be generally language-independent, with the most popular languages being C++ and Python, (Open Source Robotics Foundation, 2018b).

A list of tutorials presenting detailed information about each concept can found in (Open Source Robotics Foundation, 2018a).

4.2 Gazebo

Having the autonomous system built, a virtual simulator such as Gazebo is essential to permit further tests and to debug the algorithms employed. Gazebo is the most compatible virtual simulator for ROS, being both from the Open Source Robotics Foundation. In Gazebo's home page[1], the reason presented for Gazebo's usage, is that "Robot simulation is an essential tool in every roboticist's toolbox". Being open-source, it contributes to an increasing community, and the fact that it is easily integrated with ROS, provides a powerful simulating environment to promptly test the algorithms, design robots, perform regression testing, and train AI systems using realistic scenarios.

When using Gazebo, multiple scenarios were created with an open sea as the environment. Static and dynamic obstacles were implemented in order to assess and authenticate the various circumstances that a vehicle can encounter. Gazebo was used in order to test and debug the algorithms while being confident that, when applied in the physical model, the algorithms behaviour will be exactly the same.

4.3 Obstacles

The obstacles implemented in the Gazebo simulation are either vessels or static objects, as presented in section 1.2. These obstacles can be seen in figure 4.2.

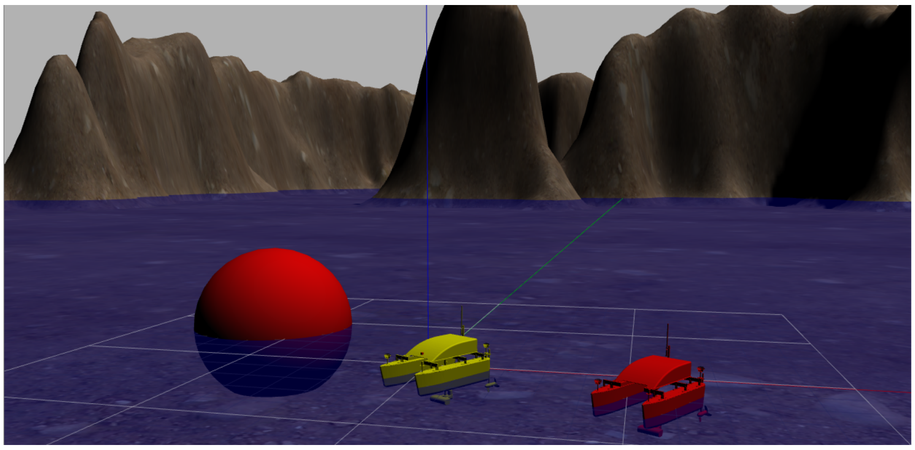


Figure 4.2: Obstacles used in the simulation. Static obstacle in the left, representing all static obstacles, the *ownship* in the middle, and the *contact* vehicle in red in the right, representing the dynamic obstacle.

For simplicity, the static obstacles are represented as buoys, which will occupy an area in the map and thus will need the corresponding collision avoidance system to work. These buoys are geometrically simpler and thus have a reduced computational impact compared to other static obstacles. Moreover, the buoys can represent any other obstacle, such as icebergs and debris.

The *Contact* vehicle is a copy of the *Ownship* vehicle, having been coloured red for distinction. In some simulations, it will have the same collision avoidance behaviour as the *Ownship*, while on others it

[1]gazebosim.org

will act without the correct behaviour in order to further verify the capacity of the developed algorithms. The dimensions and velocity of the *Contact* vehicle were, in some simulations, also changed.

The detection of these obstacles is realized with the exteroceptive sensors, such as the AIS and the Lidar, and the collision avoidance method is thus defined with perimeters of safety, established around the *Ownship* vehicle.

4.4 Perimeters of safety

Having different types of obstacles led to the implementation of three perimeters of safety, as is depicted in the figure 4.3, having a fourth perimeter for Head On collision mode, as will be further explained.

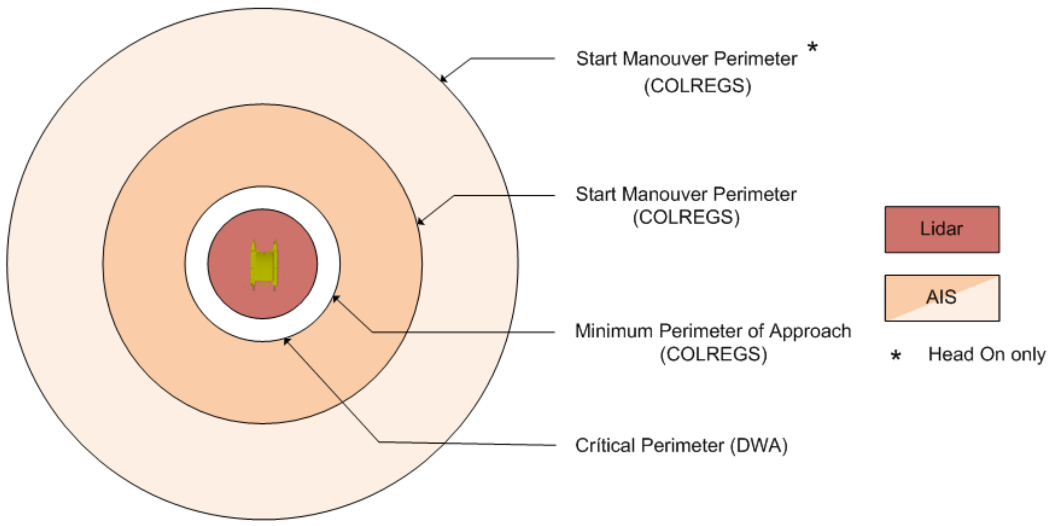


Figure 4.3: Perimeters of safety.

The inside perimeter, which was named Critical Perimeter, has a fixed diameter of 15 meters. Whatever obstacle that enters this perimeter and is detected by the Lidar, or any other detecting and ranging equipment available, is treated as an imminent risk of collision for the vehicle, thus leading to the reactive collision avoidance DWA, which will avoid the obstacle disregarding the COLREGS rules. This critical perimeter is defined mainly for static obstacles, which have no need of maritime rules compliance and also for cases of AIS malfunctioning or non-equipped vessels, where the critical situation demands immediate effects.

In the same manner there are behaviour rules for vehicles on land, rules for the sea also exist in multiple vehicle situation. In order to apply COLREGS based behaviour on the vehicle, the other two perimeters are required. An outer perimeter and a middle perimeter were defined, respectively named start maneuver and minimum distance admissible, and are used for dynamic obstacles (other vehicles) situations exclusively. These perimeters have non-static diameters which are dependent on the vehicle's dimensions. As the name implies, the start maneuver perimeter is the perimeter in which, when a contact vehicle is detected and is deemed in risk of collision, the collision avoidance maneuver starts. The minimum distance admissible defines the limit as to which the *Contact* vehicle can approach the *Ownship* vehicle. These radius are the limits by which the algorithm implemented will behave. These radius can be obtained following the equations 3.42 and 3.43. A special case is when the vehicles are in a head on situation, in which they approach each other faster than the other modes. Consequently, the perimeter at which the vehicle will start the evasion will be double the normal start maneuver perimeter. For cases that the critical perimeter is breached with a COLREGS mode already defined, the perimeter will disregard the breaching up to 5 meters inside. The exchange of data is secured by the AIS system, which for the simulation, the exchange of data between the vehicles, is simulated with ROS messages.

4.5 AIS message

An AIS system, when in movement, shares it's own vehicle data in order to facilitate collision avoidance to the other vehicles. For simulation, the main message data with relevance for the algorithm were simulated as ROS messages as in table 4.1.

Table 4.1: ROS message simulating AIS messages.

AIS.msg

Data Type	**Variable Name**	**Unit**
uint32	mmsi	-
uint8	status	-
float64	SOG	[m/s]
float64	longitude	[deg]
float64	latitude	[deg]
float64	COG	[deg]
float32	length	[m]

The *mmsi* stands for Maritime Mobile Service Identity and it is a series of nine digits to uniquely identify the ship. The algorithm developed needs to know its own *mmsi* number in order to filter and remove its own AIS publications.

The status is used as the current mode of collision avoidance. The modes can be: [0] Null, [1] CPA, [2] GiveWayOT, [3] Head On, [4] StanOnOT, [5] GiveWayX and [6] StandOnX. The Null mode is the free mode, where no collision risk was detected so far, and the vehicle is following the mission plan. The CPA is the critical situation where no mode is recognized or the vehicle is inside the minimum perimeter of approach, thus leading to disregard of rules and immediate retreat to the most promising path. The other modes are the possible situations the vehicle can encounter, as explained in 3.9 and the algorithms presented in appendix B.

The parameters which allow the algorithm to detect the mode are: the SOG, which stands for speed over ground (referenced as only speed); the COG, which stands for course over ground (mainly referenced as heading); and the local coordinates, which are obtained with the latitude and longitude variables, and are thus transformed. The length variable is also vital for the algorithm, in order to define the non-static perimeters of safety.

4.6 System Architecture

The Obstacle Detection and Avoidance module, as presented in section 2.1, is comprised of three parts. Before proceeding to the results, a description of the collision avoidance process helps to provide a better understanding of how the system globally works.

With the waypoints defined in a global map, the vehicle navigates through the route between the waypoints. In this part, the proprioceptive sensors localize the vehicle and represent the vehicle in the map. The exteroceptive sensors on the other hand are continuously searching for both static and dynamic objects. Then three[2] perimeters layers (section 4.4) are defined. The AIS is then responsible for the detection and exchange of the data between the *ownship* and the *contact*, which, when in range, proceed as follows:

1. Estimation/Update of *ownship* and *contact* states.
2. Assessment of whether a risk of collision exists.
   - 2.1. In case no risk of collision is detected: no alteration in the course is made, however proceed with surveillance on the *contact*.
   - 2.2. In case a risk of collision is detected: (a) With AIS data exchanged (position and heading) determine the applicable COLREGS rule. (b) Apply the constraints associated with the corresponding COLREGS rule to the motion planning algorithm.

An inner perimeter is also defined (critical area), in which collision avoidance is applied disregarding COLREGS rules. The Lidar is the sole detection mechanism for this layer. This inner perimeter is for static obstacles which need no COLREGS rules and for vehicles in cases that the AIS is malfunctioning or no AIS system is presented in the contact vessel and an immediate response is needed.

4.6.1 Estimation/Update of states

With the combination of both GNSS and IMU data, which are treated as ROS messages, the odometry is produced. The message now handles both pose and twist type messages. With twist type, the velocity in all axis is obtained and converted to linear velocity, producing SOG. With pose data, the quaternions are converted to euler angles, and thus the yaw obtained is converted to the heading, also known as COG. With GNSS data, the latitude, longitude and altitude are treated, and the local coordinates are then obtained. Therefore, all the states are available to its own vehicle. In order to get the other vehicle's parameters, the AIS is continuously subscribed, and in case the *mmsi* number in the message is different than the *ownship's mmsi*, the data is saved as the opposing vehicle states.

4.6.2 Risk and heuristics

The next step is the assessment of whether the vehicle is in risk of collision or not, applying the equation 3.37, and using the perimeters defined in section 4.4. Being in risk, the algorithm will now compute all the equations from chapter 3. These values will enter the mode algorithms (appendix B), thus the mode

[2]The third perimeter is doubled for the Head On mode.

is determined. A loop is now entered, ranging from the space of conceivable velocities, and ranging in all the heading values, $[0; 360[$. The functions computed inside the loop are the necessary to calculate the CPA value (equation 3.37), and then follow the equations from section 3.10. It is also given preference to the group of velocities and headings, the most similar to the previous, in cases where the function output is equal. In the end of the function, a desired velocity and a desired heading is ready to be sent to the control module. However, the input available is velocity and yaw, thus a conversion from heading to yaw is made before sending.

4.6.3 AIS publishing

After sending the desired input to the control module, the main loop function is not yet complete. It is crucial to also broadcast the actualized states of the vehicle as an AIS message, in order to permit the others to know our location and take measures to initialize theirs own collision avoidance procedures.

4.6.4 Inside critical perimeter

As explained in both sections 4.4 and 4.6, in case the Lidar system detects obstacles within 15 meters of the vehicle, and no COLREGS mode is already defined, the Dynamic Window Approach algorithm, described in subsection 2.1.2, is employed as the primary function. The vehicle already knows its own parameters from the estimation/update part of the algorithm. The number of rays emitted from the 2D Lidar used are also knowable, either by defining manually or by analyzing the size of the data array obtained from the measures. With the heading from the vehicle, and the position and range of the ray, the obstacle is then represented on the system as an array of local coordinates. Having the goal defined, and with the obstacles array, the system loops now in the DWA functions, which take into account the search space of the possible velocities as defined in subsubsection 2.1.2, the new desired heading, and the distance between the vehicle and the obstacles in the trajectory. All these components are used in an objective function, being the desired output to maximize the loop instance. In case there is a mode already defined, the DWA switch will only be made when the Lidar detects the obstacle within 5 meters.

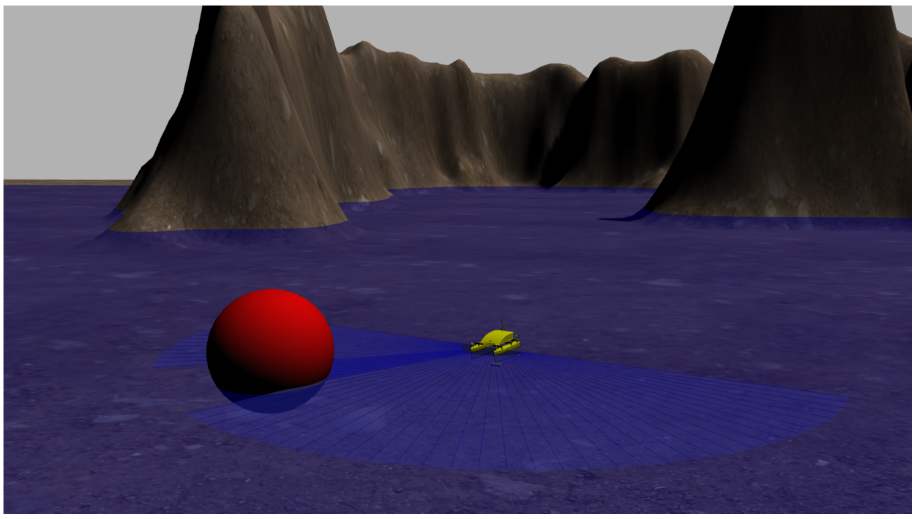


Figure 4.4: ASV with critical perimeter in blue, defined with the Lidar rays, and detecting a static obstacle.

Chapter 5

Simulation and Results

In order to verify the capability of the algorithm developed, different scenarios were used. Scenarios with other vehicles as obstacles ought to follow COLREGS rules, such as:

- Head On
- Overtaking
- Crossing

These simulations have the *Ownship* always following COLREGS rules. The *Contact* on the other hand, has in some cases to be COLREGS compliant, while on others it can be non-compliant, in order to further verify the algorithm.

It is also considered two other scenarios:

- Multi-static obstacle
- Big static obstacle

In the different scenarios, variations in the initial velocity and dimension of obstacles are applied, in order to further test and to meet conditions.

5.1 Head On

The Head On simulations comprises two vehicles, the *Ownship* and the *Contact*, which have the same length (roughly 3 meters) and the same cruising velocity (1.5 meters per second). What will differ in the following simulations is whether the *Contact* will also follow COLREGS behaviour or not, maintaining the same heading and velocity.

The Head On mode (algorithm 3 in appendix B), has the absolute release check different than the other modes, which is the double of r_{cpa_max}, also defined in the algorithms as r_{pwt}. Thus, is due to the vehicles being in the same orientation but the opposing direction, approaching faster than the other modes. The entry mode criteria is that α and β are less or equal than ϕ value, which is the relative bearing of each vehicle to the other. The value used was $\phi = 18^o$.

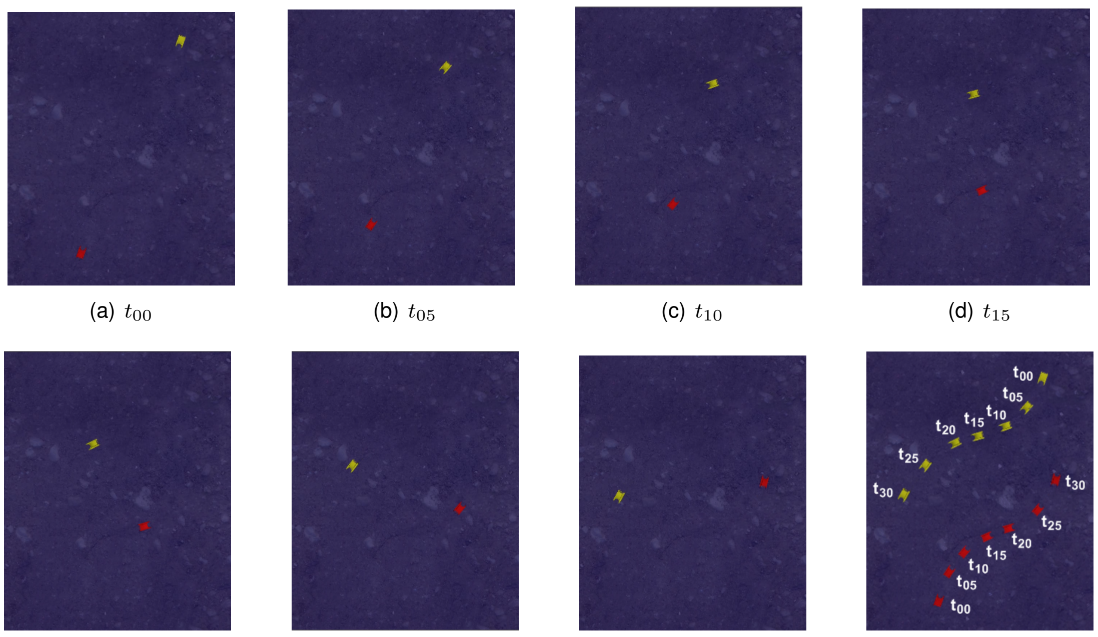
5.1.1 *Contact* with COLREGS behaviour

Having the *contact* also using the same algorithm, the results are presented in table 5.1, where the values of the local coordinates, course over ground (COG), speed over ground (SOG), and the respective collision avoidance mode are presented with a time step of 5 seconds.

Table 5.1: Results for HeadOn Simulation with a 3 meters COLREGS compliant *Contact*.

	Ownship					*Contact*				
	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode
t_{00}	-10.85	-27.00	200.21	1.50	0	-35.86	-81.10	19.97	1.50	0
t_{05}	-13.48	-33.60	214.50	1.58	3	-32.45	-72.62	23.02	1.15	3
t_{10}	-17.90	-38.90	238.90	0.78	3	-30.56	-68.34	37.57	0.96	3
t_{15}	-22.72	-40.78	254.69	1.02	3	-27.61	-64.46	52.76	0.93	3
t_{20}	-27.83	-42.13	255.80	1.02	3	- 22.69	-61.03	70.77	0.97	3
t_{25}	-33.21	-44.55	229.36	1.62	0	- 19.05	-59.70	60.41	1.25	0
t_{30}	-39.67	-52.63	210.21	1.50	0	- 11.56	-54.03	30.71	1.32	0

To better visualize the results from table 5.1, images were taken from the Gazebo simulation (figure 5.1), which allows a better vision of the trajectory taken by both vehicles. The images are also presented with a time step of 5 seconds, having the figure 5.1(h) the complete trajectory.



(a) t_{00} (b) t_{05} (c) t_{10} (d) t_{15}

(e) t_{20} (f) t_{25} (g) t_{30} (h) Complete path

Figure 5.1: *Ownship* HeadOn simulation with a 3 meters COLREGS compliant *Contact*.

Analyzing the results, it is observable that both vehicles started in *Null* mode, and with a velocity of 1.5 meters per second. At the same time step, both vehicles had already switched from *Null* mode to the Head On mode (t_{05}), and started the maneuver, reducing the speed while changing the heading. Applying the equations 3.37 to 3.41 for the time step t_{05}, we get the following results: $k_0 = 1882.42$, $k_1 = -235.23$, $k_2 = 7.38$, $t_{CPA} = 15.94\ s$ and $CPA = 2.83\ m$. Applying the equation 3.42, $r_{cpa_{min}} = 2 \times contact_{length} = 2 \times 3 = 6\ m$. Thus, being $CPA < r_{cpa_{min}}$, we are indeed in risk of collision. The entry condition for Head On mode was that both α and β needed to be below ϕ which was defined as 18^o. Applying equations 3.5 and 3.6, we obtain $\alpha = 2.91^o$ and $\beta = 8.57^o$ which are both below the

ϕ value. When the algorithm deemed that the Head On mode was complete (t_{25}), the mode changed again to *Null*, and the speed stabilized again to the cruise speed of 1.5 meters per second. The release check for the Head On mode is that $r > \breve{r}_{cpa}$ and $\dot{r} > 0$, which applying the equation 3.42, $r_{cpa_{min}} = 2 \times contact_{length} = 2 \times 3 = 6\ m$, and applying the equations 3.7, 3.8a and 3.8b, $v_{os}^{cn} = -0.07\ m/s$, $v_{cn}^{os} = -0.29\ m/s$ and $\dot{r} = 0.36\ m/s$. The distance between the vehicles is obtained from equation 3.4, $r = \sqrt{(-33.21 - (-19.05))^2 + (-44.55 - (-59.70))^2} = 20.74\ m$, thus the absolute release check is verified with both conditions.

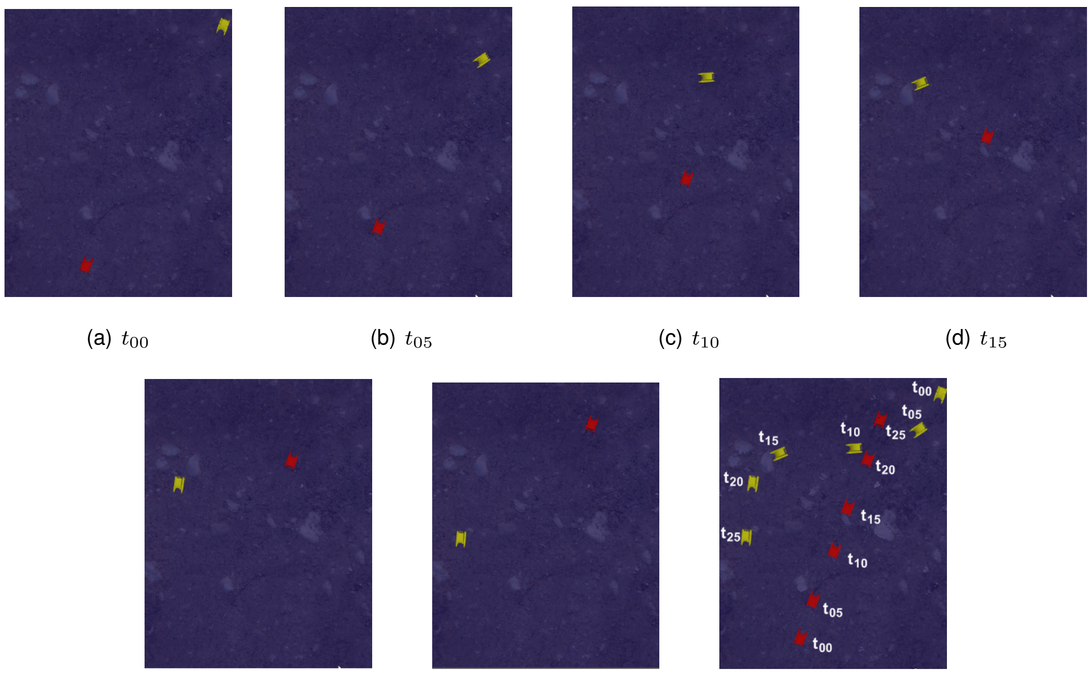
5.1.2 *Contact* with non-COLREGS behaviour

The second Head On simulation comprised two vehicles, the *ownship* and *contact*, having the same length (roughly 3 meters) and the same cruising velocity (1.5 meters per second). However, contrary to the previous simulation, the *contact* was non-COLREGS compliant. The results are shown in table 5.2.

Table 5.2: Results for HeadOn Simulation with a 3 meters non-COLREGS compliant *Contact*.

	Ownship					*Contact*				
	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode
t_{00}	-11.24	-33.75	199.97	1.50	0	-33.16	-73.89	19.99	1.50	0
t_{05}	-14.38	-39.10	236.14	1.53	3	-30.89	-68.13	20.01	1.50	0
t_{10}	-22.27	-41.78	264.78	2.49	3	-28.90	-62.57	19.99	1.50	0
t_{15}	-33.11	-42.26	258.23	1.77	3	-26.37	-55.34	20.00	1.50	0
t_{20}	-39.80	-46.10	193.53	1.78	0	-23.36	-46.83	19.99	1.50	0
t_{25}	-40.82	-54.52	184.78	1.53	0	-21.87	-42.73	20.00	1.50	0

To better visualize the results from table 5.2, images were taken from the Gazebo simulation (figure 5.2), which allows a better vision of the trajectory taken by both vehicles. The images are also presented with a time step of 5 seconds, having the figure 5.2(g) the complete trajectory.



(a) t_{00} (b) t_{05} (c) t_{10} (d) t_{15}

(e) t_{20} (f) t_{25} (g) Complete path

Figure 5.2: *Ownship* HeadOn simulation with a 3 meters non-COLREGS compliant *Contact*.

Analyzing the results, it is observable that both vehicles started in *Null* mode, and with a velocity of 1.5 meters per second. At the time step t_{05}, the *ownship* switched from mode *Null* to the Head On mode, starting the maneuver. Contrary to the previous simulation, the *contact* vehicle maintained the heading and speed. Due to this, at time step t_{10} it can be observed an increase in the speed in order to make up for the stand on *contact*. At time step t_{20}, the *ownship* had already passed the *contact*, and, thus, it switches to *Null* again, having the velocity converge to the cruising speed of 1.5 meters per second.

5.2 Overtaken

Similarly with the Head On simulations, the overtaken simulations also comprise two vehicles, the *ownship* and *contact*. In the overtaken mode exists two submodes: port and starboard. The submode depends on the $pass^{os}_{cns}$ boolean condition (as can be seen in algorithm 2 in appendix B), being starboard in case the condition is true, and port otherwise.

The first two cases presented are port and starboard submode with the *contact* vehicle having a length of roughly 3 meters, whereas on the last, it is presented a starboard submode with the *contact* vehicle with 7.5 meters. The *ownship* vehicle has a cruising velocity of 1.5 meters per second, while the *contact*, in every simulation, has a velocity of 0.5 meters per second, in order to allow being overtaken.

5.2.1 Port Submode

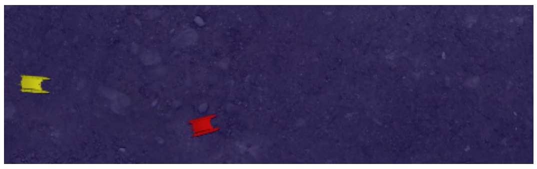
Having the COLREGS algorithm running in both vehicles, the results are presented in table 5.3.

Table 5.3: Results for Overtaking Simulation. *Ownship* overtaking in port submode and *Contact* being overtaken.

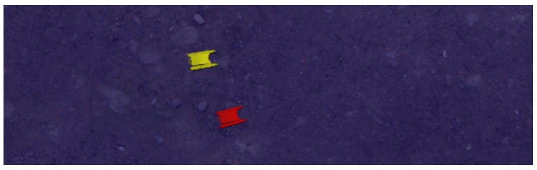
	Ownship					*Contact*				
	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode
t_{00}	-18.77	-38.96	100.85	1.58	0	1.11	-35.75	74.99	0.50	0
t_{05}	-5.84	-38.47	79.82	2.92	2(1)	2.83	-34.71	75.00	0.50	4
t_{10}	12.10	-27.28	87.16	2.49	0	3.83	-34.06	75.00	0.50	0
t_{15}	16.60	-27.64	94.18	2.04	0	5.19	-33.35	75.11	0.50	0
t_{20}	24.83	-28.16	96.77	1.68	0	7.58	-32.95	75.00	0.50	0

To better visualize the results from table 5.3, images were taken from the Gazebo simulation, figure 5.3, which allows a better vision of the trajectory taken by both vehicles. The images are also presented with a time step of 5 seconds, having the figure 5.3(f) with the complete trajectory.

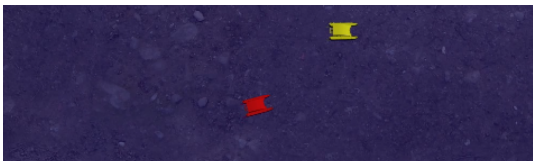
Analyzing the results, we can observe that both vehicle started in *Null* mode. Having the *contact* enter the outer perimeter of the *ownship* at roughly t_{05}, the mode switched to GiveWayOT with the submode being port. It can be verified that, when applying the equations 3.37 to 3.41 for the time step t_{05}, we have the following results: $k_0 = 89.31$, $k_1 = -44.37$, $k_2 = 5.87$, $t_{CPA} = 3.78\,s$ and $CPA = 2.33\,m$. Applying the equation 3.42, $r_{cpa_{min}} = 2 \times contact_{length} = 2 \times 3 = 6\,m$. Thus, we are indeed in risk of collision, being $CPA < r_{cpa_{min}}$. The entry condition for the Overtaking mode is that α cannot be greater than 247.5^o or less than 112.5^o, which is true, being $\alpha = 171.56^o$, and the boolean condition $pass^{os}_{cn}$ must be true, which is verified by applying the equation 3.13a, and the equations from section 3.6: $fore^{os}_{cn} = True$, $\theta^{os}_{\epsilon} = 75^o$, $v^{os}_{cnh} = 2.91\,m/s$, $v^{os}_{\epsilon} = 3.41\,m/s$, and $pass^{os}_{cn} = True$. Having checked the GiveWayOT mode entry, it now needs to define the submode, which is defined as Starboard in case



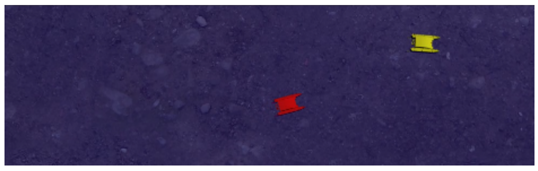
(a) t_{00}



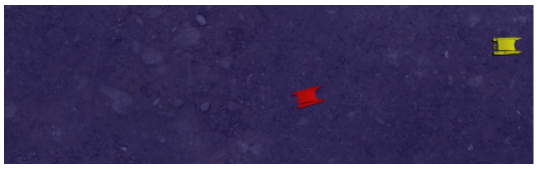
(b) t_{05}



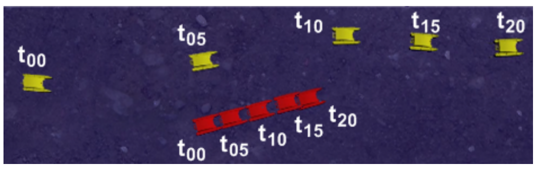
(c) t_{10}



(d) t_{15}



(e) t_{20}



(f) Complete path

Figure 5.3: *Ownship* GiveWayOT simulation in port submode, with a StandOnOT COLREGS compliant *Contact*.

$pass^{os}_{cns}$ is $True$ and Port submode otherwise. In this case, with equation 3.13b having $aft^{os}_{cn} = False$ and applying equation 3.32, the output is $pass^{os}_{cns} = False$ which leads to Port submode. The loop through the space of possible headings and velocities, in order to reduce the risk function value (equation 3.45), led to a heading variation and an increase in speed in order to help the overtaking situation. On the other hand, the *contact* verifies the StandOnOT situation and maintains both heading and velocity. At time step t_{10}, both vehicle switched to mode *Null* as the overtaking was deemed complete, due to the fact that the *ownship* was now ahead of *Contact*. Thus leads to verify the condition of $\dot{r} > 0$, which releases the vehicle from the mode condition. The $\dot{r}$ condition is obtained from equations 3.7, 3.8a and 3.8b: $v^{cn}_{os} = -2.00\, m/s$, $v^{os}_{cn} = 0.46\, m/s$ and $\dot{r} = 1.55\, m/s$. From time step t_{10} to t_{20}, the *ownship's* surge speed decreased, converging to the cruising velocity.

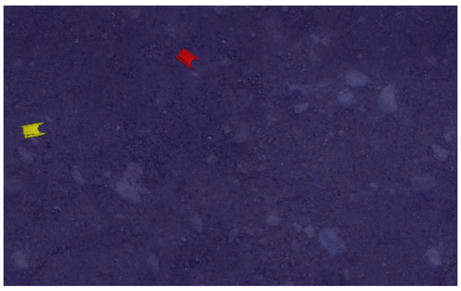
5.2.2 Starboard submode

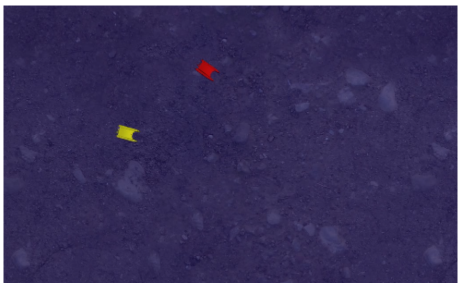
Having the COLREGS algorithm running in both vehicles, the results are presented in table 5.4.

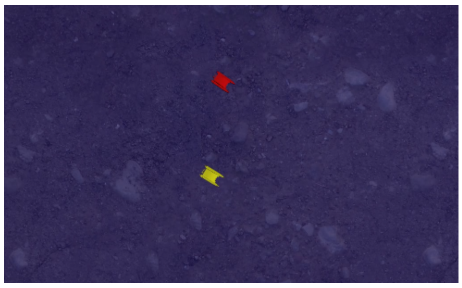
Table 5.4: Results for Overtaking Simulation. *Ownship* overtaking in starboard submode and *Contact* being overtaken.

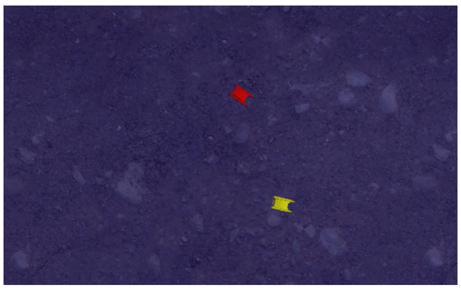
	Ownship					*Contact*				
	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode
t_{00}	-31.15	-48.06	79.98	1.53	0	-12.89	-38.64	125.00	0.50	0
t_{05}	-22.80	-47.83	101.74	2.42	2(2)	-11.65	-39.87	125.00	0.50	4
t_{10}	-12.79	-58.94	129.58	2.47	2(2)	-9.57	-48.95	124.99	0.50	4
t_{15}	-2.36	-56.40	103.52	1.89	0	-6.78	-43.02	125.00	0.50	0
t_{20}	4.48	-57.74	94.37	1.69	0	-5.18	-44.09	125.00	0.50	0
t_{25}	13.68	-57.79	86.49	1.51	0	-2.46	-45.78	125.00	0.50	0

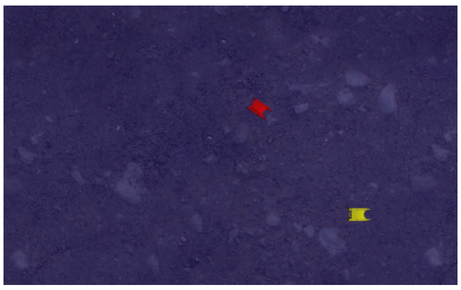
To better visualize the results from table 5.4, images were taken from the Gazebo simulation, figure 5.4, which allows a better vision of the trajectory taken by both vehicles. The images are also presented with a time step of 5 seconds, having the figure 5.4(g) with the complete trajectory.

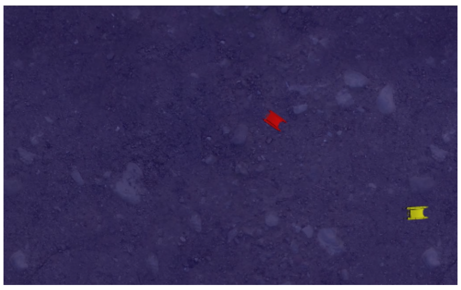

(a) t_{00}

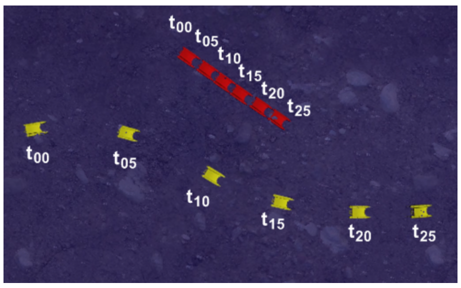

(b) t_{05}


(c) t_{10}


(d) t_{15}


(e) t_{20}


(f) t_{25}



(g) Complete path

Figure 5.4: *Ownship* GiveWayOT in starboard submode simulation with a StandOnOT COLREGS compliant *Contact*.

Analyzing the results, we can observe that both vehicle started in *Null* mode. Having the *contact* enter the outer perimeter of the *ownship* at roughly t_{05}, the mode switched to GiveWayOT, with the submode being starboard. As opposing to the previous simulation, following the same procedure, the

$pass^{os}_{cns}$ boolean condition is now set to $True$. An increase in speed is verified in order to help the overtaking situation and the corresponding heading variation as well. On the other hand, the *contact* verifies the StandOnOT situation and maintains both heading and velocity. At time step t_{15}, both vehicles switched to mode *Null* as the overtaking was deemed complete, due to the fact that the *ownship* was now ahead of *contact* ($\dot{r} > 0$). From time step t_{15} to t_{25}, the *ownship's* surge speed decreased, converging to the cruising velocity.

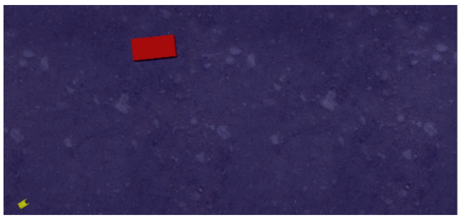
5.2.3 Starboard submode with 7.5 meters long *contact*

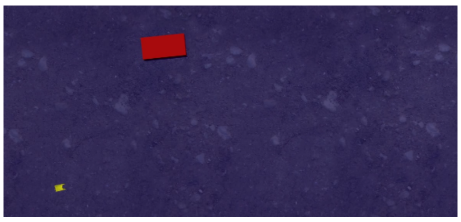
Contrary to the previous simulation, the *contact* vehicle's length was changed to 7.5 meters. The results are presented in table 5.5.

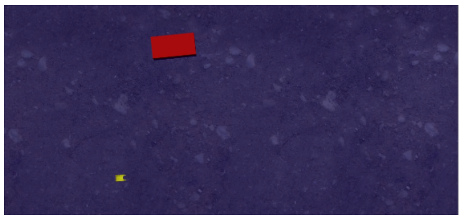
Table 5.5: Results for Overtaking Simulation with a 7.5 meters long *contact*. *Ownship* overtaking in starboard submode and *Contact* being overtaken.

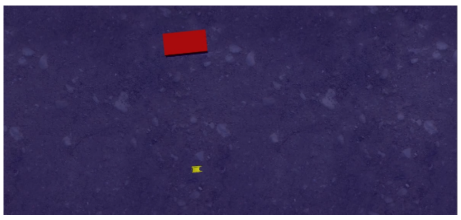
	Ownship					*Contact*				
	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode
t_{00}	-45.94	-79.26	59.80	1.44	0	-13.38	-41.23	84.98	0.58	0
t_{05}	-38.82	-75.24	70.17	2.27	2(2)	-10.24	-40.99	85.00	0.58	4
t_{10}	-25.77	-72.00	86.86	3.20	2(2)	-7.84	-40.71	85.00	0.58	4
t_{15}	-8.17	-78.69	85.94	3.34	2(2)	-5.86	-40.65	84.99	0.58	4
t_{20}	7.02	-69.53	81.15	2.44	0	-3.04	-40.54	84.99	0.50	0
t_{25}	15.26	-66.82	58.48	2.38	0	-0.35	-40.23	85.00	0.50	0
t_{30}	29.08	-58.51	59.46	1.95	0	3.13	-39.83	85.00	0.50	0
t_{35}	36.00	-54.12	59.86	1.69	0	5.62	-39.68	85.81	0.50	0
t_{40}	43.16	-58.32	59.94	1.59	0	7.78	-39.39	85.00	0.50	0
t_{45}	49.30	-46.77	59.96	1.50	0	10.06	-39.21	85.00	0.50	0
t_{50}	55.41	-43.25	60.01	1.50	0	12.09	-38.98	85.81	0.50	0

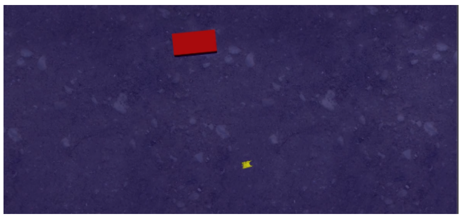
To better visualize the results from table 5.5, images were taken from the Gazebo simulation, figure 5.5, which allows a better vision of the trajectory taken by both vehicles. The images are also presented with a time step of 5 seconds, having the figure 5.5(l) with the complete trajectory. This time, the *contact* vehicle is represented as a huge rectangular brick, in order to reduce computational effort.


(a) t_{00}

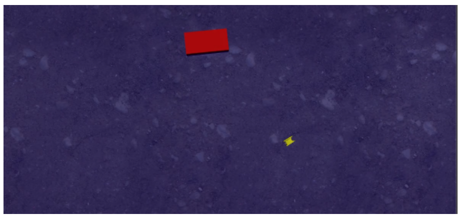

(b) t_{05}


(c) t_{10}

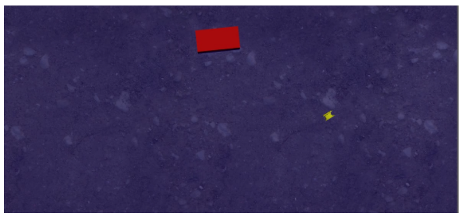

(d) t_{15}



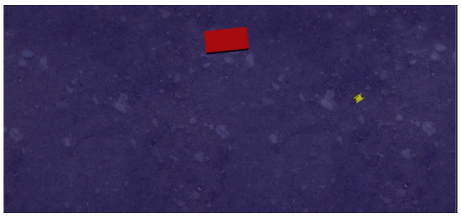
(e) t_{20}



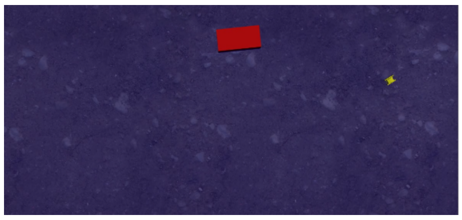
(f) t_{25}



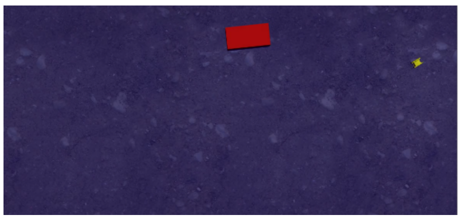
(g) t_{30}



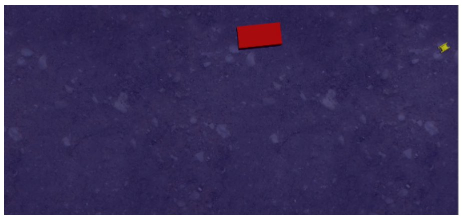
(h) t_{35}



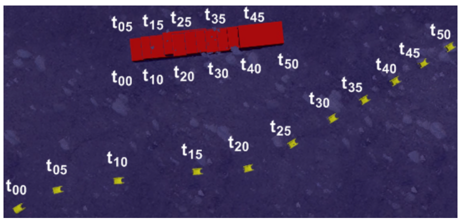
(i) t_{40}



(j) t_{45}



(k) t_{50}


t_{05} t_{15} t_{25} t_{35} t_{45}
t_{50}
t_{45}
t_{00} t_{10} t_{20} t_{30} t_{40} t_{50}
t_{40}
t_{35}
t_{30}
t_{25}
t_{20}
t_{15}
t_{10}
t_{05}
t_{00}

(l) Complete path

Figure 5.5: *Ownship* GiveWayOT in starboard submode simulation with a StandOnOT COLREGS compliant *Contact* with 7.5 meters length.

Analyzing the results, we can observe that both vehicle started in *Null* mode. Having the *contact* enter the outer perimeter of the *ownship* at roughly t_{05}, the mode switched to GiveWayOT with the submode being starboard. Comparing the results from tables 5.4 and 5.5, we can verify the increase of the radius in the perimeters of safety, as expected, due to the increase in the *contact's* length. Having a length of $3\ m$ in the previous simulation, the start of maneuver was set to start at $3 \times 6.5 = 19.5\ m$ apart from each other. On the other hand, a *contact's* length of $7\ m$, led to a start of maneuver of $7 \times 6.5 = 45.5m$, which in time step t_{05}, is verified that the distance between vehicles is $r = 44.61m$. After entering the overtaking mode, an increase in speed occurred, in order to help the overtaking situation, and the corresponding heading variation occurred as well. On the other hand, the *contact* verifies the StandOnOT situation and maintains both heading and velocity. At time step t_{20}, both vehicle switched to mode *Null* as the overtaking was deemed complete. The *ownship* regained the course heading and speed, which converged to its initial conditions.

5.3 Crossing

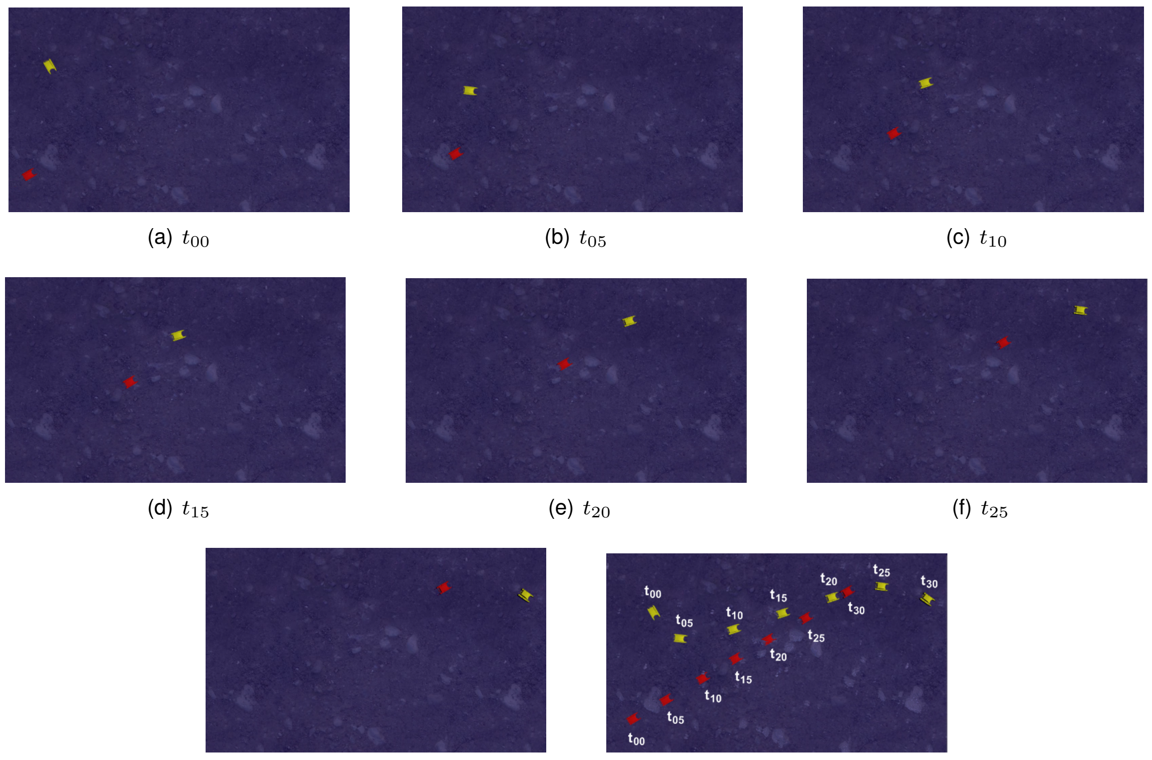
The Crossing simulations comprises two vehicles, the *ownship* and *contact*, which have the same length (roughly 3 meters) and the same cruising velocity (1.5 meters per second).

Having the *contact* also using the same algorithm, the results are presented in table 5.6, where the values of the local coordinates, course over ground (COG), speed over ground (SOG) and the respective collision avoidance mode are presented with a time step of 5 seconds.

Table 5.6: Results for Crossing Simulation. *Ownship* crossing and *Contact* on StandOn.

	Ownship					*Contact*				
	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	mode
t_{00}	-20.02	-35.50	149.77	1.51	0	-25.96	-59.04	59.90	1.50	0
t_{05}	-14.24	-41.93	97.94	1.72	5(3)	-19.53	-55.37	59.96	1.50	6
t_{10}	-6.49	-41.15	73.33	2.00	5(3)	-14.33	-52.14	59.95	1.50	6
t_{15}	1.36	-38.71	72.37	2.03	5(3)	-9.30	-49.17	59.96	1.50	6
t_{20}	13.57	-34.82	72.29	2.04	5(3)	-0.14	-43.82	59.95	1.50	6
t_{25}	20.56	-32.71	85.74	1.60	0	5.00	-40.74	59.95	1.50	6
t_{30}	31.48	-35.34	125.00	1.53	0	13.88	-35.47	60.00	1.50	0

To better visualize the results from table 5.6, images were taken from the Gazebo simulation (figure 5.6), which allows a better vision of the trajectory taken by both vehicles. The images are also presented with a time step of 5 seconds, having figure 5.6(h) the complete trajectory.



(a) t_{00} (b) t_{05} (c) t_{10}

(d) t_{15} (e) t_{20} (f) t_{25}

(g) t_{30} (h) Complete path

Figure 5.6: *Ownship* GiveWayX simulation with a StandOnX COLREGS compliant *Contact*.

Analyzing the results, it is observable that both vehicle started in *Null* mode, and with a velocity of roughly 1.5 meters per second. At the time step t_{05}, the *Ownship* switched to crossing mode GiveWayX, with the Bow submode, while the *Contact* changed to StandOnX mode. The entry for GiveWayX mode requires that the conditions $\alpha < 247.5^o$ and $\beta > 112.5^o$ are not met, which is verified by having an $\alpha = 38.48^o$, and $\beta = 103.55^o$. For determining the submode, in case $cross^{os}_{xcnb}$ is set $True$ and $r^{os}_{xcnb} > (\hat{r}_{cpa} + \breve{r}_{cpa})/2$, the Bow submode is defined, being otherwise Stern submode. In order to verify the entry condition, some pre-calculations ought to be made: from equations 3.9, 3.10a and 3.10b, $v^{os}_{tn} = -1.67m/s$, $v^{cn}_{tn} = -0.93m/s$ and $\dot{\beta} = 10.34^o/s$; from equations equations 3.14a and 3.14b, $port^{os}_{cn} = True$ and $star^{os}_{cn} = False$; applying the equations from section 3.5, $\theta^{os}_{\gamma} = 149.96^o$, $v^{os}_{\gamma} = 1.06\,m/s$, $r_{\gamma} = 8.99\,m$, $cross^{os}_{xcn} = True$ and $cross^{os}_{xcnb} = True$; and the equations from the numerical crossing subsection (subsection 3.5.2), $t^{os}_{\gamma} = 8.49\ s$, $r^{xos}_{\epsilon} = 11.51\ m$, $r^{xcn}_{\epsilon} = 12.74\ m$ and $r^{os}_{xcnb} = 12.95\ m$. These conditions make the entry to Bow submode. In order to avoid collision, the *ownship* redefined the heading, and increased its velocity. At time step t_{25}, the *ownship* deemed there was no risk of collision and switched to mode *Null*. At last in time step t_{30}, both vehicles were in *Null* mode, going back to the initial conditions.

5.4 Static Obstacles

The static obstacles simulations comprise the *ownship* travelling with a velocity of roughly 1.75 meters per second, encountering buoys with diameters of 5 meters. These buoys represent any type of static obstacle, as explained in section 4.3, in order to reduce the computational necessity in the simulation. These buoys will be displayed in two different dispositions: scattered around the map or closely together, in order to represent an even bigger obstacle. The algorithm employed will be the Dynamic Window Approach, where the critical perimeter set for the Lidar response is 15 meters, which, when violated, sets the boolean danger variable to be True. In the images taken from the Gazebo simulation, the Lidar rays will appear as a more intense blue, which represent the range of 15 meters, and when in contact with an obstacle, an even darker blue appears around.

5.4.1 Scattered Obstacles

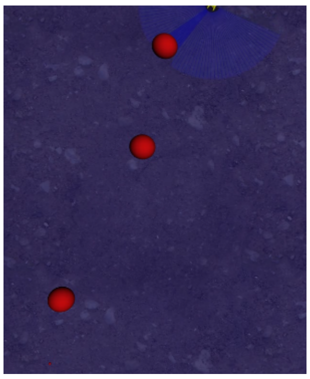
Having the obstacles scattered, the results are presented in table 5.7, where the values of the local coordinates, course over ground (COG), speed over ground (SOG), and boolean information of danger are presented with a time step of 5 seconds.

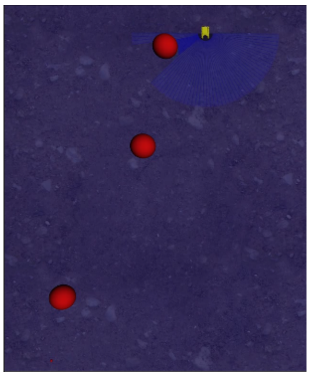
Table 5.7: Results for multi-static collision avoidance.

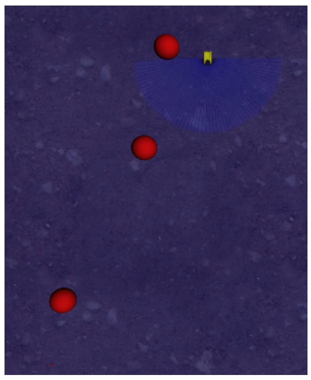
	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	Danger		x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	Danger
t_{00}	-15.13	-31.27	115.90	1.71	True	t_{30}	-21.12	-73.13	116.21	1.99	False
t_{05}	-17.00	-37.59	90.16	1.78	True	t_{35}	-27.11	-80.51	128.21	1.59	False
t_{10}	-16.99	-41.20	87.74	0.91	False	t_{40}	-34.10	-88.79	135.13	1.78	True
t_{15}	-18.33	-48.57	112.30	1.92	False	t_{45}	-38.36	-92.99	137.13	1.72	True
t_{20}	-21.23	-55.57	105.33	1.75	True	t_{50}	-45.00	-100.00	139.83	1.70	False
t_{25}	-22.10	-65.02	86.58	1.85	False						

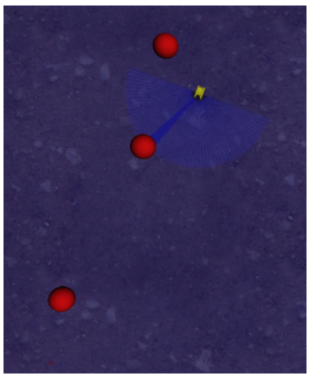
To better visualize the results from table 5.7, images were taken from the Gazebo simulation (figure 5.7), which allows a better vision of the trajectory taken by the vehicle. The images are also presented with a time step of 5 seconds, having figure 5.7(l) the complete trajectory.

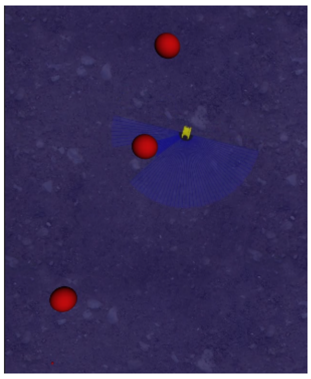
Analyzing the results, it is observable that, since time step t_{00} to t_{05}, there was already an obstacle inside the critical perimeter, having the danger set to True. At time step t_{10}, danger exited to False and the vehicle tried to regain the mission course, reducing it's velocity and adjusting the heading. However, at time step t_{20}, another obstacle entered the perimeter, thus reducing slightly the velocity, compared to the previous time step. From time step t_{25} to t_{35}, no other obstacle was detected, adjusting now the heading to the goal of the mission. At time step t_{40}, the last obstacle entered the perimeter. This time, even though there was danger, the vehicle had slightly no change in its course as avoidance was not needed. The goal was then reached, being the local coordinates $(-45.00; -100.00)$.

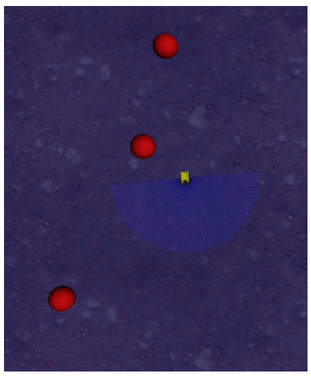

(a) t_{00}

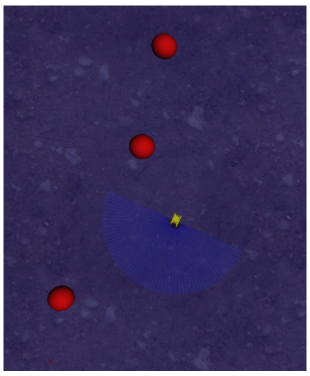

(b) t_{05}

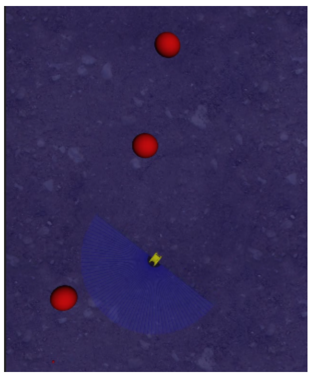

(c) t_{10}

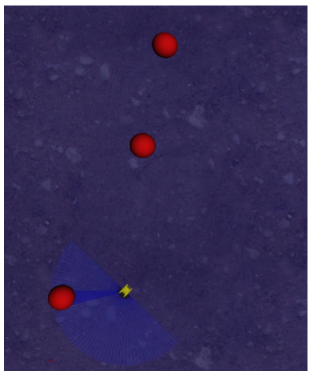

(d) t_{15}

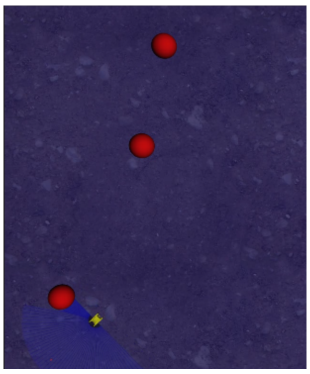

(e) t_{20}

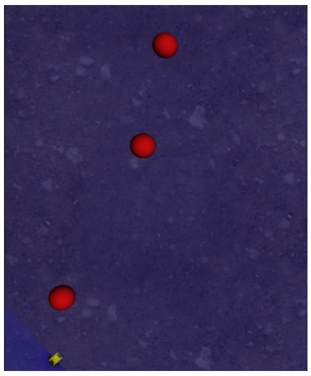

(f) t_{25}


(g) t_{30}


(h) t_{35}


(i) t_{40}


(j) t_{45}


(k) t_{50}

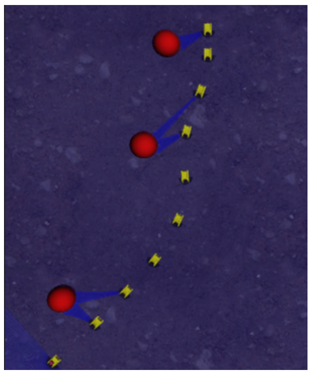

(l) Complete path

Figure 5.7: Multi-static collision avoidance.

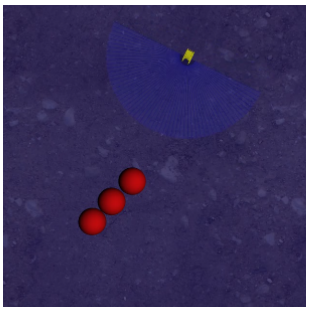
5.4.2 Big Obstacle

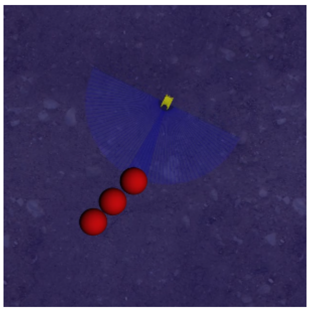
Having a big obstacle in the middle of the course, the results are presented in table 5.8, where the values of the local coordinates, course over ground (COG), speed over ground (SOG), and boolean information of danger are presented with a time step of 5 seconds.

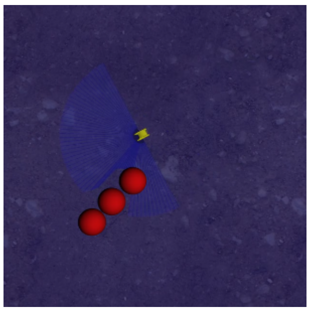
Table 5.8: Results for big static obstacle collision avoidance.

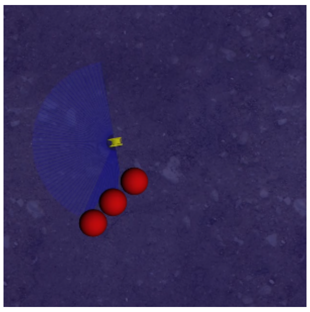
	x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	Danger		x (EAST) [m]	y (NORTH) [m]	COG [deg]	SOG [m/s]	Danger
t_{00}	-16.00	-33.43	116.29	1.75	False	t_{20}	-33.14	-49.70	167.87	1.14	True
t_{05}	-19.65	-40.82	116.15	1.75	True	t_{25}	-39.59	-53.20	125.54	1.83	True
t_{10}	-24.04	-47.34	155.25	1.29	True	t_{30}	-44.79	-61.32	107.70	1.86	True
t_{15}	-29.14	-48.95	170.14	1.07	True	t_{35}	-47.04	-68.24	100.94	1.99	False

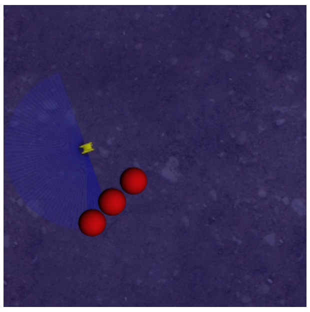
To better visualize the results from table 5.8, images were taken from the Gazebo simulation (figure 5.8), which allow a better vision of the trajectory taken by the vehicle. The images are also presented with a time step of 5 seconds, having figure 5.8(i) the complete trajectory.

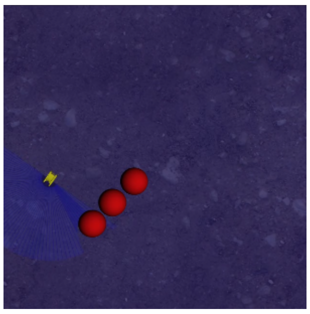

(a) t_{00}

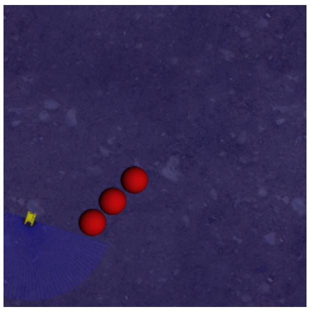

(b) t_{05}

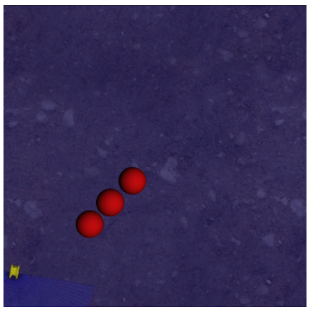

(c) t_{10}


(d) t_{15}


(e) t_{20}


(f) t_{25}


(g) t_{30}


(h) t_{35}

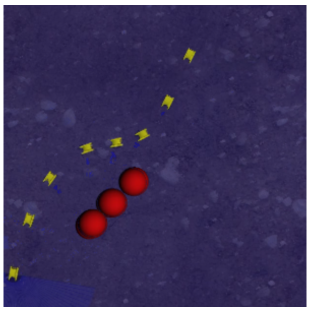

(i) Complete path

Figure 5.8: Big static obstacle collision avoidance.

Analyzing the results, it is observable that the vehicle started with the same velocity and heading until time step of t_{05}, where the obstacle enters the critical perimeter, having the danger boolean condition set to True, maintaining it until time step t_{30}, when it passes the big obstacle and resumes the mission. From time step t_{10} to t_{20}, the reduction of velocity was in order to facilitate the change in heading, in order to avoid collision.

Chapter 6

Conclusions

The ability to autonomously avoid collision, of both static and dynamic vehicles following the maritime rules, has been successfully implemented on the tested ASV. The algorithm developed is capable of working in any other maritime vehicle, in the condition that the inputs and outputs match the requirements specified throughout this document, and specifically explained in appendix D. The implementation of ROS on the ASV makes it much more versatile, and it is believed that it will facilitate the integration of other systems, such as the ones proposed in this thesis (section 2.2).

The Lidar was preferred compared to other ranging and measurement systems. Reason being that the ASV can run into situations which demand specially speed and accuracy.

The Dynamic Window Approach has been implemented for static obstacles, which are detected and mapped with the Lidar lasers. Some modifications were made on the algorithm in order to facilitate the implementation with ROS and also to reduce the computational effort. The obstacles detected with the Lidar lasers were represented as an array of coordinates for the DWA algorithm to loop and search the best path to take. In order to reduce the computational effort, that a huge array of obstacles would leave, all obstacles outside of the critical perimeter are deleted from the array.

The COLREGS node implemented for collision avoidance with other vehicles, follows the COLREGS rules. Regarding the fundamental part of detecting the mode which the vehicle should behave, the formulation presented in chapter 3 went through some changes compared to the cited paper. For the main logic of the node algorithm, a completely new one was developed, in order to better suit the ASV in test.

The choice for Python usage in most algorithms is due to being a high-level, interpreted and general-purpose dynamic programming language which focuses on code readability. In addition, there are pure Python client libraries developed for ROS integration. However, compared to C++, which is a native programming language, the Python programming language takes more time to run the same algorithm.

The usage of Gazebo simulation was of utmost importance, providing a space to debug and test the algorithm. Using Gazebo, which provides the simulated inputs, such as the IMU and GPS data, and receives the outputs from the algorithm, such as the desired velocity and heading, it is certain that switching to the physical model with the respective sensors, the algorithm would behave exactly the same.

In summary, the implemented system has a completely functional GNC: the Guidance framework provides the desired trajectory commands to the control system, such being provided by the collision avoidance algorithms or the path following algorithm; the Navigation framework estimates and updates the ASV's current and future states, with help of the sensors; the Control framework determines the proper control forces and moments, to be generated in conjunction with instructions provided by the guidance and navigation systems.

6.1 Future Work

At present, the system has only been tested in a virtual simulator. It is now essential to further test it in the physical world, where external forces for the control system might prove challenging.

This work had the assumption that every vessel possessed an AIS system integrated. In the case that no AIS system was integrated, only the DWA algorithm would be employed and the COLREGS rules would be disregarded, thus, an algorithm for vehicles with no AIS system should be implemented. Detecting the vehicle with a Lidar, and ascertain whether the obstacle is in fact a vehicle, could be a starting approach. In case heading, velocity, and position could be measured, the main algorithm provided in this thesis could be promptly used. Otherwise, a new algorithm to detect the mode of collision avoidance would be needed.

The author would like to advise that some modules in the algorithm should be changed to C++, in order to reduce the computational power needed. The objective function loops trough the space of heading and velocities available. With the computer used, some loops could take up to 1 or 2 seconds which is not ideal.

A Radar system is also advisable to be implemented, because some weather conditions may cause the Lidar system to restrict the operation, due to it being adversely affected by smoke, rain, and fog, thus the Radar system could be a complement.

The author would also like to advise that the implementation of SLAM (Simultaneous Localization and Mapping) algorithms, which were not used in this thesis as there was no necessity to generate a map of the world with obstacles in it should be made. However, if a SLAM algorithm is implemented, specially ROS SLAM packages (such as *Hector-SLAM*), it would improve the accuracy of the coordinates for the obstacles.

Bibliography

Carlos Almeida, Tiago Franco, Hugo Ferreira, Alfredo Martins, Ricardo Santos, José Miguel Almeida, João Carvalho, and Eduardo Silva. Radar Based Collision detection developments on USV ROAZ II. *OCEANS '09 IEEE Bremen: Balancing Technology with Future Needs*, 2009.

Morten Breivik and Thor I Fossen. Guidance Laws for Autonomous Underwater Vehicles. In Alexander V Inzartsev, editor, *Underwater Vehicles*, chapter 4. IntechOpen, Rijeka, 2009.

Christopher Byrne, Matthew Franklin, and Dylan Rodriguez. A Study of the Feasibility of Autonomous Surface Vehicles, 2012. URL `https://web.wpi.edu/Pubs/E-project/Available/E-project-121212-135500/unrestricted/ASV_IQP.pdf`. Accessed: 2018-02-30.

S. Campbell, W. Naeem, and G. W. Irwin. A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. *Annual Reviews in Control*, 36(2):267–283, 2012.

Nicola Ceccarelli and Steven Rasmussen. *Appendix B: Path Planning for UAVs*, volume 30, pages 141–158. SIAM, 2010.

Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons Ltd., 1 edition, 7 2011.

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

David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4): 1135–1145, 2016.

Hordur K. Heidarsson and Gaurav S. Sukhatme. Obstacle detection and avoidance for an autonomous surface vehicle using a profiling sonar. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 731–736, 2011.

Voemir Kunchev, Lakhmi Jain, Vladimir Ivancevic, and Anthony Finn. Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review. In Bogdan Gabrys, Robert J Howlett, and Lakhmi C Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, pages 537–544, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Zhixiang Liu, Youmin Zhang, Xiang Yu, and Chi Yuan. Unmanned surface vehicles: An overview of developments and challenges. *Annual Reviews in Control*, 41:71–93, 2016.

Gergely Magyar and Peter Sincak. Comparison study of robotic middleware for robotic applications. In *Emergent Trends in Robotics and Intelligent Systems*, volume 316 of *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2015.

MarineTraffic. What is the Automatic Identification System (AIS)? – MarineTraffic Help, 2017a. URL https://help.marinetraffic.com/hc/en-us/articles/204581828-What-is-the-Automatic-Identification-System-AIS-. Accessed: 2018-04-17.

MarineTraffic. What kind of information is AIS-transmitted?, 2017b. URL https://help.marinetraffic.com/hc/en-us/articles/205426887-What-kind-of-information-is-AIS-transmitted-. Accessed: 2018-04-17.

Ministry of defence of Portugal. Continental shelf submission of Portugal. *United Nations Conversion on the Law of the Sea*, pages 1–88, 2009. URL https://www.un.org/Depts/los/clcs_new/submissions_files/prt44_09/prt2009executivesummary.pdf. Accessed: 2018-02-27.

National Oceanic and Atmosferic Admnistration (NOAA). Oceans & Coasts, 2000. URL http://www.noaa.gov/oceans-coasts. Accessed: 2018-02-27.

Open Source Robotics Foundation. Gazebo tutorials, 2014. URL http://gazebosim.org/tutorials?tut=quick_start. Accessed: 01-09-2018.

Open Source Robotics Foundation. rqt_graph, 2014. URL http://wiki.ros.org/rqt_graph. Accessed: 2018-08-27.

Open Source Robotics Foundation. Ros filesystem concepts: Packages, 2015. URL http://wiki.ros.org/Packages. Accessed: 2018-04-19.

Open Source Robotics Foundation. Setup and configuration of the navigation stack on a robot, 2016. URL http://wiki.ros.org/navigation/Tutorials/RobotSetup. Accessed: 12-07-2018.

Open Source Robotics Foundation. roslaunch/xml, 2017. URL http://wiki.ros.org/roslaunch/XML. Accessed: 12-07-2018.

Open Source Robotics Foundation. Ros tutorials, 2018a. URL http://wiki.ros.org/Packages. Accessed: 2018-04-19.

Open Source Robotics Foundation. Introduction, 2018b. URL http://wiki.ros.org/ROS/Introduction. Accessed: 2018-08-26.

Eric S Raymond. AIVDM / AIVDO protocol decoding, 2016. URL http://catb.org/gpsd/AIVDM.html#_types_1_2_and_3_position_report_class_a. Access: 2018-05-09.

SNAME The Society of Naval Architects and Marine Engineers. Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. In *Technical and Research Bulletin*, volume 1-5, page 16, 1950. URL `http://kom.aau.dk/~nickoe/random/Sname1950.PDF`. Accessed: 2018-04-25.

Chiew Seon Tan. *A collision avoidance system for autonomous underwater vehicles*. PhD thesis, University of Plymouth, 1 2006. URL `https://core.ac.uk/download/pdf/29818298.pdf`. Accessed: 2018-02-27.

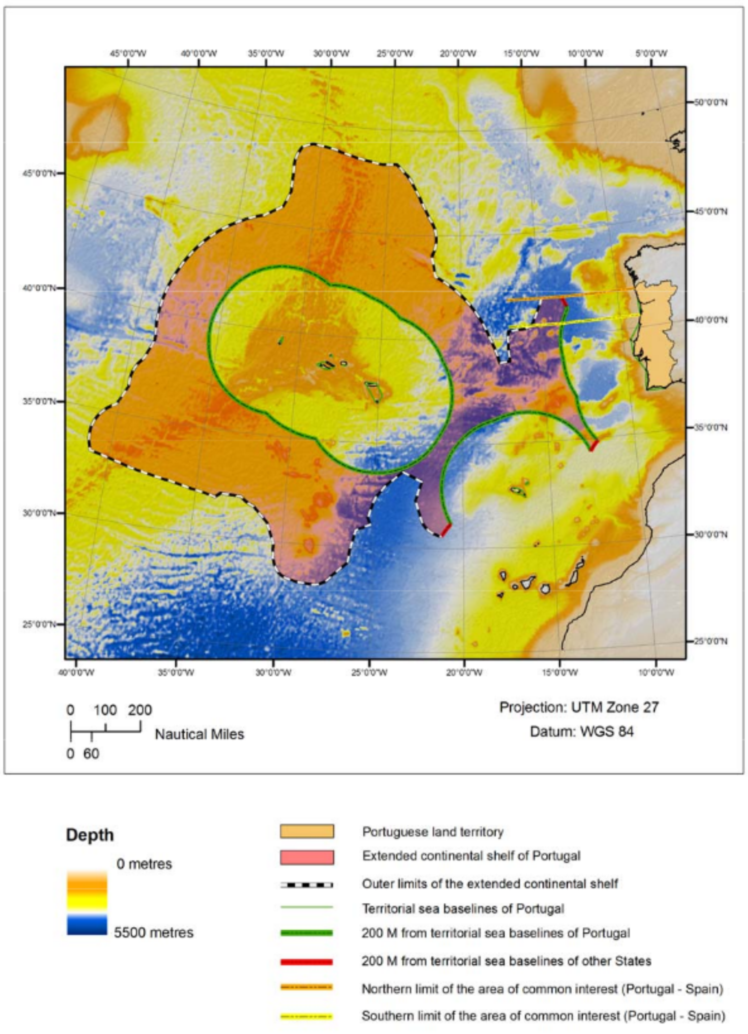
U.S. Department of Homeland Security United States Coast Guard. AIS Requirements, 2017. URL `https://www.navcen.uscg.gov/?pageName=AISRequirementsRev`. Accessed: 2018-04-17.

M Ventura. Colregs - international regulations for preventing collisions at sea. *Lloyd's Register Rulefinder*, pages 1–74, 2005.

Srdjan Vujicic, Đani Mohović, and Robert Mohović. A model of determining the closest point of approach between ships on the open sea. *PROMET - Traffic&Transportation*, 29:225, 04 2017.

Appendix A

Extended Continental Shelf of Portugal



0 100 200
Nautical Miles
0 60
Projection: UTM Zone 27
Datum: WGS 84
Depth
0 metres
5500 metres
Portuguese land territory
Extended continental shelf of Portugal
Outer limits of the extended continental shelf
Territorial sea baselines of Portugal
200 M from territorial sea baselines of Portugal
200 M from territorial sea baselines of other States
Northern limit of the area of common interest (Portugal - Spain)
Southern limit of the area of common interest (Portugal - Spain)

Figure A.1: Map of the outer limits of the extended continental shelf of Portugal. (Ministry of defence of Portugal, 2009)

Appendix B

Mode Algorithms

The ModeAlgorithm function is executed in every iteration, called by the MainAlgorithm (function presented in algorithm 8 in appendix C). The function starts with a mode that is already defined from the previous loop. The displayed conditions, allow the algorithm to enter every loop in case the mode is defined as Null, CPA, or in the same mode it is going to check. For example, in case the mode at the beginning of the function is StandOnX, it will only enter the algorithm 6. In case no mode is defined at the end of the function, it will check if there is really any risk with CheckModeCPA function.

Algorithm 1: Determining COLREGS Major Mode

```
Function ModeAlgorithm():                          // Executed each iteration of the behavior
    if $mode \in \{Null, CPA, GiveWayOT\}$ then
        $mode \leftarrow CheckModeGiveWayOT()$                           // Algorithm 2
    end
    if $mode \in \{Null, CPA, HeadOn\}$ then
        $mode \leftarrow CheckModeHeadOn()$                              // Algorithm 3
    end
    if $mode \in \{Null, CPA, StandOnOT\}$ then
        $mode \leftarrow CheckModeStandOnOT()$                           // Algorithm 4
    end
    if $mode \in \{Null, CPA, GiveWayX\}$ then
        $mode \leftarrow CheckModeGiveWayX()$                            // Algorithm 5
    end
    if $mode \in \{Null, CPA, StandOnX\}$ then
        $mode \leftarrow CheckModeStandOnX()$                            // Algorithm 6
    end
    if $mode \in \{Null, CPA\}$ then
        $mode \leftarrow CheckModeCPA()$
    end
```

The following algorithms are the corresponding check mode functions. Each have a part 1, being the absolute release check, a part 2, being the check release condition from the mode, and a part 3, which is the entry mode criteria. For those modes with submodes, there also exists a part 4, in order to determine the entry submode. These parts are presented in sequence with corresponding return calls, in case the conditions are met, thus, for the algorithm to reach the entry check condition, it must fail both the absolute release and the check release.

Algorithm 2: COLREGS GiveWayOT Mode

```
Function CheckModeGiveWayOT():                 // Called from within MainModeAlgorithm()
    if ($r > r_{pwt}$) then                                  // Part 1: Absolute Release check
        return Null
    end
    if ($mode == GiveWayOT$) then                  // Part 2: Check release from GiveWayOT
        if (($\alpha > 337.5$) or ($\alpha < 22.5$) or ($\dot{r} > 0$)) then      // GiveWayOT completed
            return CPA
        if (($submode == Port$) and ($180 < \beta < 355$)) then           // Switch submode
            return GiveWayOT, Starboard
        else if (($submode == Starboard$) and ($\beta < 5$)) then          // Switch submode
            return GiveWayOT, Port
        else                                                  // Mode/Submode unchanged
            return GiveWayOT
        end
    end
    if (($\alpha > 247.5$)or($\alpha < 112.5$)or($\neg pass^{os}_{cn}$) then   // Part 3: Check GiveWayOT entry
     criteria
        return CPA;
    end
    if ($pass^{os}_{cns}$) then                          // Part 4: Determine entry submode
        return GiveWayOT, Starboard;                          // GiveWayOT entry case A
    else
        return GiveWayOT, Port;                               // GiveWayOT entry case A
    end
```

Algorithm 3: COLREGS Head-On Mode

```
Function CheckModeHeadOn():                       // Called from within MainModeAlgorithm()
    if ($r > (2 \times r_{pwt})$) then                  // Part 1:  Absolute Release check
        return Null
    end
    if ($mode == HeadOn$) then                      // Part 2:  Check release from HeadOn
        if ($r > \breve{r}_{cpa}$) and ($\dot{r} > 0$) then                      // HeadOn completed
            return CPA
        else if $aft^{os}_{cn}$ then                          // Ownship passed Contact
            return CPA
        else if $star^{os}_{cn}$ and $star^{cn}_{os}$ then       // Port-Port may be no longer advisable
            $\beta_\delta \leftarrow (\beta - \phi)$
            $\alpha_\delta \leftarrow (\alpha - \phi)$
            if (($\beta_\delta > \frac{\phi}{2}$) or ($\alpha_\delta > \frac{\phi}{2}$) or (($\beta_\delta + \alpha_\delta) > \frac{2\times\phi}{3}$)) then
                return CPA
            end
        else                                                  // Mode unchanged
            return HeadOn
        end
    end
    if (($mode == HeadOn$) or ($mode == Null$)) then          // Part 3:  Check HeadOn entry
     criteria
        if ($|[\alpha]^{180}| \leq \phi$) and ($|[\beta]^{180}| \leq \phi$) then
            return HeadOn;
        else
            return CPA;
        end
    end
```

Algorithm 4: COLREGS StandOn Overtaken Mode

```
Function CheckModeStandOnOT():             // Called from within MainModeAlgorithm()
    if ($r > r_{pwt}$) then                          // Part 1:  Absolute Release check
        return Null
    end
    if ($mode == StandOnOT$) then                 // Part 2:  Check release from StandOnOT
        if ($\dot{r} > 0$) then                                  // StandOnOT completed
            return CPA
        else
            return StandOnOT
        end
    end
    if (($\beta < 112.5$) or ($\beta > 247.5$) or ($\neg pass^{cn}_{os}$)) then    // Part 3:  StandOnOT entry criteria
        return CPA;
    else
        return StanOnOT
    end
```

Algorithm 5: COLREGS GiveWay Crossing Mode

```
Function CheckModeGiveWayX():              // Called from within MainModeAlgorithm()
    if ($r > r_{pwt}$) then                          // Part 1:  Absolute Release check
        return Null
    end
    if ($mode == GiveWayX$) then                  // Part 2:  Check release from GiveWayX
        if ($r > \check{r}_{cpa}$) and ($\dot{r} > 0$) then           // GiveWayX completed
            return CPA
        else if ($\alpha \leq 180$) then                // Ownship on Contact starboard side
            return CPA
        else                                              // Mode/Submode unchanged
            return GiveWayX
        end
    end
    if (($\alpha < 247.5$) or ($\beta > 112.5$)) then          // Part 3:  GiveWayX entry criteria
        return CPA;
    if (($cross^{os}_{xcnb}$) and ($r^{os}_{xcnb} > (\hat{r}_{cpa} + \check{r}_{cpa})/2$)) then      // Part 4:  Determine Submode
        return GiveWayX, Bow;
    else
        return GiveWayX, Stern
    end
```

Algorithm 6: COLREGS StandOn Crossing Mode

```
Function CheckModeStandOnX():                    // Called from within MainModeAlgorithm()
    if $(r > r_{pwt})$ then                           // Part 1: Absolute Release check
        return Null
    end
    if $(mode == StandOnX)$ then                      // Part 2: Check release from StandOnX
        if $(\dot{r} > 0)$ then                             // StandOnX completed
            return CPA
        else                                              // Mode/Submode unchanged
            return StandOnX
        end
    end
    if $((\alpha > 112.5)$ or $(\beta < 247.5)$ or $(\dot{r} \geq 0))$ then   // Part 3: StandOnX entry criteria
        return CPA;
    else
        return StandOnX
    end
```

Appendix C

Main Algorithm

Algorithm 7: Vehicle Top-level behaviour

```
1  Define reference values $ref$          // $headOn\ angle,\ lat\ and\ lon\ reference\ values,\ ...$
2  While ROS is running :                 // Top level behavior
3      $Contact \leftarrow$ Subscribe Contact AIS data                 // Section 4.5
4      Convert Contact's latitude and longitude into local coordinates $(x, y)$
5      $\hat{r}_{cpa}$ ; $\check{r}_{cpa} \leftarrow$ Define dynamic perimeters of safety   // Section 4.4
6      $Ownship \leftarrow$ Subscribe Ownship data                     // Subsection 4.6.1
7      $mode,\ submode,\ \theta_{desired},\ v_{desired},\ yaw_{desired} \leftarrow$
        $MainAlgorithm(Ownship\,; Contact\,;\ \hat{r}_{cpa}\ ;\ \check{r}_{cpa}\ ;\ ref)$   // Algorithm 8
8      Publish $v_{desired}\ and\ yaw_{desired}$ to the control module
9      Publish to AIS the vehicle updated data and intentions
```

Algorithm 8: Determining desired heading and velocity

```
1  Function MainAlgorithm():              // Second level entry behavior
       Input: $Ownship\ updated\ data$ ; $Contact\ updated\ data$ ; $\hat{r}_{cpa}$ ; $\check{r}_{cpa}$ ; $ref$
                                          // Explained in section 4.6
2      Update Ownship and Contact parameters    // Loop through functions in chapter 3
3      Assess risk of collision and Start Manoeuvre Boolean Condition
4      $mode,\ submode \leftarrow$ ModeAlgorithm()            // Algorithm 1
5      if $(mode\,! =\ Null)$ then
6          Set risk function and heuristic function    // Heuristic depends on the mode.  Loops
            through the space of heading and velocities.  For each, updates the vehicle
            parameters
7          Set Objective function and determine desired heading and velocity   // Section 3.10
8      else                                   // Proceed with mission
9          Keep mission control heading and velocity
10     end
11     Convert heading to yaw
12     Send desired yaw and velocity to control module
```

Appendix D

Software Set-Up and Installation

This appendix presents the necessary steps to implement the collision avoidance system to run in a vehicle. Some prerequisites are needed, such as a vehicle able to function with ROS as will be further explained in the following sections.

D.1 Installing ROS and Ubuntu

For this section it is needed the following:

- Laptop/computer with Ubuntu installed
- Vehicle able to take in ROS commands (optional)

D.1.1 Ubuntu and ROS on your personal computer

The version used and tested was Ubuntu 16.04 (Xenial Xerus) and can be downloaded from the Ubuntu's main page[1].

After having Ubuntu installed, it is time to install ROS. The version used and tested for the algorithm which was presented is ROS Kinetic[2].

The need to install ROS in the computer can be, either to use a simulated vehicle in a software such as Gazebo (Open Source Robotics Foundation, 2014), or in case the user is in possession of a vehicle able to also run ROS, for a debugging phase and to exchange information.

D.2 Getting started with ROS

It is not the purpose of this thesis to point out how to build a robot or vehicle. The intention is to specify the necessary inputs needed from the system and the type of outputs which it gives out to the control frame. Some packages are mentioned throughout the thesis and some will also be mentioned here.

[1] https://www.ubuntu.com/download/desktop
[2] http://wiki.ros.org/kinetic

The following commands create a personal workspace on ROS, in case one is not yet created.

Create the workspace:

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspaces
```

In order to source the workspace each time a new terminal is opened, the following command open the bashrc-file:

```
$ sudo gedit ~/.bashrc
```

Then, the following line should be added at the bottom of the bashrc-file:

```
$ source gedit ~/catkin_ws/build/setup.bash
```

D.3 Package installation

The installation of the algorithms nodes can be done via git clone:

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/uc2013171665/asv_colregs.git
$ cd..
$ catkin_make
```

After this step, the following files are added to the working directory:

```
catkin_ws
├─ build
├─ devel
└─ src
   ├─ ASV_messages
   │  ├─ msg
   │  │  ├─ AIS.msg
   │  │  ├─ debug.msg
   │  │  ├─ desired.msg
   │  │  └─ debug2.msg
   │  ├─ CMakeLists.txt
   │  └─ package.xml
   ├─ ASV
   │  ├─ launch
   │  │  ├─ ASV.launch
   │  │  └─ ASV2.launch
   │  ├─ src
   │  │  ├─ Ownship_vehicle.py
   │  │  ├─ Contact_vehicle.py
   │  │  ├─ lidar.py
   │  │  └─ COLREGS_utils
   │  │     ├─ functions.py
   │  │     ├─ lat_lon_convert.py
   │  │     ├─ main_algorithm.py
   │  │     ├─ mode_algorithm.py
   │  │     └─ quat_euler_yaw.py
   │  ├─ CMakeLists.txt
   │  └─ package.xml
   └─ CMakeLists.txt
```

In this repository, there also exists the *contact* node, used for test and debugging. Both nodes are called from the ASV.launch file and are generated from the Ownship_vehicle.py and Contact_vehicle.py files respectively.

Both of these, utilize the python files from within COLREGS_utils folder. The functions.py has the functions presented in chapter 3. The lat_lon_convert.py has the conversion of latitude, longitude and altitude, given by the GNSS, into the local coordinates used in the algorithm. The main_algorithm.py, as the name indicates, is the main algorithm (algorithm 8). The mode_algorithm.py possesses all the mode algorithms (algorithm 1 and appendix B). The quat_euler_yaw.py has two conversions: the conversion of quaternions into euler angles, from which we get yaw; and the conversion from yaw to heading.

The Ownship_vehicle.py and Contact_vehicle.py, mainly call the main_algorithm.py and subscribe and publish to other nodes, which are the inputs and outputs of the algorithm developed.

D.3.1 Inputs and Outputs

The physical vehicle, or the simulated vehicle, must have working sensors (described throughout this thesis), such as the proprioceptive (subsection 2.2.1) and the exteroceptive (subsection 2.2.2) sensors. Thus, the inputs needed are: a viable odometry, GNSS data, and AIS data from other vehicles. For outputs, the algorithm produces data for the control module, AIS data, and debugging data. All these are published and subscribed with the respective topic names. Thus, in order to implement in other vehicle, open the Ownship_vehicle.py file, and change the topic names to the ones used by the vehicle, and take into account the message used by the topic.

Odometry

The Odometry is essential to know the *ownship's* position, heading, and velocity, which are necessary for the algorithm. Usually, odometry is used with the message type nav_msgs/Odometry. It is also this type of message used in order to set up the robot with the *navigation* package (Open Source Robotics Foundation, 2016). The name of the topic can be set to the preferred, having to be consistent with the one used by the vehicle.

GPS

The GPS data, from the GNSS system, is used to get the local coordinates, as previously explained. GNSS and sensors data are frequently used with the message type sensor_msgs/NavSatFix. Again, the name of the topic can be set to the preferred, having to be consistent with the one used by the vehicle.

AIS

The AIS is subscribed and published by the *Ownship* node. In the algorithm, the current topic name is “chatter”, with the message type being ASV_messages/AIS.

Control module

The control block is accountable for the calculation of the necessary forces/moments to be passed on to the thrusters, and to achieve the control objective, which consists of ensuring that the ASV follows the desired path as accurately as possible. A reduction to the six-DOF[3] model can be made to only consider motion in surge/forward, sway/lateral and heading/yaw, neglecting both roll, pitch, and heave as to maintain model simplicity (Fossen, 2011).

[3]Degrees Of Freedom

To derive the motion equations describing the ASV kinematics, the definition of two reference frames is mandatory. The notation that will be used from now on is the one defined by SNAME (SNAME The Society of Naval Architects and Marine Engineers, 1950), with slight differences as in (Fossen, 2011).

- The body-fixed $\{b\}$ is the non-inertial frame, composed by the axes $\{x_b, y_b, z_b\}$, and the O_b is chosen, for simplicity, to coincide with the Center of Gravity (CG) of the ASV.
- The local NED (North-East-Down) $\{n\}$ is the inertial reference frame, composed by the orthonormal axes $\{x_n, y_n, z_n\}$, and the origin is represented by O_n.

In order to determine the position and orientation of the vehicle, three independent coordinates are needed, namely (x, y, ψ), expressed in the inertial frame, $\{n\}$. For linear and angular velocity, the three coordinates are (u, v, r), and (X, Y, N) for control forces/moments, both expressed in the non-inertial frame, $\{b\}$. The three motion components can be expressed as in Table D.1, or, in the vector form, as:

- $\eta = [x \;\; y \;\; \psi]^T$ expressed in the $\{n\}$ frame;
- $v = [u \;\; v \;\; r]^T$ expressed in the $\{b\}$ frame;
- $\tau = [X \;\; Y \;\; N]^T$ expressed in the $\{n\}$ frame;

Table D.1: Corresponding notation following SNAME.

	Forces/moments	linear/angular speeds	positions/angles
Motions in the x-direction (surge)	X	u	x
Motions in the y-direction (sway)	Y	v	y
Rotations about the z-axis (yaw)	N	r	ψ

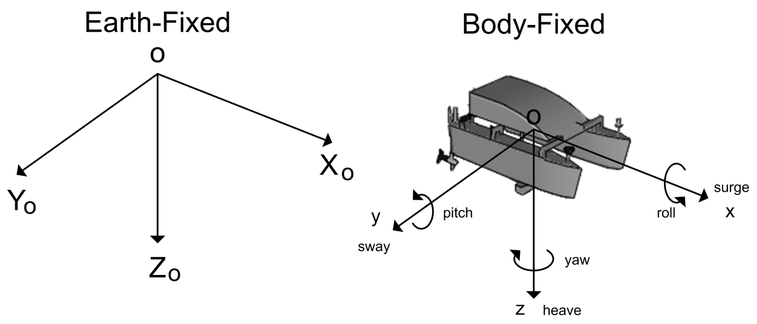


Figure D.1: Earth-Fixed and Body-Fixed reference frames. Adapted from (Fossen, 2011).

A control frame ought to be implemented. If in need to know how to create the frame, (Fossen, 2011) is a complete reading on control for maritime vehicles.

The *ownship* node publishes 4 arguments which the control block is subscribing: desired.yaw; desired.velocity; desired.vid; Reset. Therefore, the control frame of the vehicle must subscribe to a topic with these arguments as the input. The name of the topic can be defined as preferred, taking into account that both node and the vehicle must have the coherent topic name, and the message arguments have to be as detailed.

The desired.yaw and desired.velocity are as the names imply. The desired.vid was introduced in order to identify which vehicle was publishing the data in the simulation, as both *ownship* and *contact* published to the same topic, thus the need to introduce this reference. The Reset is a boolean variable which when set true, the values sent are reseted by the control module.

Debugging

A debugging message is essential to facilitate corrections and to understand the anomalies which appear through the development. The *Ownship* node publishes a topic named "DEBUG" with the message type being ASV_messages/debug. The debug.msg arguments are as presented in table D.2

Table D.2: ROS debug message.

debug.msg

Data Type	**Variable Name**	**Unit**
int32	counter	-
int32	mode_cn	-
int32	mode_cn	-
float64	lat_cn	[deg]
float64	lon_cn	[deg]
float32	SOG_cn	[m/s]
float32	COG_cn	[deg]
int32	length_cn	[m]
float32	x_cn	[m]
float32	y_cn	[m]
int32	mode_os	-
int32	mode_os	-
float64	lat_os	[deg]
float64	lon_os	[deg]
float32	SOG_os	[m/s]
float32	COG_os	[deg]
int32	length_os	[m]
float32	x_os	[m]
float32	y_os	[m]
float64	Start_Evasion	[m]
float64	r	[m]

Laser scan

For static obstacles, the Lidar is used to scan data in order to detect and generate an array of obstacles for the algorithm. A Lidar system or a simulated laser scan data, are needed for the subscription by the algorithm, having the topic being /mybot/laser/scan, with the msg type being sensor_msgs.msg.

Debugging 2

A debugging message was also implemented when debugging the DWA algorithm, brought by the *lidar* node. The *lidar* node publishes a topic named “DEBUG2” with the message type being ASV_messages/debug2. The debug2.msg arguments are as presented in table D.3

Table D.3: ROS debug2 message.

debug2.msg

Data Type	Variable Name	Unit
int32	counter	-
bool	danger	-
float32	v	[m/s]
float32	w	[rad/s]
float32	yaw	[rad]
float32	x	[m]
float32	y	[m]
float32	dist_to_goal	[m]

D.4 Launch the program

As can be seen in the directory tree (subsection D.3), the directory /catkin_ws/src/ASV/launch has the file ASV.launch and ASV2.launch. The launch file documentation can be seen in (Open Source Robotics Foundation, 2017), which describes the XML format used for roslaunch .launch files. The launch file has most of the arguments in comment as default, as it will depend on its use. It can be used to launch a Gazebo simulation, define the vehicle properties, or it could be used to launch the physical vehicle. In the ASV.launch, the only argument not in comment is the COLREGS node launch, which is independent of the case, whereas in the ASV2.launch it is the LIDAR node. The COLREGS algorithm and the DWA algorithm were separated to facilitate the testing and debugging.

Having defined the arguments in the desired launch file, the launch is done following one of the command:

```
$ roslaunch ASV ASV.launch
```

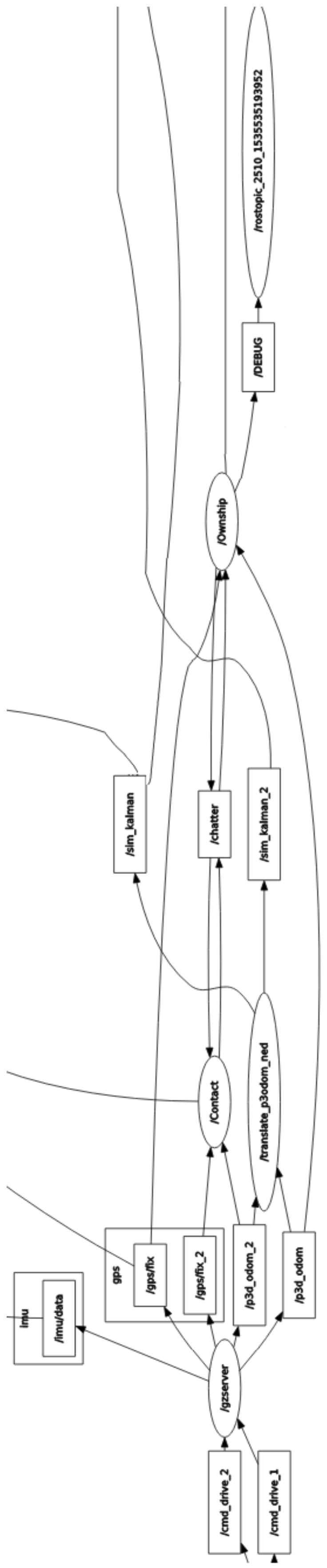
```
$ roslaunch ASV ASV2.launch
```

D.5 rqt graph

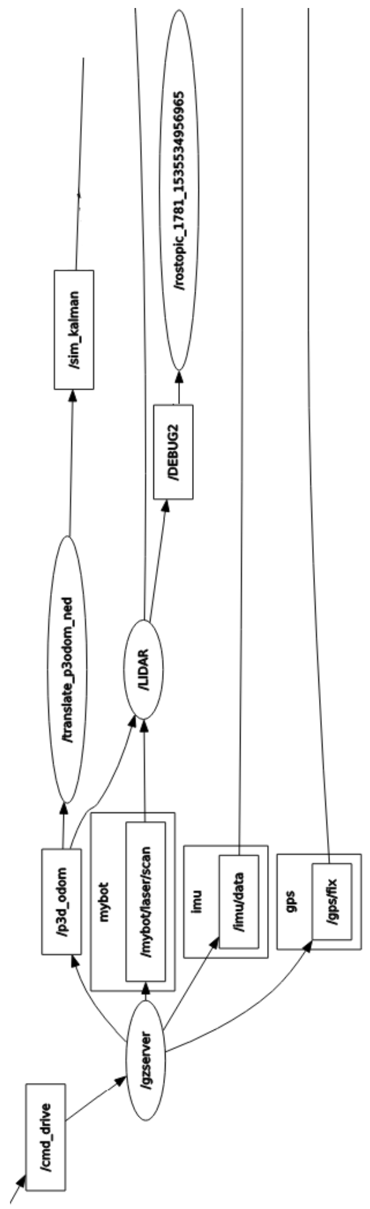
The rqt_graph provides a GUI plugin for visualizing the ROS computation graph (Open Source Robotics Foundation, 2014). Figure D.2 has a cropped part of the outputs of the rqt_graphs from COLREGS and DWA algorithms, providing a better visualization of the nodes and topics used. The nodes are presented in circular geometries, and the topics are in boxes.

In figure D.2(a), we have in the middle two nodes, the */Ownship* node and the */Contact* node, which are subscribing to the respective GPS and odometry topic, and both subscribe and publish to the /chatter topic which has the AIS messages. The output for the control is not visualized in this rqt_graph, however it is send and treated in the control module which then produces the commands for the thrusters, which are on the left side of the rqt_graph with the topic names /cmd_drive1 and /cmd_drive2. Because the test and debugging is realized on a virtual simulator (Gazebo), they are being subscribed by Gazebo's node, /gzserver, which is also responsible for publishing the sensor topics. For debugging, in the simulation, a node is subscribing to the /DEBUG topic, published by the */Ownship* node with the message debug.msg.

In figure D.2(b), the main node is now called */LIDAR* which is subscribing to the odometry topic. With a Lidar system implemented, the node is now subscribing to the data from the laser scan, which is in the topic /mybot/laser/scan. The output for the control is also not visualized in this rqt_graph, however it is send and treated in the control module which then produces the commands for the thrusters, which are on the left side of the rqt_graph with the topic name /cmd_drive. Gazebo is again used, where the input for thrusters are being subscribed by Gazebo's node (/gzserver), which is also responsible for publishing the sensor's topics. For debugging, in the simulation, a node is subscribing to the /DEBUG2 topic, published by the */LIDAR* node with the message debug2.msg.



(a) COLREGS



(b) DWA

Figure D.2: Rqt graph.