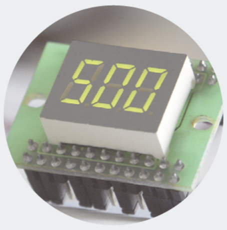
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Control-law for Oil Spill Mitigation with a team of Heterogeneous Autonomous Vehicles

DIOGO COELHO PEDROSA
julho de 2018

POLITÉCNICO
DO PORTO

Control-law for Oil Spill Mitigation with a team of Heterogeneous Autonomous Vehicles

Diogo Coelho Pedrosa,
Nº 1120331

Master in Electrical and Computer Engineering
Branch of Autonomous Systems

July 26, 2018

Dissertation for partial satisfaction of the requirements of the Master in Electrical and Computer Engineering

Candidate: Diogo Coelho Pedrosa,
Nº 1120331

Supervisor: Dr. André Miguel Pinheiro Dias
apd@isep.ipp.pt

Master in Electrical and Computer Engineering
Branch of Autonomous Systems

Agradecimentos

Gostaria de utilizar esta secção para, em primeiro lugar, agradecer ao meu orientador Eng.º André Dias, por me ter proporcionado a oportunidade de realizar a minha dissertação num tema interessante e com tanto impacto como este e pela ajuda disponibilizada ao longo da mesma.

A toda a "família" do laboratório de sistemas autónomos (LSA) que me acompanhou e forneceu ajuda, sempre que necessária, durante este percurso.

A todos os meus amigos e colegas de curso que participaram nesta caminhada comigo e me ofereceram apoio na realização desta dissertação, sem os quais não chegaria à sua conclusão, Alexandre Amaral, André Filipe, Caio Lomba, Denys Sytnyk, Eduardo Soares, Fábio Azevedo, Henrique Silva, Ricardo Pereira, Tiago Miranda, Tiago Pereira. Um enorme agradecimento ao meu amigo de longa data Pedro Meireles por toda a amizade e apoio ao longo destes árduos ultimos anos.

A toda a minha família, pela educação que me proporcionou, pelos esforços efetuados na minha formação académica e por estarem presentes sempre que necessitei.

Por fim, um agradecimento muito especial à minha namorada, Lénia Carmo, pela paciência, carinho, apoio e dedicação demnstrados durante a elaboração desta dissertação.

Diogo Coelho Pedrosa

This page was intentionally left blank.

Abstract

Oil spill incidents in the sea or harbours occur with some regularity during exploration, production, and transport of petroleum products. In order to mitigate the impact of the oil spill in the marine life, immediate, safety, effective and eco-friendly actions must be taken. Autonomous vehicles can assume an important contribution by establishing a cooperative and coordinated intervention.

This dissertation presents the development of two path planning control-laws, the first one an autonomous surface vehicle (ASV) being able to contour the oil spill while is deploying microorganisms and nutrients (bioremediation) capable of mitigate and contain the oil spill spread, and the second one for a unmanned aerial vehicle (UAV) in order to perform the coverage of the entire spillage area with the same microorganisms and nutrients deployment capabilities.

In order to validate both methods, a simulation environment was developed in Gazebo with a oil spill scenario, an ASV and an UAV.

Field tests have been conducted in the Leixões Harbour in Porto, Portugal.

Keywords:
Path planning, oil spill, detection, mitigation, simulation, Gazebo, PX4, UAV, STORK I, ASV, ROAZ II, Artificial Potential Field, Normal Vectors.

Resumo

Incidentes relacionados com derrames de petróleo no oceano ou em portos ocorrem com alguma regularidade, durante a exploração, produção e transporte de petróleo e seus derivados. Para mitigar o impacto desses derramamentos na fauna e flora marinha de uma forma imediata, segura, efectiva e amiga do ambiente novas ações são necessárias. Veículos autonomos podem providenciar uma importante contribuição establecendo uma intervenção cooperativa e coordenada.

Esta dissertação apresenta o desenvolvimento de dois algoritmos de controlo para o planeamento de trajectórias, a primeira para um veículo de superfície autónomo (ASV) ser capaz de contornar o perímetro do derrame enquanto distribui microorganismos e nutrientes (bio-remediação), capazes de mitigar e conter a propagação do derramamento de petróleo e a segunda para um veículo aéreo não-tripulado (UAV) ser capaz de cobrir todo a área de derrame enquanto distribui os mesmos microorganismos e nutrientes.

De forma a validar ambos os métodos, um ambiente de simulação foi desenvolvido em Gazebo com cenário do derrame de petróleo, um ASV e um UAV.

Testes de campo foram realizados no porto de Leixões, no Porto, Portugal.

Palavras-Chave:
Path planning, derrame de petróleo, deteção, mitigação, simulação, Gazebo, PX4, UAV, STORK I, ASV, ROAZ II, Artificial Potential Field, Normal Vectors.

Contents

This page was intentionally left blank.

List of Figures

This page was intentionally left blank.

List of Tables

This page was intentionally left blank.

List of Acronyms

ASV	Autonomous Surface Vehicle
AUV	Autonomous Underwater Vehicle
APF	Artificial Potential Field
CCD	Charge-Coupled Device
Co3-AUVs	Cooperative Cognitive Control for Autonomous Underwater Vehicles
DARPA	Defense Advanced Research Projects Agency
DART	Dynamic Animation and Robotics Toolkit
DRC	DARPA Robotics Challenge
DVL	Doppler Velocity Log
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICRA	International Conference on Robotics and Automation
IMU	Inertial Measurement Unit
ISEP	Instituto Superior de Engenharia do Porto
LSA	Laboratório de Sistemas Autónomos
MORSE	Modular OpenRobots Simulation Engine
NASA	National Aeronautics and Space Administration

ODE	Open Dynamics Engine
OSG	OpenSceneGraph
OSRF	Open Source Robotics Foundation
PRM	Probabilistic RoadMap
RAUVI	Reconfigurable AUV for Intervention Missions
ROCK	Robot Construction Kit
ROS	Robot Operating System
ROSM	Robotic Oil Spill Mitigation
SAR	Synthetic Aperture Radar
SDF	Simulation Description Format
SLAR	Side-Looking Airborne Radar
SRC	Space Robotics Challenge
UAV	Unmanned Aerial Vehicle
URDF	Unified Robot Description Format
USARSim	Unified System for Automation and Robot Simulation
UUV	Unmanned Underwater Vehicle
UWSim	UnderWater Simulator
V-REP	Virtual Robot Experimentation Platform
XML	eXtensible Markup Language
YARP	Yet Another Robot Platform

Chapter 1

Introduction

Marine oil spills have a large economic and ecologic impact in the world community, with high losses in the marine life in the ecosystems. Oil spill incidents occur with some regularity during the exploration, production, and transport of petroleum products[4][5]. Recently, in 2010, the Deepwater Horizon oil spill in the Gulf of Mexico, depicted in Figure 1.1, has been considered one of the largest accidental marine oil spill in the history of the petroleum industry with more than 210 million gallons of crude oil released in the ocean surface over 180.000 km^2 with the oil spill cleaning process involving over 39.000 personnel, 5000 vessels and 110 aircrafts[6]. In 2002 at coast of Galicia (North of Spain), the tanker Prestige sank at 250km from the coast and spilled more than 60.000 tons of oil crude over thousands of kilometres causing great harm to the local fishing industry.

Figure 1.1: Deepwater Horizon platform explosion on April 20, 2010.

The current oil spill cleaning technology includes physical (e.g. controlled burning; absorbing) and chemical (e.g. dispersing) actions, which is largely constrained by maritime conditions. Though these treatments are important to rapidly control the diffusion and drift of the oil, they are not suitable for ecological restoration. Recently, bioremediation using microorganisms to degrade the remaining spilled oil has been proposed as a cost-effective alternative to the use of chemical additives[7]. In order to improve the oil spill mitigation, *in situ* operational technologies must be developed to ensure immediate, safety, effective and eco-friendly actions to minimise the environmental damages[8]. Following these requirements, several autonomous vehicles have been developed, such as the Seaswarm (MIT Senseable City Lab)[9] and Protei: Open Source Sailing Drone[10]. Common to both autonomous vehicles is the oil spill mitigation being based in a system for ocean-skimming and oil removal, which is a limitation in future interventions due to the impact of the oil in the mechanical parts.

This dissertation proposes to address the development of two path planning control-laws, the first for an autonomous surface vehicle (ASV) being able to contour the oil spill while is deploying microorganisms and nutrients (bioremediation) capable of mitigate and contain the oil spill spread and the second for a unmanned aerial vehicle (UAV) allowing it to perform the coverage of the entire spillage area with the same microorganisms and nutrients deployment capabilities. In order to evaluate the path-planning developed algorithms, a simulation was also addressed during the dissertation with the goal of creating an oil spill scenario and the subsequent mitigation actions in a robotic simulator.

1.1 Dissertation Scope

This dissertation developed in the context of the master in Electrical and Computer Engineering in the branch of Autonomous Systems of the School of Engineering of the Polytechnic Institute of Porto (ISEP). It is also based in the FCT - Portuguese Foundation for Science and Technology as part of project ROSM – Robotic Oil Spill Mitigation (POCI-01-0145-FEDER-24055) and by the EU SpilLess project: First-line response to oil spills based on native microorganism cooperation[1], through Blue Labs: innovative solutions for maritime challenges program. (EASME/EMFF/2016/1.2.1.4/02/SI2.749374 -SpilLess). In both projects, the goal is the development of an autonomous and coordinated action able to increase the efficiency of the bioremediation process, by deploying

[1]https://ec.europa.eu/easme/en/first-line-response-oil-spills-based-native-microorganisms-cooperation

microorganisms and nutrients on identified targets, with the required amounts, and by making oil spill combat from air (UAV) and water surface (ASV), as depicted in Figure 1.2. The UAV is responsible for detecting the oil spill area through a thermographic camera and combat the leakage inner areas by spreading lyophilized native microbial consortium dust over the oil spill, while the ASV release the product mixed with water on the stroke border areas.

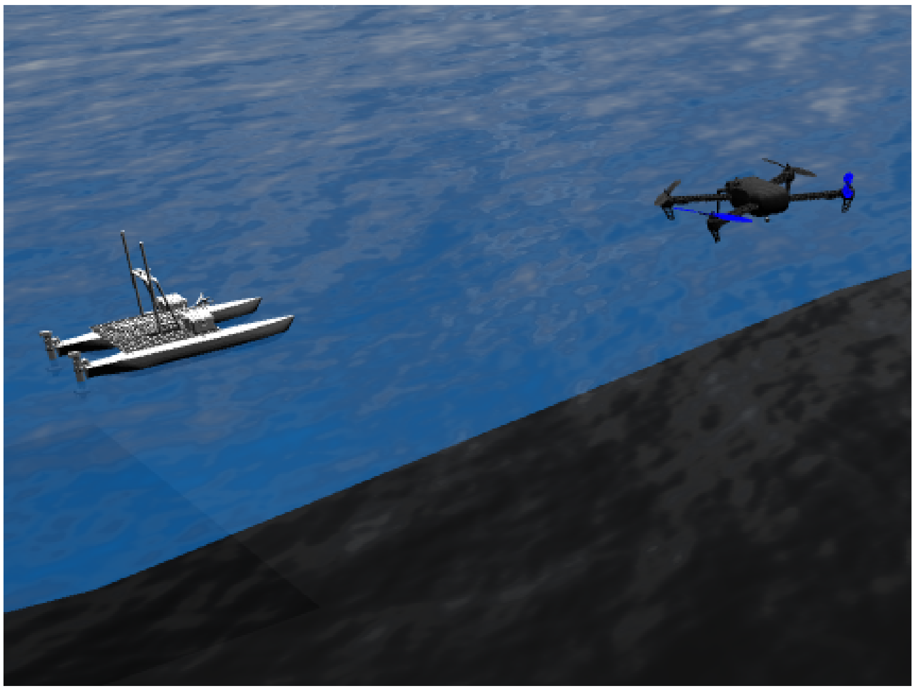


Figure 1.2: Oil spill simulation scenario. Cooperative ASV/UAV oils spill perception and mitigation(bioremediation).

1.2 Background and motivation

During the last years, the LSA from ISEP, has played a fundamental role in the development of autonomous robotic systems to operate in distinct scenarios, aerial, terrestrial or aquatic (surface or underwater), with applications ranging from monitoring and mapping to search and rescue. In this environment and within the scope of SpilLess and ROSM projects, the need to perform perception and navigation algorithms in a simulation environment is a challenging requirement, and therefore a motivation for this work.

This dissertation will try to reduce the impacts of oil spillage incidents on the environment with an attempt to develop a mitigation strategy more efficient and robust than the physical and chemical like controlled burning, ocean-skimming, absorbing and dispersion, that will be achieved through an action of bioremediation. The problem of simulating a complex manoeuvre for an ASV robot and for an UAV and a comparative analysis of simulators for marine robotics are presented.

1.3 Objectives

Having in mind the requirements of the SpilLess and ROSM projects described above and the challenges imposed by an oil spill, the main objectives of this dissertation are:

- Integrate the first line responses to oil spill incidents with the support of heterogeneous autonomous vehicles;
- Cooperative perception, combine information from different sensors with different levels of oil spill detection accuracy;
- Cooperative and coordinated manoeuvres in order to maximise the efficiency (maximum overlap of the bacteria-oil area);
- UAV: Oil spill area detection, through the thermographic camera and combat the leakage inner areas by spreading lyophilized native microbial consortium dust over the oil spill, while ensuring optimal coverage;
- ASV: Contour the oil spill area while releasing the powder mixed with water on the stroke border areas;
- Increase the overall efficiency of the oil spill combat missions;
- Decrease the overall time of reaction and mission costs;
- Development of a realistic simulation scenario able to identify and represent the mitigation manoeuvres;
- Solution validation through field tests and proof-of-concept through demonstration at a *"quasi-real"* scenario.

1.4 Dissertation Structure

This dissertation is organised in seven chapters. In Chapter 1 a brief introduction of this dissertation, its scope and objectives, the background and motivation are presented.

In Chapter 2, a study of methods and approaches already developed by the scientific community for autonomous aerial and surface vehicles path planning methods, oil spill detection and mitigation strategies are described.

In Chapter 3, the fundamentals that support this dissertation focusing mainly on the trajectory planning techniques and robotic simulators that could be the support to develop and study a novel method for oils spill mitigation are presented.

In Chapter 4, the proposed architecture and control-law manoeuvres for the ASV and UAV are present and detailed.

In Chapter 5, the system implementation with a description of the vehicles used for the trials are presented. In this chapter the author addresses also the creation of an oil spill scenario in the simulator and the mitigation manoeuvres representation.

In Chapter 6, the simulation results for each vehicle in the presence of different obstacles and also the field tests performed in Leixões harbour with the developed path planning in the presence of a simulated oil spill.

Finally, in the Chapter 7 the conclusions taken from the development of this dissertation are presented as well as some lines of future work.

This page was intentionally left blank.

Chapter 2

State of the Art

In this chapter an analysis of the methods and algorithms already developed and used by the community for path-planning and obstacle avoidance, in autonomous vehicles, are addressed.

An analysis of the current strategies applied to oil spillage detection and mitigation is also presented in this chapter.

2.1 Path planning for autonomous vehicles

The applications of the autonomous mobile robot in many fields such as industry, space, defence and transportation, and other social sectors are growing day by day[11]. To achieve those complex operations, path planning algorithms capable of handling static and dynamic environments are necessary.

A standard path planning algorithm will simply try to find the optimal path, clear of obstacles, from the initial position to the goal, minimising the distance and therefore energy and time spent. For some specific applications, a coverage path planning is necessary, in that case, the algorithm computes an optimal path that ensures a complete coverage of the interest area, while avoiding the obstacles present in the environment. Literature surveys of standard and coverage path planning techniques, used for mobile robot navigation, can be found in [11] and [12], respectively.

2.1.1 UAVs Path Planning Methods

The specific path planning algorithm for an UAV attempts to compute an optimal and collision free path, while having the physical and temporal constraints into account.

A large portion of the studied implementations can be included into one of the following categories[13][14]:

- **Sampling-based methods:** These approaches require the previous knowledge of the environment for its sampling into nodes. They can be divided into two subcategories, passive methods like Probabilistic Roadmaps (PRM), K-PRM, S-PRM, 3D Voronoi, Visibility Graphs, *etc.*, that generate road-maps from the initial node to the target node but are not capable of determine the optimal path having to resort to an additional algorithm and active methods like Rapidly-exploring Random Trees (RRT), Dynamic Domain RRT, RRT-Star, Artificial Potential Field (APF), *etc.*, that are capable of both processes.

  These algorithms are of easy implementation and appropriate for static and dynamic path planning conditions, being suitable for real-time implementation.

  An implementation of Probabilistic Roadmaps by Kavraki[15] demonstrates the division of the algorithm into two phases, the learning phase where the collision-free roadmap is created and the query phase where the best path is chosen.

  An example of a Rapidly-exploring Random Tree method from 2011 by Shen[16] details the incremental construction of a tree through samples spread in the space.

- **Node-based methods:** Similarly to the Sampling-based methods, the Node-based methods generate a path through several nodes, however, this nodes are created through the previous obtainment of sensory data. The list of created algorithms based upon this approach is fairly extensive with examples as A*, Lifelong Planning A* (LPA), Theta*, Lazy Theta*, Dynamic A* (D*), D* Lite and Harmony Search, most of which can also be applied to the output of Sampling-based methods.

  The A* algorithm, first described by Dijkstra[17], is responsible for computing the fastest path, with the lowest cost, to the target node, through the node graph in the implementation of Musliman[18].

  An implementation of the Theta* algorithm by Filippis[19] takes into account the vehicle attitude to plan its trajectory in a 3D environment.

  A simpler implementation by Grzonka[20] generates a 2D trajectory for the vehicle, using the D* Lite algorithm. This implementation allow the generation of a posterior 2.5D trajectory with the inclusion of the Yaw component.

- **Mathematical model methods:** This approach uses the models of the vehicle and environment, with the kinematic and dynamic limitations, to compute the cost function that will identify the optimal path for the vehicle. This approach is commonly used offline due to its high demand for computational power. Some classes of problems included into this category are Mixed-Integer Linear Programming (MILP), Binary Linear Programming, Nonlinear Programming and Extended Kalman Filter (EKF).

  An example of an EKF implementation combined with Simultaneous Localisation and Mapping (SLAM), by Huh[21], demonstrates how the 3D point cloud is generated from the sensory data and the how the EKF filter estimates the vehicle and landmarks states.

- **Bio-inspired methods:** These methods replicate the biological behaviour without the construction of complex environment models and try to converge to the solution, this procedure however, requires a high computational power.

  The bio-inspired methods can be divided into two subcategories, Neural Networks that generate a dynamic landscape for the neural activities and Evolutionary Algorithms (EA) that select randomly feasible solutions as the first generation, the environment, robot's capacity, goal and other constraints are taken into consideration when planning the next step, the process stops when a pre-set value is achieved. Some examples of Evolutionary Algorithms are generic algorithm, memethic algorithm, particle swarm optimisation, ant colony optimisation and shuffled frog leaping algorithm.

  The memethic algorithm is used in [22] by Shahidi to obtain a optimal path free of collisions, using a small population and in a few generations.

- **Multi-fusion based methods:** To plan an optimal trajectory in a 3D environment, these methods combine several different simpler algorithms. These methods are also divided into two subcategories, Integration of Algorithms that integrate several path planning algorithms to achieve an optimal trajectory and Algorithms Ranking that also use several path planning algorithms to achieve an optimal trajectory but in a consecutive way.

  An interesting implementation by Masehian[23] integrates a visibility graph with a Voronoi diagram and a potential field to achieve the shortest and safest path.

There is two specific implementations of area coverage that are worth mentioning. The first by Maza et al.[24], tackles the adversities of a cooperative action between multiple UAVs for exploration of an area, this goal is achieved with the polygon area decomposition and with efficient coverage algorithms that rely on zigzag patterns. The second implementation by Schwager et al.[25], presents a decentralised control strategy for positioning and orienting multiple robotic cameras to collectively monitor an environment, this approach is capable of handling adjustments on the number of vehicles and vehicles with different degrees of mobility.

2.1.2 ASVs Path Planning Methods

After the aerial and underwater vehicles contribution to the scientific world, the interest in Autonomous Surface Vehicles (ASVs) has grown considerably. Although ASVs present a high potential in maritime applications, their ability to detect and avoid obstacles still lacks development, being typically focused on above-surface obstacles while neglecting the need for the detection of sub-surface obstacles such as waters with reefs or lakes with shallow banks[26].

Although most of the 2D path planning techniques described before could be adapted to ASVs, the most noticeable development in this field was the integration of the International Regulations for Avoiding Collisions at Sea (COLREGs)[27], within the obstacle avoidance algorithms, reducing the ocean navigation conflicts and maritime accidents attributed to human error, while simultaneously, establishing legal policies for unmanned vessels[28].

The modern COLREGs were delineated in 1972 by the International Maritime Organisation as a set of rules for potential collision scenarios, as crossing paths, head-on and overtaking, in a maritime environment. Although these rules describe possible manoeuvres to avoid collisions, their creation was designed for human navigators usage, being dependent of the operator experience and interpretation. It is estimated that the subjective nature of COLREGs and human error are related to 89% to 96% of marine collisions. Therefore, if autonomous surface vehicles can operate in accordance with these rules, maritime collisions can be vastly reduced.

In 2012, Naeem et al.[29] proposed an approach that relies on a simple waypoint by line-of-sight guidance strategy, coupled with a manual biasing scheme. The vehicle follows the direct route through the multiple waypoints defined between the initial and goal positions, when no obstacles are found on its path. When an obstacle is detected by the onboard vision-based detection system, a bias is added to the current reference heading angle, in order to avoid an obstacle. The added bias is compliant with the COLREGs regulations. After the obstacle is overpassed, the heading angle, between the vehicle current position and the next waypoint, is computed once more.

Another implementation by Campbell et al.[30], uses an obstacle detection system based on vision-LIDAR (light detection and radar), accesses the risk for the vehicle and with an heuristic path planner, avoids obstacles in a COLREGs regulations compliant way.

A worth mentioning implementation of COLREGs compliant path planning for ASVs was recently developed by Hu et al.[31], this approach, in addition to the COLREGs compliant obstacle avoidance algorithm and the risk assessment, enforces a multiobjective optimisation based on particle swarm optimisation re-planning the original path.

Other examples of navigation and obstacle avoidance in accordance with the Coast Guard Collision Regulations, for Autonomous Surface Vehicles, by Benjamin et al in 2006 and by Pinto et. al in 2013, can be found in [32] and [33], respectively.

2.2 Oil Spill Detection

The detection of an oil spill and its simultaneous mapping has been subject of multiple distinct implementations in the past years and can occur through several different sensors. The interest in this field [34] comes from the urgency in reducing the impact of such incidents in the environment. Data relative to the spill such as the precise location and movement, support an effective action in order to reduce the impact of the pollution in the environment.

As stated before, several sensors can be used to achieve oil spill surveillance and mapping, varying from visible spectrum, infrared and thermographic cameras to fluorosensor lasers and radars, carried by aircrafts, UAVs or on satellites. In this section, techniques with each sensor will be presented and analysed not with goal of being implemented during the dissertation but in order to understand the vehicle behaviour during the oil spill intervention.

2.2.1 Visible Spectrum Camera

The oil has an interesting aspect, it shows higher reflectance than water while not presenting a specific light absorption/reflection behaviour, this means that for oil identification, one has rely on contrast differences from the oil to the enclosing waters within the visible region of the electromagnetic spectrum (400 to 700nm approximately).

An example for a visible spectrum camera application can be found in [35], where a push-broom scanner method is used. This method is described as the use of a CCD (Charge-Coupled Device) detector and an optical system to direct ground elements to different parts of the CCD detector.

Oil spill detection with visible spectrum cameras has however several disadvantages, being the most noticeable the harsh weather conditions and the action of the light or its absence in the environment. The effect of this last can however be reduced since its known that the light, when it its the water surface at 53 degrees (Brewster angle) becomes polarised in parallel with the water surface, so a detector can be set for this specific angle and polarised lens can be used.

In the last years different approaches have been developed, one of them is the exploration of hyperspectral imaging in this scenario, that like other spectral imaging, collects and processes information from across the electromagnetic spectrum. Although hyperspectral imaging requires the computation of extensive data, not ideal for a real-time propose, a technique called spectral un-mixing is commonly used for characterising individual pixels, that can be used for oil monitoring[34].

Even if visible spectrum cameras present worse results on oil spill detection than termographic cameras for example, due to the sun glint in the ocean surface that can be falsely identified as an oil spill, they will continue to be widely used for support operations due to its economic aspect.

2.2.2 Infrared Camera

The oil spill, when exposed to a heat source, as the sun, will emit infrared radiation and since the emissivity from the oil is distinct from the water is possible to detect a distinct substance in the water. This behaviour of emission of radiation as thermal energy is not completely understood since thick oil spills appear hot and intermediate oil thickness appears cool. While not completely understood, this transition seems to occur between the thickness of 50 and 150 µm, furthermore the minimum detectable layer for an oil slick is between 10 and 70 µm, making thin oil spills undetectable with infrared cameras.

The largest portion of infrared sensing of oil spills occur in the thermal infrared, at wavelenghts of 8-14 µm. During daylight, thick spills appear warmer than the surrounding water since they absorb solar radiation faster, while thin films tend to appear cooler than oil-free water. This lower temperature can be correlated with the electromagnetic interference on the oil layer. During the night-time the behaviour is completely inverted with thin spills appearing warmer than the surrounding water and thicker spills acting as thermal insulators thus appearing cooler[36]. This contrast is supposedly higher during daylight as stated by Dickins[37].

Hover and Plourde[38] evaluated the day and night imaging capabilities of ship-mounted thermographic sensors operating in 8 to 15 µm range, as well as hand-held sensors exploiting the 3 to 5 µm interval and found both types of systems useful in the identification of oil slicks, although the performance of individual sensors depended on environmental conditions and sensor tuning.

Even though infrared cameras present several disadvantages as not being capable of providing thickness measurements and the possibility of false detections from seaweeds, sediment, organic matter, shoreline, and oceanic fronts, they are still widely used for oil detection due its the availability of the market at a reduced cost.

2.2.3 Fluorosensor Laser

Fluorosensor lasers take advantage of the presence of aromatic compounds in petroleum products, these compounds absorb ultraviolet light and become electronically excited, this excitation is transformed into a fluorescence emission, primarily in the visible region of the spectrum, ranging from 400 to 650 nm, with peak centres in the 480 nm region. Since other natural fluorescing substances emit at sufficiently different wavelengths than oil (e. g. chlorophyll yields a sharp peak at 685 nm), in most cases is easy to identify, with high certainty, if oil is present in the environment.

Some lasers increase even further its sensitivity and selectivity using a technique called "gating", opening their detectors at the precise time that the signals return from the surface. This feature can be taken even further to target specific regions below the surface on the water column as deep as 2 meters[39]. As a sampling instrument, the laser repetition rate and the velocity of the vehicle carrying the sensor are important in the sampling rate of the surface where the oil contamination is being observed. At ground speeds of 100–140 knots, at a laser repetition rate of 100 Hz, a fluorescence spectrum is collected approximately every 60 cm along the flight path. This decreases if the instrument is scanning.

The capability to identify oil presence in water, shoreline, soil, plants, ice and snow[40] is proving the enormous potential of fluorosensor lasers in an oil spill identification scenario.

2.2.4 Radar

Radar sensors became the standard sensor for mapping oil offshore, either carried by aircrafts or as a satellite sensor. Its oil detecting capability comes from the attenuation of the capillary waves, resulting in a "dark" region within the "sea clutter", formed by the microwaves reflection in the oil-free water[41]. For a correct perception, low to moderate wave/wind conditions are necessary, bellow 1.5 m/s wind speeds the "sea clutter" formed is insufficient for oil spill detection and wind speeds stronger than 10 m/s difficult the visibility of wave troughs.

Two distinct types of radar can be employed, Synthetic Aperture Radar (SAR) that uses the forward motion of the vehicle to simulate a long antenna, trying to replicate the other type of radar, the Side-Looking Airborne Radar (SLAR), to obtain satisfactory spatial resolution, independent of the range but requiring sophisticated electronic processing, making this solution more expensive than SLAR. Even though studies show a far superior performance by SAR[42], SLAR is widely used on oil spill remote sensing, mainly due to its lower price.

Apart from the wind/waves speed limitations, this sensor presents still another important disadvantage, the possibility of false positives from fresh water slicks, calm areas, wave shadows behind structures or topographical features, shallow seaweed beds, biogenic oils and sea-life sperm. However, radars provide a very good solution for large-area, night-time, and foul weather detection work.

2.3 Oil Spill Mitigation strategies

Immediate and efficient responses to oil spills are of extreme importance to reduce the economic and ecological impacts of such incidents. The quality of the response is dependent on several factors such as, the water temperature, the locale and most importantly, on the specific type of oil spilled.

In the last years a wide range of oil spill countermeasures were developed, most of them can be inserted into one of the following categories[43]:

- **Mechanical methods** have the simplest concept, the use of booms (floating barriers) to contain the oil spreading, enclosing the spill into a well defined area, while using oil skimmers to remove the oil floating on the water surface. Mechanical methods are still widely used due to their simplicity, however they can only be enforced on calm waters;

- **Chemical methods** though more efficient than mechanical methods when properly applied, could endanger countless species. These methods transform the physico-chemical properties of the oil mainly through dispersants, mixture of emulsifiers and solvents that improves the separation of the particles, breaking the oil slick into small droplets. These small droplets however, can be dispersed into the water column and when in high concentration have an acute lethal toxicity for many species, especially on fish eggs and coral;

- **Bioremediation** demonstrates enormous potential for oil spill cleanups with the use of microorganisms such as Fusobacteria species or biological agents to colonise and degrade hydrocarbons present in the oil, though its practical use is still restricted;

- **Controlled oil burning** effectively reduces the amount of oil in the water *in situ*, at a low cost, however the viscous and dense residue may sink and extends air pollution;

- **Solidifying** the oil could be an option in oil spills. With the use of dry ice pellets and hydrophobic polymers the liquid oil is solidified into a floatable rubber-like material, that can be more conveniently collected and recycled.

- **Vacuum and centrifuge** the mixture of oil and water is another approach that obtains near pure oil as an end result. There is a debate either the resulting water should be returned to the sea or not, improving the efficiency of the process, this is not entirely accepted due to the amount of oil present in the resulting water.

Technologies that allow a safe, immediate, effective and eco-friendly operation of oil spill removal, *in situ*, have been under development in the recent years. A specific area that grow from that development was the application of autonomous vehicles for ocean exploration and conservation, with two important projects being developed in 2010, Seaswarm[9] and Protei[10]. The autonomous vehicles designed within the scope of these projects, reduce the operational time and protect the health and safety of the cleaning crew by working as an organised fleet or "swarm" of vehicles, that rely on ocean-skimming and oil removal techniques, this can be viewed as a limitation of both approaches since there is a considerable impact of oil in the mechanical parts.

This evolution on oil spill mitigation techniques has primarily occurred in the form of hardware development. The research on advanced software-based navigation algorithms still lacks sufficient attention and support, however some cases are worth mentioning, like the approach developed by Jin and Ray[6], that uses a multi-resolution navigation algorithm that, seamlessly, integrates the concepts of local navigation and global navigation, based on the sensory information for oil spill cleaning in dynamic and uncertain environments, using autonomous vehicles as a single agent. The developed algorithm provides a complete coverage of the search area for clean-up of the oil spills and does not suffer from the problem of having "*local minima*", which is commonly encountered in potential-field-based methods through adaptive decision making and online re-planning of vehicle paths.

Chapter 3

Fundamentals

3.1 Path planning techniques

Considering the requirement of the ASV being able to navigate on the stroke border areas of the oil spill, the path planning methods explored for this vehicle were focused on the ability to have a cost function capable of cover all area but at the same time avoid, in a robust manner, the oil spill. Therefore, one of the methods presented in this chapter is the Potential Functions (or Potential Fields), applied in [44][45] and [46], where several gradient vector fields are created, attracting the vehicle towards the target or repelling it from the obstacles. The method generates repelling vectors once the vehicle is within a range of a new obstacle, and the sum of all vectors provides the direction for the movement of the vehicle. This method is computationally efficient although in some scenarios it has a limitation, the existence of a critical point called "*Local Minima*", where the vehicle can be attracted to it and will not be able to generate an escape safe position. In [47], obstacle avoidance using potential fields is applied into a non-holonomic vehicle, with two independently driven wheels and is not capable of sideways movement, resulting in the attractive and repulsive forces being applied on the front and on the rear body of the vehicle, being the repulsive forces computed by the distance between obstacle points and the contour of the vehicle's body, those repulsive and attractive forces constitute the resultant force that determines the motion of the vehicle.

Another approach proposed by Lumelsky and Stepanov [48], is based on curvature estimation. This real-time path-planning algorithms rely on sensor-based exploration [49], the boundary curvature of the obstacles are followed by the vehicle until a condition is reached, through this boundary curve following, the vehicle is able to reach the target.

Similar approaches were presented in [50], [51] and [52]. Zhang and et.al[53] propose a novel curvature-based steering control law able to produce the obstacle avoidance behaviour for unicycle type robots travelling at a constant speed.

To obtain a efficient and complete coverage manoeuvre of the oil spill, Voronoi algorithms were studied to be applied into the UAV path planning algorithm. An interesting approach by Cortés et al.[54] relies on Voronoi partition of the environment to control and coordinate a sensor coverage action between a group of autonomous vehicles, where each autonomous vehicle implements a control law designed based on the gradient descent method that minimises the coverage cost in space and time, leading to an optimal partitioning. The implementation of Abbasi et al.[55] also presents the control algorithm of an heterogeneous group of robots to achieve coverage over an area resorting to Voronoi diagrams, while ensuring an ideal allocation of robots to distinct regions of interest. Similar Voronoi approaches were presented in [56] and [57].

The path planning algorithm for the UAV was also inspired on the approach by Schwager et al.[25] that presents a decentralised control strategy capable of achieving area coverage (environment monitoring), with the fields of view of multiple cameras on distinct aerial platforms, resorting to a cost function that, unlike most approaches, does not involve a Voronoi partition.

3.1.1 Potential Fields

The Artificial Potential Field (APF) method was first proposed by Khatib [58] and can be simply described as "magnetic fields", that are simulated for the goal position and for any obstacle present in the scenario. The goal position attracts the the robot, while, simultaneously, the robot is repelled from any obstacles in its path. In theory then, with correctly designed potential fields, the robot will follow the shortest trajectory to the goal position while avoiding getting too close to an obstacle.

The overall potential field, at any moment, in the robot q, $U(q)$, in the Equation 3.1, can be described as a sum of the attraction force towards the goal U_{goal} and the sum of every repulsive force of obstacles in the vicinity $\sum U_{obstacle}$.

$$U(q) = U_{goal}(q) + \sum U_{obstacle}(q) \tag{3.1}$$

The attractive force towards the goal, represented in the Figure 3.1, is given by the Equation 3.2, where $\Delta d(q, goal)$ represents the euclidean distance between the goal position and the robot q.

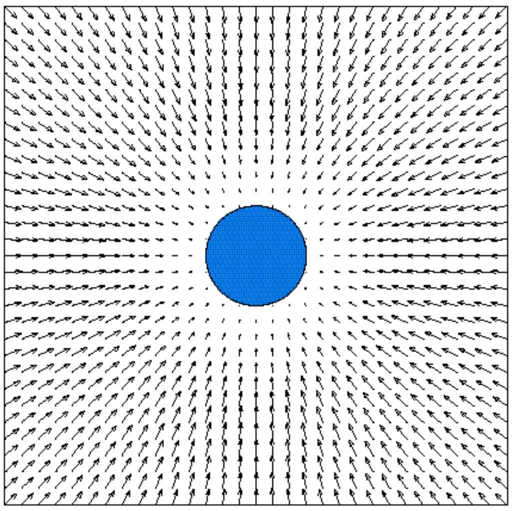


Figure 3.1: Representation of the attractive forces towards the goal.[1]

$$U_{goal}(q) = \Delta d(q, goal)^2 \tag{3.2}$$

The potential barrier imposed by each individual obstacle, represented in the Figure 3.2, is given by the Equation 3.3, where $\Delta d(q, obstacle)$ represents the euclidean distance between a specific obstacle and the robot q. This potential barrier rises to infinity when the robot approaches the obstacle.

$$U_{obstacle}(q) = \Delta d(q, obstacle)^{-1} \tag{3.3}$$

If all the potential fields from the goal and from the obstacles are combined, it's possible to obtain the overall potential field represented in the Figure 3.3. From here is possible to compute the most favourable path for the robot to reach the goal position, Figure 3.4.

This approach presents, however a disadvantage, in certain scenarios, as in a "U" shaped obstacle in the vehicle path, represented in the Figure 3.5, if the vehicle is attracted to the "inner" area of the obstacle, it will never be able to get out of it and reach the goal position, this places are called "*Local Minima*".

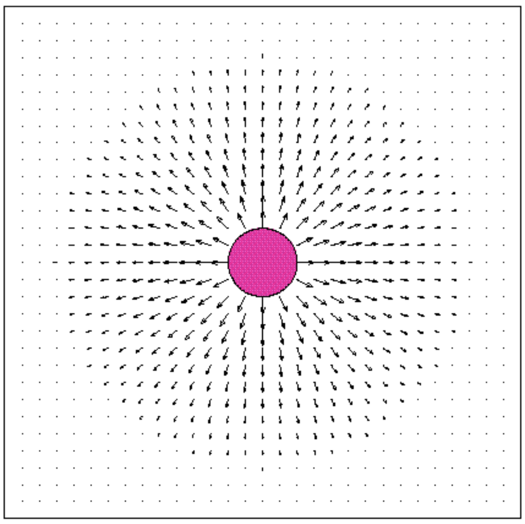


Figure 3.2: Representation of the repulsive forces imposed by each obstacle.

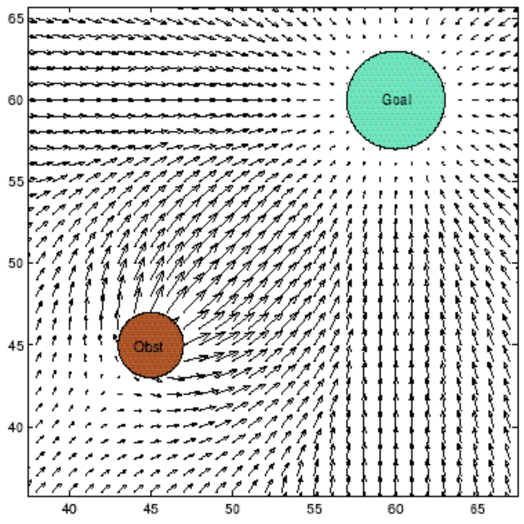


Figure 3.3: Representation of the overall potential field.

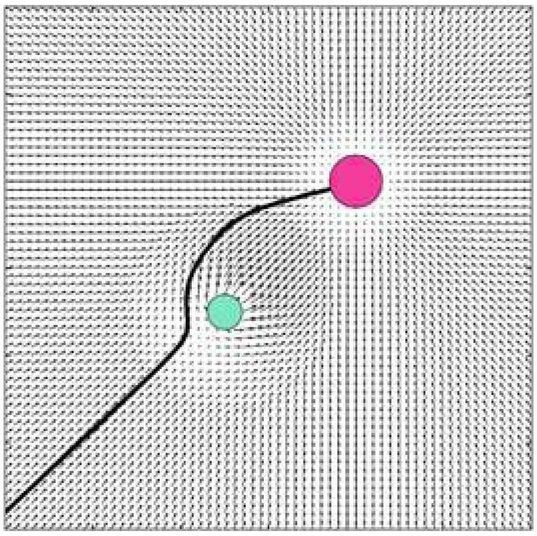


Figure 3.4: Representation of the most favourable trajectory for the vehicle to reach the goal position.

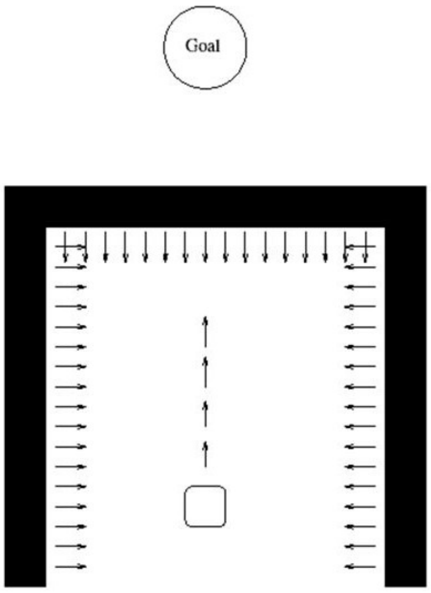


Figure 3.5: Representation of a scenario that contains a *Local Minima*.

3.1.2 Probabilistic Roadmap

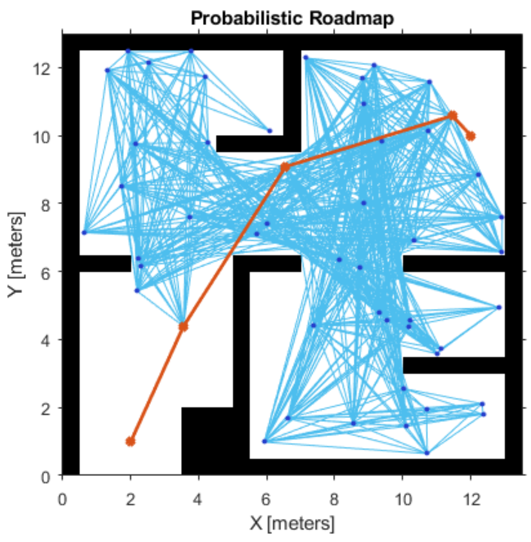
Probabilistic Roadmap (PRM)[15][59] is a path planning algorithm that computes a collision-free trajectory, from the starting configuration of the robot to a goal configuration. It receives as inputs the initial and goal position for the robot and the position of every obstacle in the map. A defined maximum number of random samples are distributed within the map, only in the space free of obstacles. To those samples is added also the initial and target positions. From there, each sample tries to "connect itself" to a defined maximum number of closest neighbour samples, by straight segments that do not escape the configuration space, creating the roadmap, if any sample is not connected in this roadmap, the number maximum of closest neighbours is increased until they're all connected.

After the roadmap is built, the shortest way from the initial position to the goal position, through the samples, is computed. For this demanding computational step several algorithms can be used, being the A* (A star) algorithm[60], first described Dijkstra[17], the most frequent. This approach uses all the possible paths to the target, to determine which one has the smallest cost (smallest distance).

The number of random samples added to the configuration space of the robot in the beginning, is directly linked to the resultant trajectory from the initial position to the goal position, when the number of samples tends to infinite it's possible to obtain the shortest resulting trajectory. In the Figure 3.6 it's possible to see the difference in the resulting trajectory, within the same map in having increased the number of random samples.

PRM has been applied with excellent results to free flying and articulated robots moving in the plane or in space, as well as to non-holonomic robots.

[2]https://www.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html



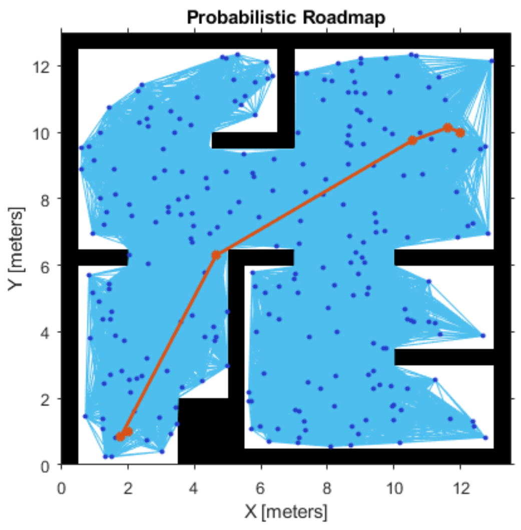


Figure 3.6: Comparison between the probabilistic roadmaps (in blue) and resulting trajectories (in orange) in the same map with different number of random samples, 50 (left) and 250 (right).[2]

3.1.3 Voronoi

Voronoi Diagrams are one of the most fundamental data structures in computational geometry. In [61], [62] and [63] a better description of the algorithm, its application and importance in a wide variety of fields inside and outside of outside computer science, are presented. Given a set P, of points, called sites in this specific algorithm to differentiate them from arbitrary points, a Voronoi diagram is simply the subdivision of a space into cells, with one cell per each site $p \epsilon P$.

$$Vor(p, P) = \{x : |px| \leq |qx|, \quad \forall q \quad \epsilon \quad P\} \tag{3.4}$$

A point q lies in the cell corresponding to a site $pi \epsilon P$ if:

$Euclidean\_Distance(q, pi) < Euclidean\_distance(q, pj)$, for each $pi \epsilon P$, $j \neq i$.

In the Figure 3.7, we can see the terms used to describe Voronoi diagrams.

The division between two cells, in a Voronoi diagram, is made by a line segment, of infinite length, unless other conditions are imposed. This line segment is Euclid's compass and straight-edge construction of the perpendicular bisector, Figure 3.8.

A point q lies on a Voronoi edge between sites pi and pj if the largest empty circle centred at q touches only pi and pj – A Voronoi edge is a subset of locus of points equidistant from pi and pj.

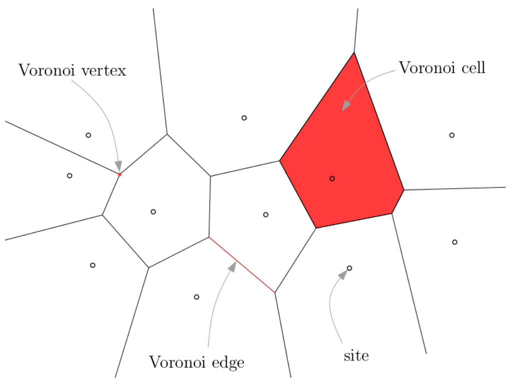


Figure 3.7: Terms used in Voronoi Diagrams.

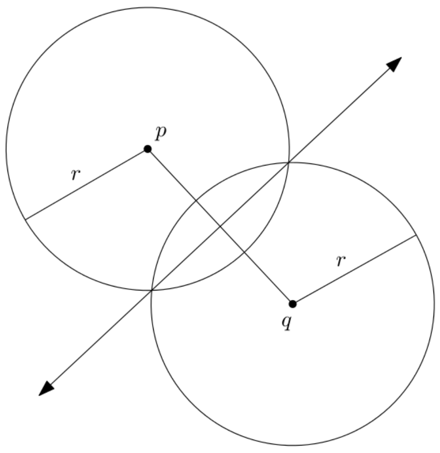


Figure 3.8: Perpendicular bisector representation.

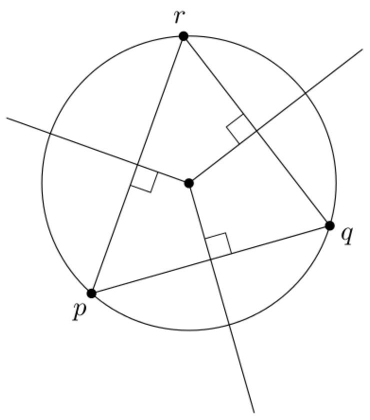


Figure 3.9: Circumcenter representation.

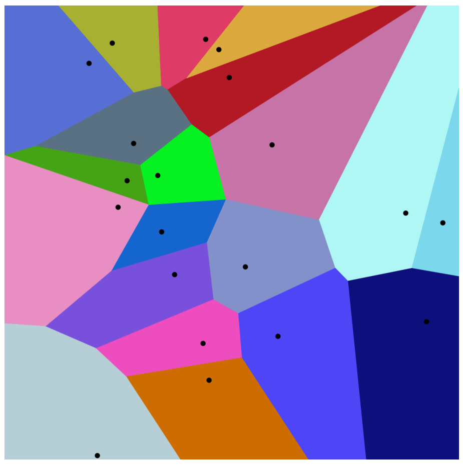


Figure 3.10: Graphical representation of a Voronoi Diagram.

Multiple collinear sites form a series of parallel lines while three non-collinear sites form Voronoi half lines that coincide in a Voronoi vertex. This vertex is the circumcenter of the triangle formed by the three sites, the only point that is at an equal distance from all vertices of the triangle, Figure 3.9.

A point q is a vertex if the largest empty circle centred at q touches at least 3 sites – A Voronoi vertex is an intersection of 3 more segments, each equidistant from a pair of sites.

In the figure 3.10, it's presented an example of a Voronoi diagram, with a set of points P, and the cell division computed using Euclidean distances.

3.2 ROS - Robot Operating System

Developed in 2007 by the U.S. robot company Willow Garage, ROS[64] stands for Robot Operating System but, in fact, ROS is not a real operating system. It can be described as a middleware since it stands between the aplication and the operating system on each individual machine it is capable of message-passing between processes, and package management with various ROS libraries open-source with implementations of common robotics functionality and algorithms, focused on maximising code reuse in the robotics research and development. Several implementations are distributed as ROS packages with each ROS distribution or through code sharing sites such as GitHub[3].

The representation of the processes of this middleware can be done in a graph architecture with several independent processes, called nodes, these nodes communicate between themselves through a publisher-subscriber mechanism sending data streams, known as topics. This peer-to-peer communication is controlled by a master node, created obligatorily using the *roscore* command when starting. The various existent nodes could be running on the same or on a different machine, being responsible for subscribe, process and publish data, control actuators between other operations.

To clarify the publish/subscribe messaging mechanism, depicted in Figure 3.11 an example of a possible communication is presented. In this example on the robot, the Camera Node takes care of the communication with the camera, another node on the robot, the Image Processing Node processes the image data and a Image Display Node, on a external laptop, displays images on a screen. To start every nodes registers itself with the ROS Master. In registering with the ROS Master, the Camera Node states that it will publish a topic called "*/image_data*". Both of the other nodes register that they wish to subscribe to the topic "*/image_data*" thus, once the Camera Node receives some data from the Camera, it sends the topic directly to the other two nodes. It is also possible for a node to request a message at a specific time, registering a service with the ROS Master.

It's possible to integrate ROS with real-time code to mitigate the flaw in this middleware of not being an real-time system. This capability is of major importance in a large number of applications were low latency in robot control is a critical aspect.

ROS provides a platform to develop a distributed and highly independent modular system to control a robot. The constituent modules i.e. nodes have limited knowledge of other nodes in system and communicate over TCP (or UDP) via standard messages i.e. topics. This is a great strength and enables rapid and clean development and high

[3]www.github.com

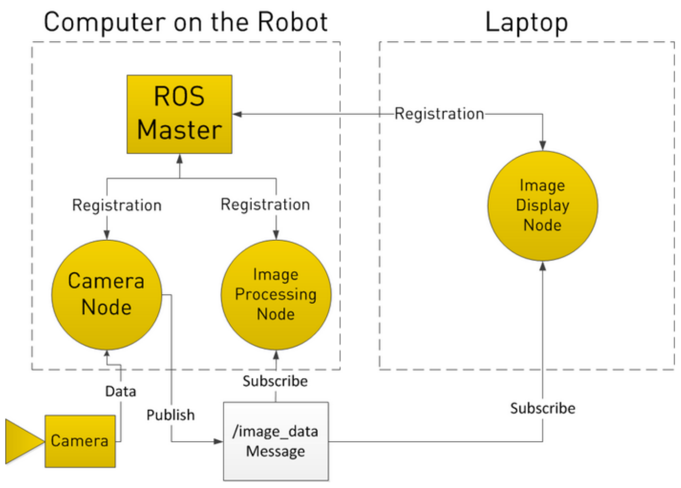


Figure 3.11: Example of the publish/subscribe messaging model.[2]

reusability. This does come, however, with a performance drawback as communication must be done over network infrastructure which can slow down things considerably especially if the message consists of huge sized data like video streams or lidar scan, to overcome this efficiency issue and still provide the same benefits of standardised message passing infrastructure, ROS presents the concept of nodelets. Nodelets can use the same interface of subscribing and publishing to topics, however, when a nodelet subscribes or publishes to a topic in the same nodelet manager, instead of message being passed over TCP/IP, only a C++ boost pointer to message is passed.

The software is open-source and free for both commercial and research use. Its open-source packages vary from hardware drivers, robot models, data-types, planning, perception, simultaneous localisation and mapping, simulation tools, and other algorithms that can be built with different programming languages like C++, Python, Octave and LISP.

As shown in Figure 3.12, ROS consists of a client library to support various programming languages, a hardware interface for hardware control, communication for data transmission and reception, the Robotics Application Framework to help create various Robotics Applications, the Robotics Application which is a service application based on the Robotics Application Framework, Simulation tools which can control the robot in a virtual space, and Software Development Tools.

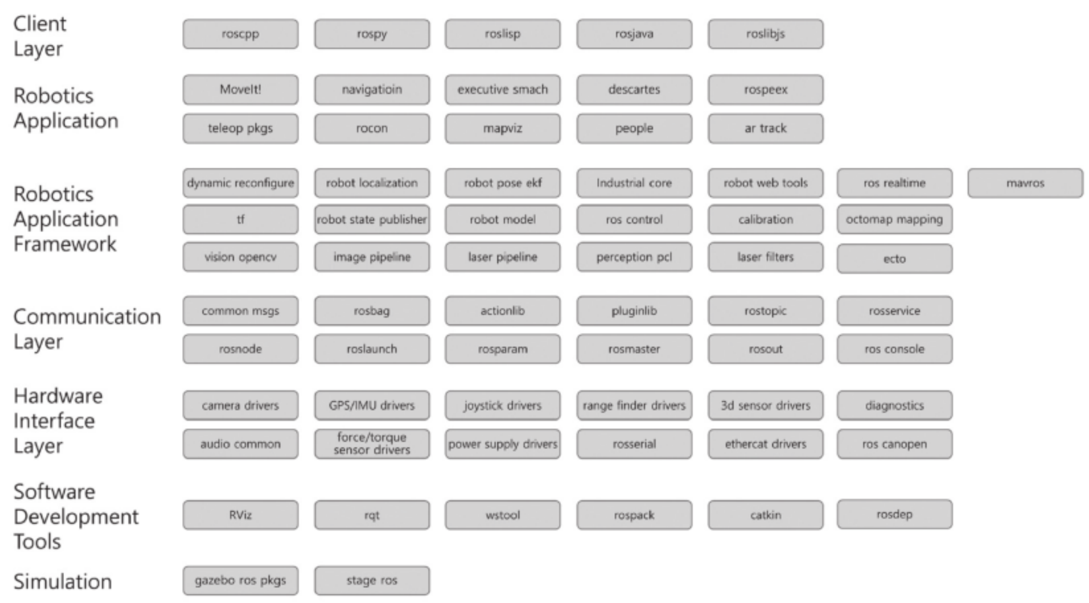


Figure 3.12: ROS components[3]

3.3 Robotic Simulators

Experiments with unmanned vehicles are complex, costly, time-consuming and in some circumstances potentially dangerous, involving the risk of losing or damaging the robots, so in most cases, the most promising course of action is the simulation of the situation. Simulators are useful tools for the development of unmanned vehicle software, algorithm benchmarking and system preliminary validation, that may later, improve the performance of the robots, in real life experiments.

Many simulation tools[65][66][67][68][69] have been used and specifically developed for robotics development purposes. The inclusion of proper dynamics, visual realism, wide variety of sensors, interfaces, modelling tools and real-time sensor-based control are major factors in a simulator.

From 2D low fidelity simulators such as Player/Stage[70] to 3D dynamic simulators such Gazebo[69], MORSE[65], USARSim[71], UWSim[72] or V-REP[73], these systems allow for simulation of mobile robots and their interactions with the environments. Some simulators, such as Gazebo for instance, have hardware in loop capabilities allowing for multiple stages of subsystem validation with real hardware.

Marine robotics[74][75][76][77], focused simulators have been developed either for specific simulation scenario requirements or as more or less generic tools under multiple European research projects such as Co3-AUVs[78], RAUVI[79] or TRIDENT[80] with

UWSim[72].

Nowadays, numerous simulators are available in the market, this research will be focused on simulators compatible with ROS such as Gazebo, UWSim and MORSE. Apart from the ROS compatibility, the software's are built on modern rendering and physics engines and have large support communities. All of the listed simulators are open source. As described previously, ROS is a distributed system where different nodes can run on different computers and mainly communicate through "topics", via publishing/subscribing mechanisms. For instance, the ROS interface of UWSim, allows running the simulator as another ROS node that can communicate with the rest of the architecture with the standard ROS communication facilities.

Robotics simulations have been presented in many projects and conferences, some of the most relevant are DARPA Robotics Challenge (DRC), ICRA, RoboCup and more recently in Space Robotics Challenge (SRC), a NASA Centennial Challenge. Earlier in 2007, during the International Conference on Robotics and Automation (ICRA'07) Robin Murphy, presented a survey[66], were the state of the art on robotic simulators, a subject under an early development, was analysed. In that survey a set of available and open source simulators capable of handling a wide spectrum of vehicles, such as unmanned terrestrial, aerial, surface and subsurface vehicles, were explored under the most diverse scenarios and situations such as urban search and rescue, bomb disposal, surveillance and military purposes. This study ended up with the conclusion that there was no need to build a robotic simulator from the ground since there was already available, in the market, multiple solutions with reasonable physical and functional fidelity. Nowadays, robotic simulation has evolved and numerous simulators are available in the market, this softwares are built upon modern rendering and physics engines and have large support communities. In 2014, Cook, Vardy and Lewis et al. [81] reviewed and compared multiple robot simulators for multi-vehicle operations.

In order to obtain a better comparison between them, different criteria were defined:

- **Physical Fidelity:** As the name suggests, this field allow a deeper comparison between the softwares, on their physical fidelity, being this, the capability for a correct interaction between the robot and the environment/objects, between the robot and its arms/actuators or between those actuators and the environment/objects. Simple actions such as pushing, picking or grasping objects involve a complex calculation of simulated forces and collisions.

- **Sensor Modelling:** This criteria defines the capability of the software in the precise simulation of multiple sensors.

- **Required Knowledge/Experience:** The amount of knowledge/experience required to work with the simulator software in a proficient way.
- **Visual Fidelity:** Quantifies the fidelity of the visualisation on the simulator, where details such as water reflection, water refraction, sediments flotation, waves movement and clouds representation are taken into account.

This and other criteria are summarised in Table 3.1.

Table 3.1: Simulator Comparison

Simulator	Gazebo	UWSim	MORSE
3D Rendering Engine	ogre3D	OSG	Blender
Physics Engine	ODE/Bullet/Simbody/DART	Bullet	Bullet
Programming Language	C++	C++	Python
Middleware Support	ROS/Player/Sockets	ROS	ROS/YARP/Pocolibs/ MOOS/Sockets
Operating System	Linux/MacOS X/Windows	Linux	Linux/MacOS X
Formats Support	SDF/URDF	URDF	URDF
Open Source	Yes	Yes	Yes
Adequate Documentation	High	Low	Medium
Physical Fidelity	High	Medium	Medium
Sensor Modelling	High	High	Medium
Required Knowledge	Medium	Medium	High
Visual Fidelity	Medium	High	High

3.3.1 UWSim

In order to have a simulator for marine robotics research and development, UWSim[72][82] was developed in the scope of the RAUVI and TRIDENT research projects. UWSim is currently used in different ongoing projects funded by European Commission (MORPH [83] and PANDORA [84]) in order to perform hardware in the loop experiments and to reproduce real missions from the captured logs. UWSim is not only useful for software validation, but also for benchmarking mechanisms inside the simulator, so that control and vision algorithms can be easily compared in common scenarios.

This underwater and surface robotic simulator renders realistic images through OpenSceneGraph (OSG)[4] rendering engine, Bullet[5] physics engine and osgOcean[6] plugin. OSG is an open source 3D graphics application, while the plugin osgOcean was developed to enhance the reality of the underwater simulation in OSG, it adds visual improve-

[4]http://trac.openscenegraph.org/
[5]http://bulletphysics.org/
[6]https://github.com/kbale/osgocean/

ments such as waves, water coloration, reflection/refraction, flotation of sediments, etc. The toolkit is written in standard C++ using OpenGL[7] and runs on various operating systems including Microsoft Windows, Mac OS X, Linux, IRIX, Solaris, FreeBSD and recently also Android. The Bullet physics engine adds the physical reliability, by detecting any kind of collisions or even with physical interactions between multiple robots/objects such as push or grab.

The physics engine is used only to handle contact forces and the implementation of the vehicle dynamics, including the simulation of thrust forces. It is located in one monolithic ROS node, but it could be modified to adhere to a more modular structure. UWSim possess also an interface to communicate with external software such as Matlab, using ROS nodes.

A vehicle in UWSim is composed of a 3D model, created by user, that can be positioned in the scene by setting 6 degrees of freedom. Support for kinematic chains are included. The robots are described with an XML file, according to the URDF format, that can include kinematic, dynamic and visual information. Interfacing with Matlab is also possible, through the *ipc_bridge* ROS package.

The model can be complemented with other dynamic models like those corresponding to the marine environment (hydro-dynamics, waves, wind, underwater currents, etc.), to the actuators (thrusters and control surfaces as rudders or fins), and to the sensors (sonar, DVL, etc.).

UWSim allows the dynamic simulation of rigid body motion, by using a state-space dynamic model in terms of state variables representing body linear and angular velocities and positions. It takes as inputs the forces and torques that act on the body and their current state vector value. The output is a future estimation of the state vector. Customizable widgets can be added to the main window that show specific data to the user. The data acquired during a survey mission, with a real robot, can be logged and then reproduced in UWSim in order to analyse the vehicle trajectory. Vehicle dynamics can also be simulated with Matlab.

In this simulator it is also possible to visualise different underwater virtual scenarios that can be configured using standard 3D modelling software (ex: Blender[8], 3D Studio Max, etc). Controllable underwater vehicles, as well as simulated sensors can be added to the scene and accessed externally through network interfaces. This allows to easily integrate the visualisation tool with existing control architectures. The scenes in UWSim are XML-formatted documents that describes the general scenario, and simula-

[7]https://www.opengl.org/
[8]https://www.blender.org/

tion parameters. On the other and, robots are described with an URDF (Unified Robotic Description File) file. However, a scene XML file may make a reference to an URDF file for including a robot into the scene. The UWSim scene XML file is divided in blocks, which define the different aspects of the scene. The available blocks are the *oceanState* block that allows configuring ocean parameters, *simParams* block, that makes possible to modify the settings of the simulator, the camera block for set the main camera parameters, the vehicle tag that is used to create and configure underwater robots and the sensors available on them, the object block that allows inclusion of other 3D models to interact with the robots and the ROS interfaces block that allows the attachment of ROS interfaces to certain objects, robots or sensors, specifying the communication possibilities. The supported 3D models formats are all that are supported by OSG, like *.osg*, *.obj*, *.ive*, *.stl*, *.3ds* and others. So, it is possible to use a 3D modelling program such as Blender to simply export the 3D model with one of the formats above.

All the different robots sensors and actuators can be interfaced with external software through the network. UWSim includes an interface for its integration with ROS, that is a set of libraries and tools that assist software developers create robotic applications.

This allows to seamlessly validate control methods developed in ROS either on UWSim or on the real robots, as long as they provide the same interface. Through the ROS interfaces, it is possible to access/update any vehicle position or velocity, to move arm joints, and to access the data generated by virtual sensors. This provides to the software the capability to detect collisions and forces and automatically updating the scene accordingly. The different bodies collision shapes can be automatically generated from the 3D models. In addition, it is possible to set the position and attitude of collision shapes automatically from the scene graph, which is necessary, for instance, for updating the collision shapes transforms of kinematic chains like manipulators.

The output variables (position and attitude of vehicles) are published on the ROS network and captured by the UWSim core for updating the visualisation.

It is also possible to use UWSim for mission playback. For instance, the navigation data acquired during a survey mission with a real robot can be logged and then reproduced in UWSim in order to analyse the vehicle trajectory. If bathymetry and images of the seafloor are gathered during the survey, it would be possible to build a textured terrain from them and visualise it in UWSim.

There are twelve sensors available for vehicles in the current version of UWSim such as camera, range sensor, pressure sensor, DVL, IMU, Global Positioning System (GPS), Multibeam, force sensor and structured light projector.

UWSim is not a difficult software to work with, though it requires some level of previous experience using ROS. The UWSim wiki page contains articles on installing the software and on the configuration and creation of simulation scenes.

The major drawbacks in UWSim are that is only a kinematic simulator and even with its external dynamic module coded in Matlab it is only capable of handling single-body vehicles, excluding most of the situations, where a AUV carry a robotic arm and the aspect that there is no convenient way of extending the software, any modifications, e. g. adding a new type of sensor must be written in the core source code.

3.3.2 Gazebo

One of the most used softwares, is Gazebo, a primary tool used by ROS for simulations, which is very common among robotics professionals and academics. It was built with ROS compatibility since the beginning, and is easy to work with, however, it is also possible the use of another middleware wrapper for Gazebo, such as YARP plugin (Yet Another Robot Platform)[85]. This simulator uses Bullet for physics simulation.

Gazebo features high fidelity models and a enormous user base, a recent example of use is shown in [86]. Gazebo supports plugins, allowing users to embed custom C++ code into simulations. Communication method used by Gazebo, are topics that export simulated data to third party applications. Gazebo topics are flexible but, there is no data transfer control.

This simulation software uses the Ogre3D[9] rendering engine and, in opposition to UWSim, supports multiple physics engines, being the first to support four different physics engines such as ODE[10], Bullet, Simbody[11] and DART[12], being its default physics engine ODE. Having this in mind, its clear that, the physical fidelity of this simulator will be dependent on the physics engine chosen for the compilation. The simulation system is also able to simulate multi-robot collaboration.

Gazebo overcomes the drawbacks of UWSim, since it handles better the dynamics, contact physics and is very versatile through its plugin-based design.

As UWSim is only a kinematic simulator, the URDF used to describe the robots are usually simpler that the Gazebo ones, that contains all the inertial and collision-related data. On the opposite, the scene file, used to describe the whole world setup in UWSim, contains all information about the considered simulation while the same data is separated in several files when using Gazebo. The considered modelling is focused

[9]http://www.ogre3d.org/
[10]http://www.ode.org/
[11]https://github.com/simbody/simbody/
[12]https://dartsim.github.io/

on the main effects due to hydrodynamic forces, that are drag and buoyancy that are already available through plugins. A possible improvement is to take into account the added inertia and Coriolis, but these effects are even harder to quantify precisely.

Gazebo uses Simulation Description Format (SDF) and Universal Robot Description Format (URDF), to describe the simulation, the robot and its sensors, this data can be used in any other software that supports this format. A robot model in SDF is mainly a set of links and joints. The links represent the rigid parts of the robot, and the joints represent connections between two links and therefore define the kinematic and dynamic properties of the robot. In a world file, the simulation world is described, which are lighting, simulation step size, simulation frequency and other simulation properties. Robot, sensor or world models are described in their respective SDF file, an XML format designed for Gazebo. The URDF file format used by ROS is automatically converted to SDF format when used by Gazebo. Visual geometries used by the rendering engine are provided in COLLADA format and the collision geometries in *.stl* format. In Gazebo 8, *.obj* format was added as alternative input option for COLLADA (*.dae* format).

Gazebo's rendering system is not optimised for underwater scenario, where underwater characteristics are not taken into account. However, Gazebo simulator gives the user the possibility to extend the simulation with plugins. It can be extended for new dynamics, rendering, sensors and world models. Graphical user interface is included to visualise the scenario with extended capabilities. Regarding future work within this project, one of the most noticeable features missing in this underwater robotics simulator for the Gazebo environment is the lack of characteristic visual effects as floating particles and proper light damping as a function of water depth, along with the generation of waves on the sea surface.

As a drawback, Gazebo's rendering system has a far worse performance, where underwater characteristics such as water colour, visibility and floating particles are not represented. Since Gazebo 6, the software already includes a hydrodynamics package where the buoyancy and drag forces is already taken into account.

In 2014[74], Olivier Kermorgant developed *freefloating_gazebo*[13], a plug-in for the integration between Gazebo and UWSim, achieving a realistic simulation, where the dynamics from Gazebo and the appropriate visualisation from UWSim are combined.

In 2016, within the scope of the SWARMs project, a open-source package for Gazebo simulator was created, named UUV (Unmanned Underwater Vehicle) simulator[77] for underwater intervention and multi-robot simulation where a realistic ocean environment representation was achieved.

[13]https://github.com/freefloating-gazebo/freefloating_gazebo/

Another example of the integration between Gazebo and other softwares, is mentioned in [87]. This work consists in the integration between Rock-Gazebo, which provides a solution to simulate ROCK (Robot Construction Kit)[14] based systems in Gazebo. This solution synchronises framework components in the simulation. All components states are updated within the simulation step, thus synchronising the simulation. The synchronisation is important to define robots states and control the data transfer. It also includes Vizkit3d[15] visualisation plugins that make the simulator more flexible, allowing the user to include features, such as, an underwater environment or even a spatial environment.The underwater environment was released in the *simulation-gazebo_underwater* package. This package is parsed by a Gazebo world file and reads the simulation parameters like the model centre of buoyancy, water level, fluid density and drag coefficient.

Various sensors are already available in Gazebo such as camera, multi-camera (stereo), laser scanner, IMU, GPS, sonar, etc. There is also a provided application programming interface, for the creation of new sensors as plugins for Gazebo.

For an inexperienced user, Gazebo can be fairly easy understood and used, with the help of the large number of tutorials available in Gazebo's web-page and with a big community, though it also requires some previous experience with ROS for a deeper understanding. Gazebo has a regularly updated and well-defined road-map for new releases. The integration with ROS, also developed by OSRF[16], is already guaranteed through Gazebo/ROS packages. Contributions from the Gazebo user community allow it to be regularly improved with new features.

3.3.3 MORSE

MORSE[88] uses Blender Game Engine and the Bullet physics engine. Physical fidelity is completely dependent of the fidelity provided by the Bullet engine. Attached components such as arms and grippers, can interact with the world up until some degree, fine grasping of objects is generally not possible. MORSE can be entirely controlled from the command-line. Two configuration are provided by MORSE for time handling: best effort, that tries to keep a real-time simulation and fixed step that ensures accuracy of simulation. Multiple middlewares are compatible with MORSE, including ROS, YARP, Pocolibs[17], Mavlink[18] and MOOS[19]. In addition to those, MORSE also supports

[14]http://rock-robotics.org/
[15]http://rock-robotics.org/stable/documentation/graphical_user_interface/450_vizkit3d.html/
[16]https://www.openrobotics.org/
[17]https://github.com/openrobots/pocolibs/
[18]http://qgroundcontrol.org/mavlink/
[19]http://oceanai.mit.edu/moos-ivp/pmwiki/pmwiki.php/

a socket-based protocol that allows the integration of unsupported middlewares or tools. The most stable and extensible method of communicating data between the MORSE simulator and an external process, is with the use of ROS.

Like UWSim and Gazebo, MORSE already includes multiple sensors including accelerometer, battery sensor, contact sensor, depth camera, GPS, odometry sensor, laser scanner, etc. It's also provided a convenient facility for the creation of nonexistent sensors and actuators.

MORSE provides multiple tutorials divided by levels of proficiency, making the general use of the simulator by inexperienced users easier. The programmer can choose to either use Blender's internal integration engine for basic physics models, such as the differential drive robot, or the programmer can integrate complex models with the use of an external C++ program. One additional property of MORSE are modifiers. Those can modify data produced by sonar ("perfect data"), by adding additional noise functions, similarly to the Gazebo and UWSim simulators.

MORSE is designed to allow simulations of multiple robots systems. The biggest advantage of using Blender is the high customisation and the high level of graphical detail that can be achieved, thanks to the advanced modelling of meshes, and effects such as texturing, lighting and shades. Blender also offers the capability of using multiple camera views, displaying a global view of the scenario, as well as views from each of the cameras on–board the various robots. MORSE component consists of a Python and a Blender file. The Python file defines an object class for the component type, with its state variables, data and logical behaviour. The Blender file specifies the visual and physical properties of the object in the simulated world. Sensors and actuators have a reference to the robot they are attached to, and the relative position/orientation with respect to it.

MORSE have many advantages like high detail world, ability to create new sensor and cameras view, but at the same time, presents the drawback of having to understand its interface, as well as the additional computational overhead.

Chapter 4

Conceptual approach

This chapter will describe the conceptual approach formalised in this dissertation, starting with a brief description of the proposed architecture, followed by a detailed explanation on the UAV and ASV trajectory computation proposed methods.

4.1 Proposed Architecture

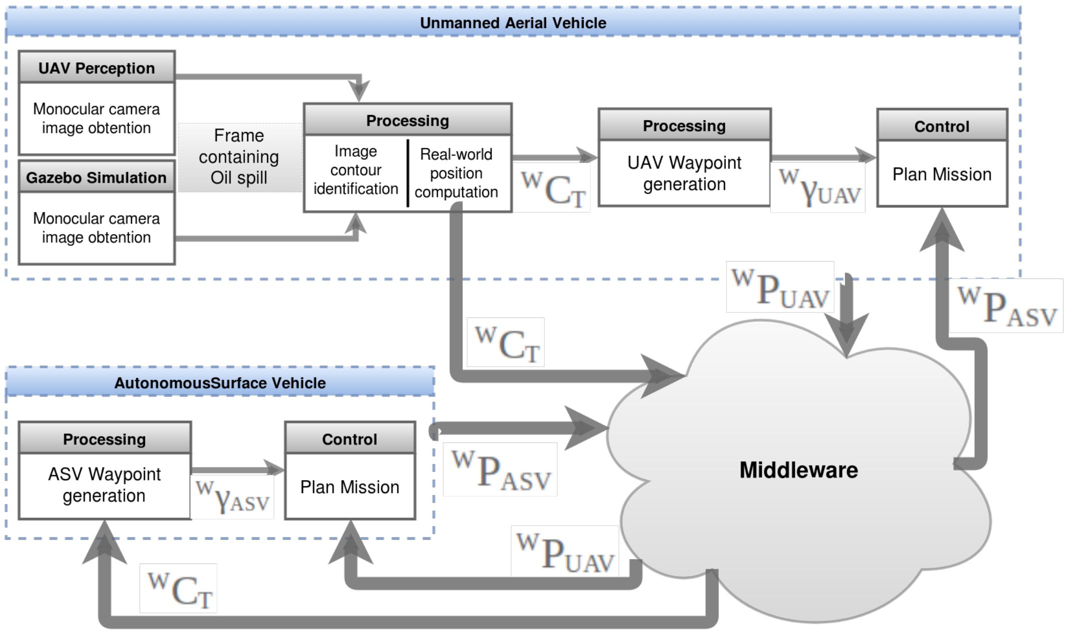


Figure 4.1: Proposed Architecture for Oil Spill Mitigation.

In this section the oil spill mitigation proposed architecture, for a cooperative action between the ASV and the UAV, is described. The figure 4.1 depicts the data flow between both vehicles through a middleware, a middleware is used to allow the increase of the number of vehicles used in the manoeuvre. The oil spill is either detected through a monocular camera on the UAV or through a simulation of that camera within the Gazebo simulator scenario. The frame containing the oil spill is processed to extract the coordinates of the points that form the oil contour, this contour points image coordinates are then transformed into real-world coordinates, ${}^{W}C_{T}$, and sent through the middleware to the ASV. The aerial vehicle is also responsible for using those contour points positions, that describe the oil spill, to plan its oil spill mitigation waypoints position in the world referential, depicted in the Figure as ${}^{W}\gamma_{UAV}$.

The ASV subscribes to the contour points world coordinates, ${}^{W}C_{T}$, and uses the developed control-law path planning algorithm, to plan its oil mitigation waypoints, ${}^{W}\gamma_{ASV}$. Furthermore, both vehicles share their positions, ${}^{W}P_{ASV}$ and ${}^{W}P_{UAV}$, through the middleware, to plan efficiently the cooperative manoeuvre.

In order to evaluate the performance of the control-law algorithms, a simulation environment was developed in Gazebo under the framework ROS. This approach provides a straightforward integration between the simulation environment and the real robots.

Considering the careful analysis of simulators presented in section 3.3, the decision for the Oil Spill Simulator fell in the Gazebo Simulator due to its better performance in sensor modelling and physical fidelity. This simulator granted the author the capability to recreate a environment with high visual and physical fidelity, with behaviours as buoyancy or drag and with a vast number of already developed sensors. Different perspectives of this simulation environment are depicted in the Figures 5.6, 5.7 and 5.8.

4.2 UAV's trajectory computation

To compute the UAV trajectory, in a efficient manner, is necessary to know in advance the optimal manoeuvre height for the UAV, represented in the Figure 4.2 as the distance between the spreader nozzle and the surface, and the angle of dispersion of the powder spreader nozzle.

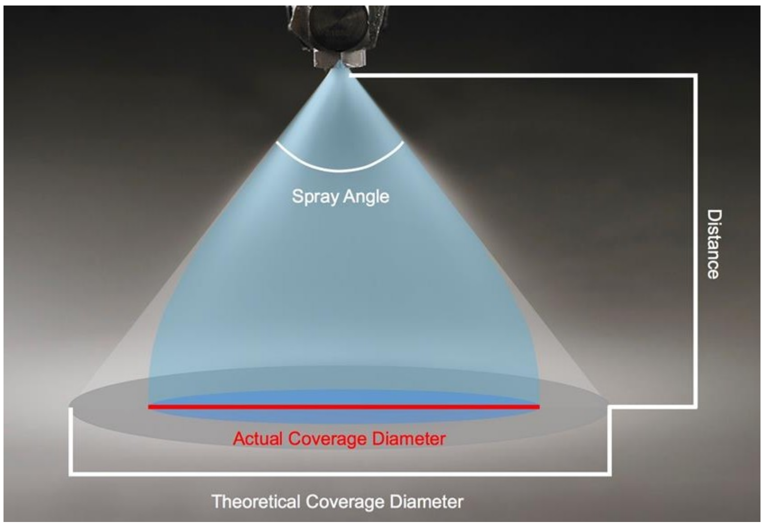


Figure 4.2: Angle of dispersion of the powder spreader nozzle.[1]

Assuming that the dispersion of the nozzle forms the cone depicted in Figure 4.3 and to simplify the algorithm computation, its action area on the water surface can be roughly estimated as the largest square to fit in the base of said cone, represented in blue in the Figure. The length of said square, L, can be obtained if the manoeuvre height for the UAV, H, and the angle of dispersion of the powder spreader nozzle, θ are known. First, the diagonal of the cone base, D, is computed using the equation 4.1. The next step is to use the equation 4.2, to obtain the length of the action square.

$$D = 2 * (H * \tan\theta) \tag{4.1}$$

$$D^2 = L^2 + L^2 \Leftrightarrow L = \sqrt{(D^2)/2} \tag{4.2}$$

After the determination of the dimensions of the action area, is possible to start the computation of the UAV's trajectory. This trajectory should ensure a complete

[1]https://www.lorric.com/en/WhyLORRIC/Nozzle/Cone-spray-coverage-by-distance

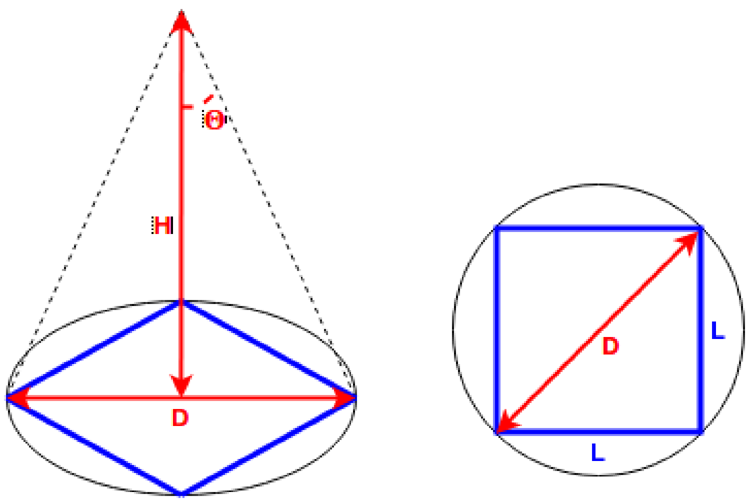


Figure 4.3: Coverage area of the powder spreader nozzle.

coverage of the surface of the spill, for a distribution of the lyophilized powder over its entire surface.

To achieve the trajectory depicted in red in the Figure 5.14, the UAV path planning algorithm starts by dividing the points that define the spill into several horizontal layers, represented in the same figure in green, with the corresponding width from the action area square, at the previously determined flight altitude.

The computed UAV's waypoints, represented in the Figure 4.4, correspond to the minimum and maximum contour points, along the X axis on each horizontal layer. These waypoints, depicted as red asterisks, when reached in the correct order and maintaining the flight altitude between them, ensure complete coverage of the oil spill area with minimal overlap and consequently, minimal waste of resources.

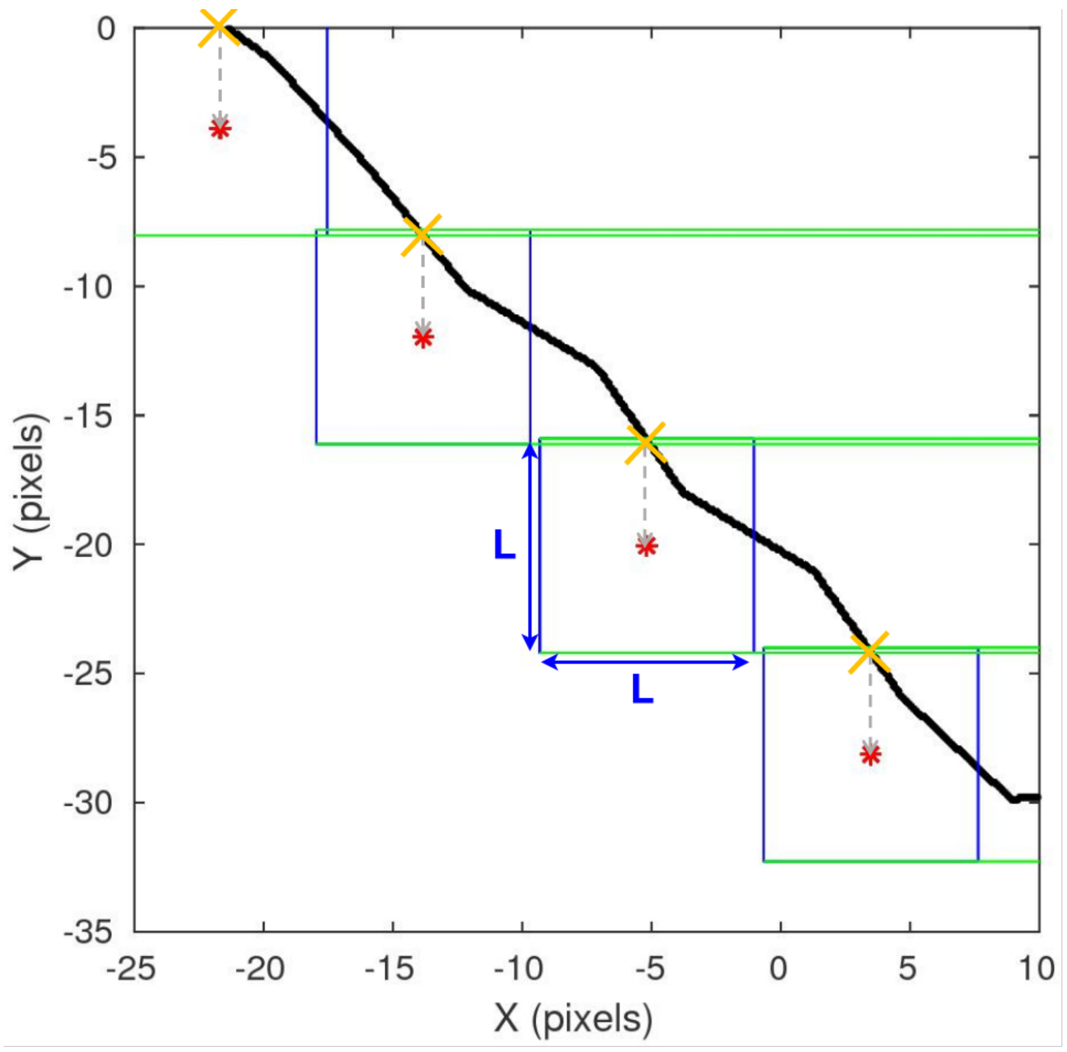


Figure 4.4: Waypoints computation through the minimum and maximum contour points along the X axis on each horizontal layer.

4.3 ASV's trajectory computation

The computation of the trajectory for the ASV, is obtained through potential fields applied to the surface vehicle. This non-holonomic vehicle is incapable of sideways movement since it only has two independently driven thrusters, resulting in distinct attractive and repulsive forces, $U_{goal}(q)$ and $U_{obstacle}(q)$, respectively, being applied on the front and on the rear body of the vehicle, being the repulsive forces, represented in Figure 4.5, computed by the distance between obstacle points and the contour of the vehicle's body, those repulsive and attractive forces constitute the resultant force, $U(q)$, represented in Figure 4.6, that determines the motion of the vehicle.

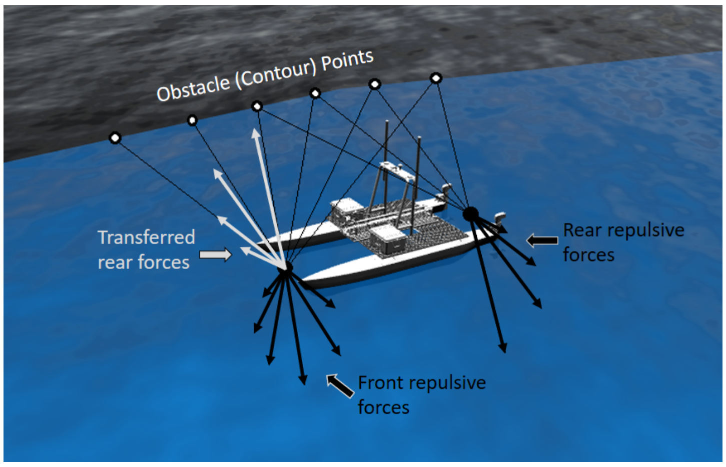


Figure 4.5: Repulsive forces applied to the ASV for oil spill avoidance.

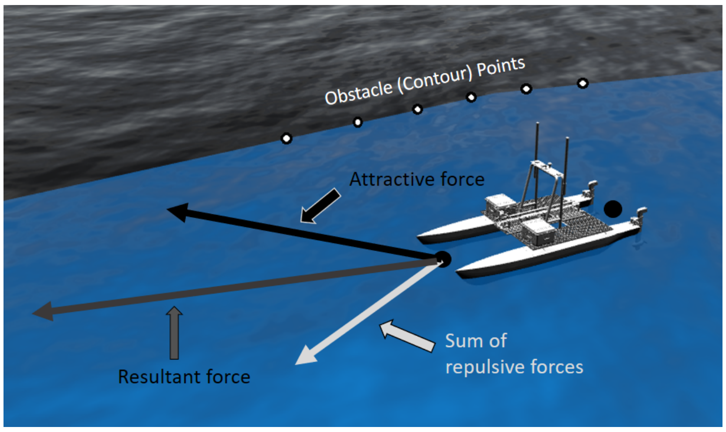


Figure 4.6: Resultant force from repulsive and attractive forces.

For the scenario described on this dissertation, the used algorithm can't be the standard Potential Fields algorithm, since there is no real goal defined and the objective is not simply to avoid getting to close to obstacles, but to follow the contour boundaries of an obstacle, the oil spill, while maintaining a safety distance from it. From the standard Potential Field approach two new algorithms were created to better suit the needs of this scenario, in addition to those, a third algorithm based on a new curvature-based approach, formulated in this dissertation and denominated "Normal Vectors", this approach computes a new and enlarged contour.

4.3.1 Method I: Artificial Potential Fields with 8 interchangeable goals

To obtain the obstacle contour behaviour, for each individual contour detected, 8 interchangeable goals were created on the extremes of the contour. One at a time, each of these goals attract the robot, in a successive clockwise way, granting the vehicle a almost circular motion centred in the centroid of the oil spill (obstacle). Apart from the attractive force exerted by a predetermined goal at any moment, $U_{goal}(q)$, the sum of the repulsive forces exerted by each of the near contours of the oil spill, $\sum U_{obstacle}(q)$, is still exerted over the robot. This algorithm follows the standard potential field equation, presented in the equation 4.3, to compute the resulting force at any moment, $U(q)$.

$$U(q) = U_{goal}(q) + \sum U_{obstacle}(q) \tag{4.3}$$

This algorithm, will move the robot towards the next goal, avoiding to get to close to the obstacle boundary, granting it then, in theory, a contour trajectory distanced from the obstacle's boundaries. This distance can be increased if the ratio between attractive and repulsive forces is increased and vice versa.

4.3.2 Method II: Artificial Potential Fields with an extra Tangential Force

The second approach is also an adaptation of the Potential Fields algorithm, this time the robot is simultaneously attracted to the goal, positioned in the centre of the obstacle and repelled by each point in its contour. This set of forces keep the robot at a determined distance from the obstacle's boundaries. This distance can be adjusted by balancing the ratio of attracting and repulsive forces.

Now to grant a contour motion to the robot, a third force is applied to it, the resultant force from a new Tangential Field, described in [89] and depicted in Figure 4.7. This

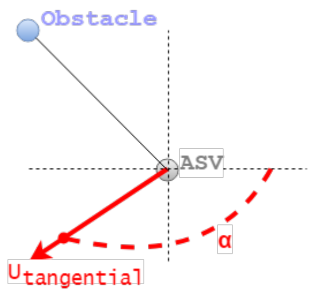


Figure 4.7: Representation of the behaviour of the orientation component of the Tangential Force.

clockwise circular motion, by itself, forces the robot to move in a circular motion, with the motion centre aligned with the oil spill centre. Now, instead of simply following the equation 4.3, a Tangential force, $U_{tangential}$ is combined with the attractive and the repulsive forces, as represented in the equation 4.4, moving the robot on a contour trajectory, distanced from the obstacle's boundaries.

$$U(q) = U_{goal}(q) + \sum U_{obstacle}(q) + U_{tangential}(q) \tag{4.4}$$

While $U_{tangential}$ represents a constant force during the entire path planning, that can be adjusted to surpass the "*local minimas*" on each scenario, its orientation component, α, varies during the manoeuvre depending on the ASV position relatively to the obstacle centroid, as depicted in Figure 4.7. That orientation can be obtained with the equation 4.5, where $P^{W}_{ASV}\_x$ and $P^{W}_{ASV}\_y$ represent the position of the ASV and $P^{W}_{obstacle}\_x$ and $P^{W}_{obstacle}\_y$ represent the position of the obstacle (oil spill) centroid.

$$\alpha = arctan2(P^{W}_{ASV}\_y - P^{W}_{obstacle}\_y, P^{W}_{ASV}\_x - P^{W}_{obstacle}\_x) - \pi/2 \tag{4.5}$$

4.3.3 Method III: Control Points through Normal Vectors with Artificial Potential Fields

This algorithm computes a new enlarged contour, resorting to normal vectors at points from the original contour. After obtained the new control points from the enlarged contour, they are put through a standard potential field algorithm as goals, following the equation 4.3 to compute the resulting force. The algorithm moves the surface vehicle through all of the points, while avoiding getting too close to the original contour.

The control points previously described are computed using a method that resorts to Normal Vectors. This method uses three successive oil spill contour points at a time ($N-1$, N and $N+1$) to form a new set of points, distanced from the original contour points, that describe a new, enlarged contour. This newly generated list of points represents then, the intended ASV trajectory.

The algorithm starts by taking three consecutive points from the contour, N-1(x$_{N-1}$, y$_{N-1}$), N(x$_N$, y$_N$) and N+1(x$_{N+1}$, y$_{N+1}$) and computing the Euclidean distance $d(N-1, N+1)$ between N-1 and N+1, equation 4.6.

$$d(N-1,N+1) = \sqrt{(x_{N-1}-x_{N+1})^2+(y_{N-1}-y_{N+1})^2} \tag{4.6}$$

The variables D_x and D_y are obtained by simply taking the positions differences in x and y, respectively, and divide them by the Euclidean distance calculated previously, as represented in equations 4.7 and 4.8.

$$D_x = \frac{(x_{N-1}-x_{N+1})}{d(N-1,N+1)} \tag{4.7}$$

$$D_y = \frac{(y_{N-1}-y_{N+1})}{d(N-1,N+1)} \tag{4.8}$$

Now, with D_x and D_y computed, is relatively easy to obtain a new point (x, y), distanced κ from N(x$_N$, y$_N$), as represented in equations 4.9 and 4.10.

$$x = x_N + \kappa * D_y \tag{4.9}$$

$$y = y_N - \kappa * D_x \tag{4.10}$$

By repeating this procedure through every successive combination of three points in the contour ($N-1$, N and $N+1$), a new set of points is obtained, distanced from the original contour points, that describe a new, enlarged contour, representing then, the intended ASV trajectory.

These equations describe the simple procedure of taking three consecutive points from the original contour, $N-1$, N and $N+1$, grey points at the figure 4.8, compute the line segment that passes through $N-1$ and $N+1$, represented in black. Next, the normal vector to that line segment at N is obtained, represented in red in the figure. The last step is the computation of a new point, represented in green, on that normal, distanced κ from the original point N.

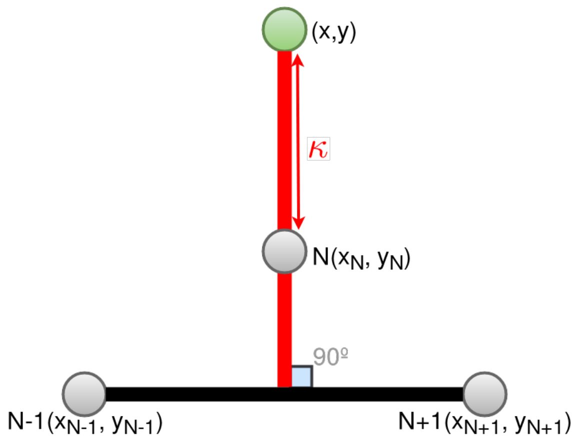


Figure 4.8: New point based on three consecutive contour points.

Chapter 5

Implementation

In this chapter, a brief description of the UAV and ASV used for the field tests, described later on this dissertation, is presented, followed by an explanation on how the oil spill scenario was created and simulated on the Gazebo simulator. The process of identifying the oil spill in the camera frame, the extraction of its contour borders, the computation of its real world position and the computation of the UAV and ASV trajectories from the simulated scenario are also addressed in this chapter.

5.1 Autonomous Vehicles

5.1.1 Unmanned Aerial Vehicle (UAV): STORK I

STORK I, depicted in Figure 5.1, is an hexarotor UAV, built in 2015 by a Portuguese research institution, INESC TEC and by the Autonomous Systems Lab of the School of Engineering of the Polytechnic Institute of Porto (ISEP), capable of autonomous take-off and landing, real time sensor data acquisition with on-board processing and autonomous missions with obstacle avoidance.

The vehicle was developed by INESC TEC for this project, built from carbon fiber and plastic, with 90 cm of total diameter and with a height of 70 cm, a payload capacity of 4.9 kg and with an autonomy reaching the 25 minutes of flight. The UAV makes use of the open-source autopilot from Pixhawk project, running PX4 firmware.

The navigation sensors, IMU and Ublox Neo M8N Global Navigation Satellite System (GNSS) and the interchangeable sensorial payload, 0.3 MP PointGrey Firefly FMVU-03MTC-CS visible camera, 2.3 MP Pointgrey Grasshoper GS3-U3-23S6C-C, termographic camera and a LIDAR allow the vehicle to be suitable for a wide range of distinct

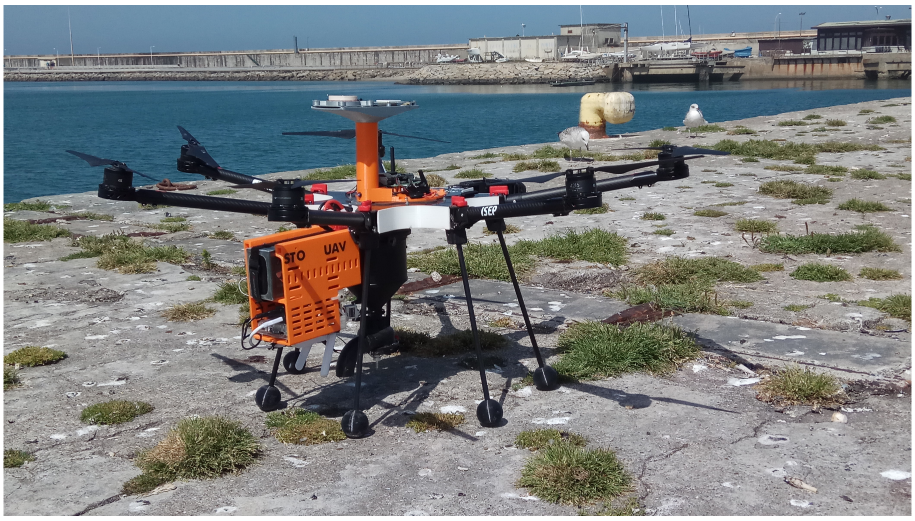


Figure 5.1: Hexarotor AUV STORK I.

applications such as search and rescue, aerial surveillance and inspection, 3D mapping and target identification, localisation and tracking.

For the SpilLess and ROSM projects a sprinkler like, spraying system for the Lyophilized powder, described in the Figure 5.2, was designed and linked to the UAV bellow its frame, visible in the Figure 6.3.

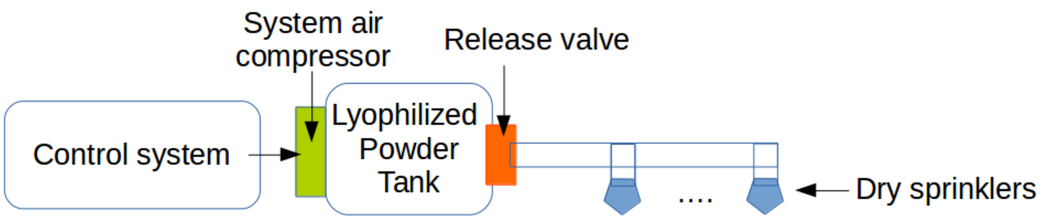


Figure 5.2: UAV's spraying system for the Lyophilized powder.

Other vehicle specifications:

- On-board computation: Odroid XU3 running Ubuntu 14.04 with ROS Indigo;
- Weight: 3 kg;
- Power: rechargeable batteries (LiPo, 22000 mAh);
- Payload interfaces/ports: USB (2.0 and 3.0);
- Maximum height: 300 m;

- Range: 300 m;
- Propulsion: six brushless rotors;
- Horizontal speed: 0-10 m/s;
- Vertical speed: 0-6 m/s (ascent and descent);
- Degrees of freedom: throttle, roll, pitch, yaw;
- Communications: Wi-Fi (5 GHz), telemetry (433 MHz), emergency stop (2.4 GHz);
- Navigation system: GPS, Flight Control Unit (with accelerometer, barometer, gyroscope, magnetometer) and IMU.

5.1.2 Autonomous Surface Vehicle (ASV): ROAZ II

The vehicle ROAZ II, depicted in Figure 5.3, is a ASV with the shape of a catamaran, built in 2008 by a Portuguese research company, INESC TEC and by the Autonomous Systems Lab of the School of Engineering of the Polytechnic Institute of Porto (ISEP), capable of autonomous operation as a result of on-board sensor processing and high precision navigation.

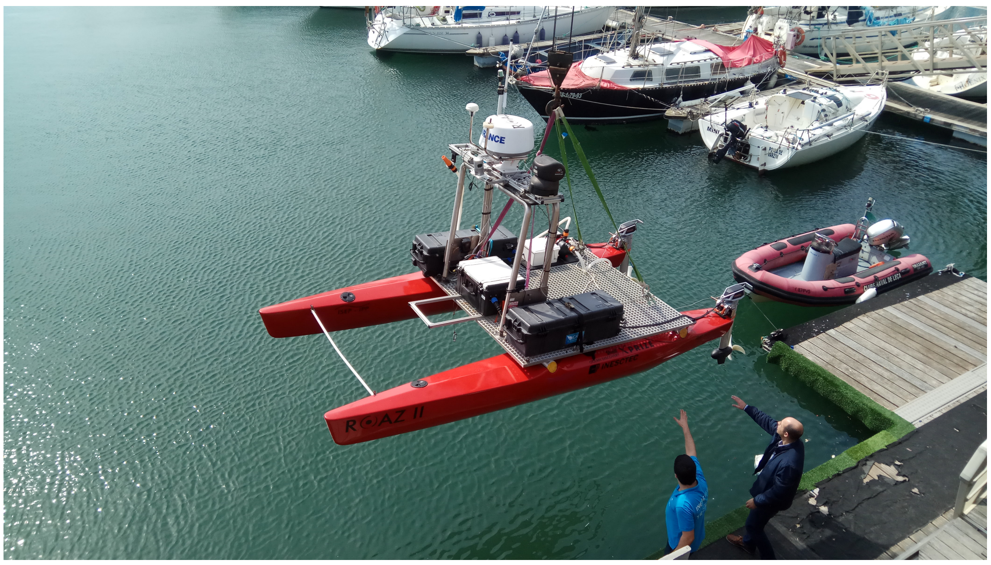


Figure 5.3: Autonomous Surface Vehicle ROAZ II.

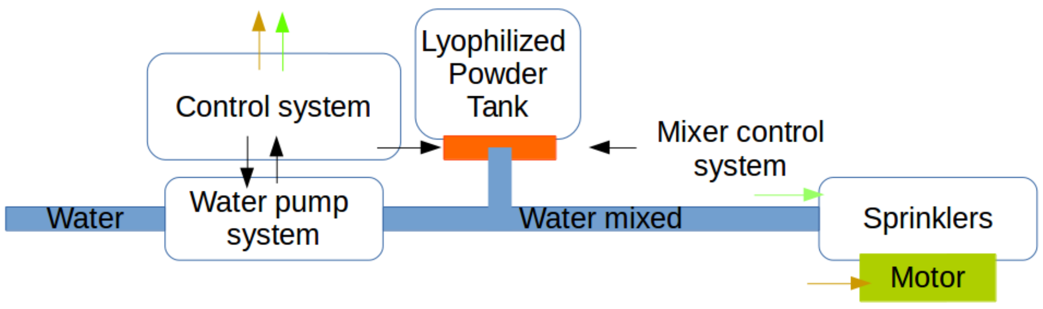


Figure 5.4: ASV's spraying system for the Lyophilized powder.

The developed vehicle with a length of 4.5 m, width of 2.2 m and weight of 400kg with the capability to handle 300 kg more of payload, is built from Polyethylene and Aluminium and it's continuously being adapted to different projects and missions.

With several interchangeable sensors: Side-scan Sonar, Multibeam Echosounder, Sub-Bottom Profiler, Video cameras (visible and thermographic), Sound velocity probe, DVL with ADCP option, GPS with RTK and INS, Radar, Infra-red, and CTD (conductivity, temperature, and depth) instrument, is suitable for a wide range of applications such as aquatic environment monitoring, data collection and oceanography, environmental modelling (oceanographic, 3D sea floor modelling), bathymetry, security, area patrol, automated intrusion detection, target identification and tracking, search and rescue and cmmunications relay in multi-vehicle scenarios and surface support to underwater assets.

For the SpilLess and ROSM projects a sprinkler like, spraying system for the Lyophilized powder, described in the Figure 5.4, was designed and linked to the side of the ASV, visible in the Figure 5.5. This system is composed by a water pump that obtains water from the ocean and mixes the Lyophilized powder in that water before releasing it through its sprinkler.

Other vehicle specifications:

- Deployment: boat trailer or crane;
- Power: rechargeable batteries (LiFePO4, 4800 Wh);
- Autonomy: 10 h;
- Range: 60-100 km;
- Payload interfaces/ports: ethernet, serial RS232/485, CAN bus (IP68 or underwater connectors plugs);

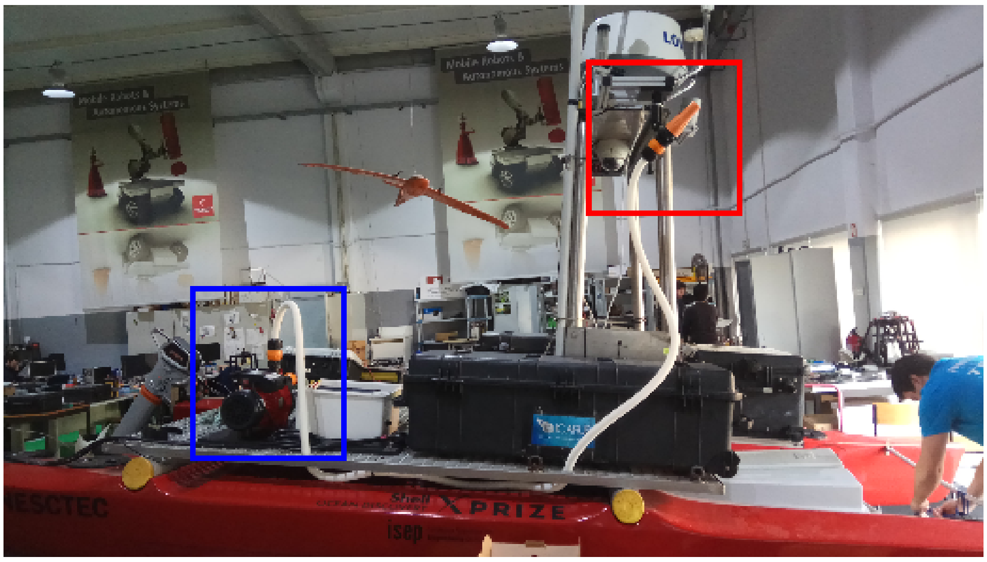


Figure 5.5: Implementation of the spraying system in the ASV being the water pump system noticeable in blue and the motorised sprinkler in red.

- Wireless communication (data/video);
- Propulsion: two electric thrusters (independent);
- Propulsion power: 10 HP;
- Maximum speed: 5 m/s (10 knots);
- Degrees of freedom: 3DOF;
- Communications: Wi-Fi, RF, Iridium, underwater acoustic communications;
- Modes of operation: teleoperation, autonomous waypoint following, station keeping and adaptive autonomous mission.
- Landing system for UAVs;

5.2 Scenario creation and simulation

To obtain a realistic simulation of the mitigation of an oil spill incident, the recreation of this scenario was implemented within the Gazebo simulator. The main features of this scenario are the oil spill in open waters, the existence of two vehicles, an ASV ROAZ II and an UAV STORK I with the capability for a cooperative and coordinate manoeuvre between them for the oil spill mitigation. That manoeuvre consists on a contour trajectory by the ASV, while deploying microorganisms and nutrients (bioremediation), capable of mitigating and containing the spillage and, simultaneously, on a coverage trajectory of the affected area by the UAV.

To achieve the level of realism depicted in the Figures 5.6, 5.7 and 5.8, several different parts had to be combined:

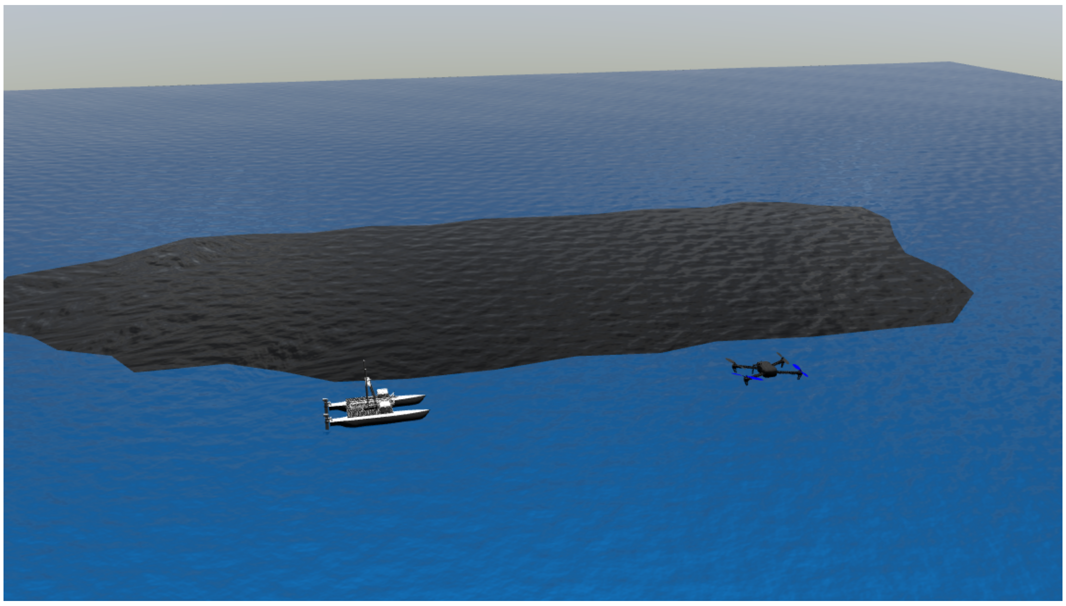


Figure 5.6: Oil spill and vehicles simulation in Gazebo.

The usage of Ubuntu 14.04 LTS;

The installation of the chosen robotic simulator, Gazebo 7.0;

The installation of ROS Indigo;

The integration of the package "*uuv_simulator*" (Unmanned Underwater Vehicle simulator) from 2016, under constant development by Manhães et Al. [77]. This package provided the simulation with an realistic ocean environment representation;

Modification of the squared mesh without thickness used by the "*uuv_simulator*"

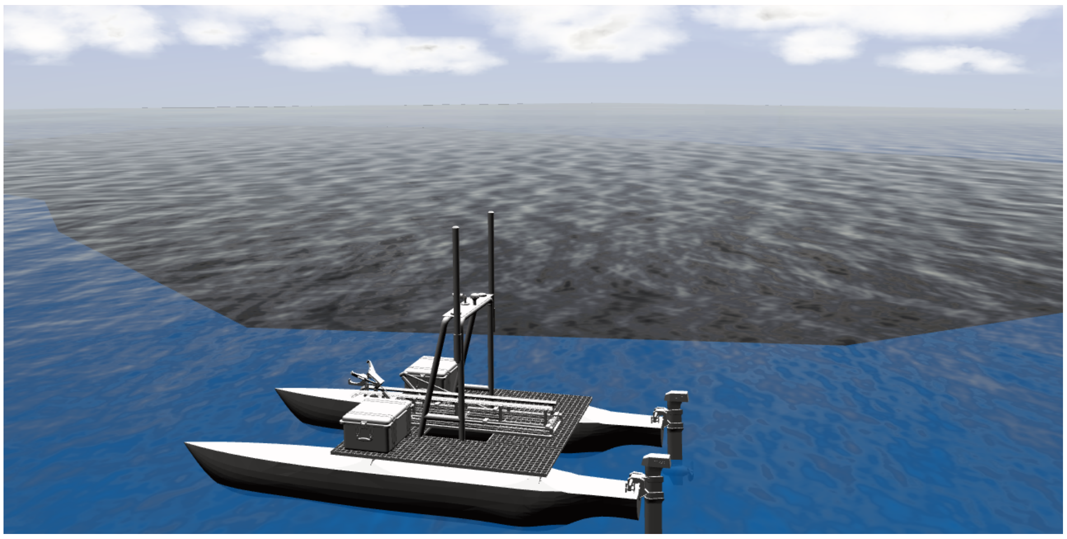


Figure 5.7: Oil spill from the ASV perspective.

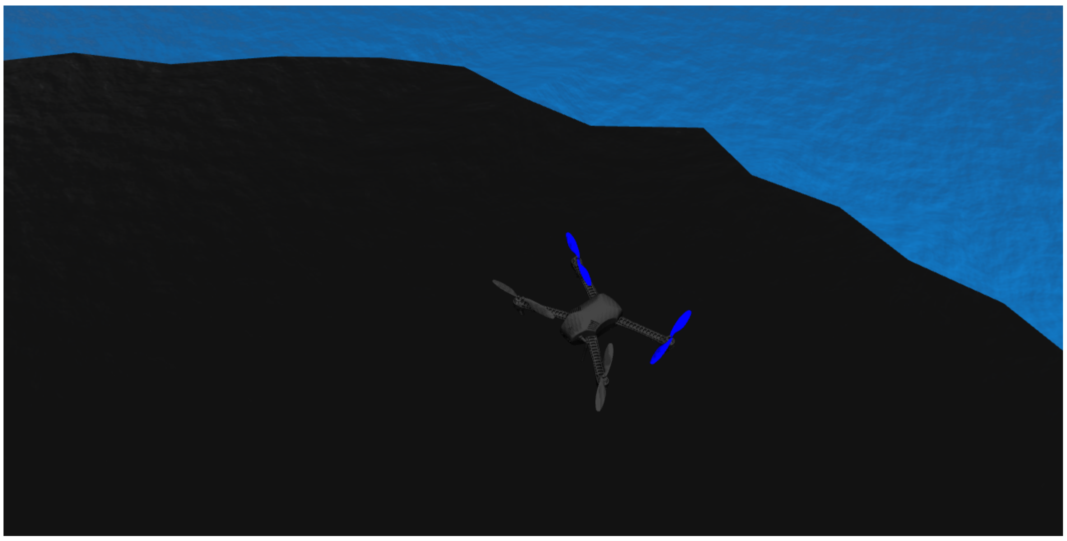


Figure 5.8: Oil spill from the UAV perspective.

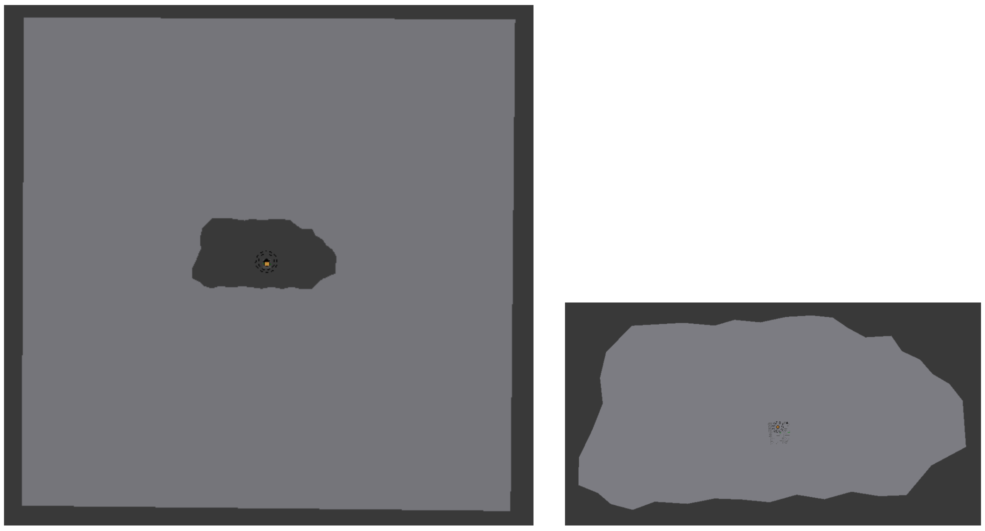


Figure 5.9: Model of the ocean surface (left) and oil spill (right).

package to represent the ocean surface and exportation as two different meshes, ocean surface mesh without the oil spill and the oil spill mesh itself, as depicted in Figure 5.9 and inclusion of both into the simulation world. If a new mesh was created for the oil spill area and overlapped over the previously existent ocean mesh, the distinct wave motion on both surfaces would be noticeable, therefore, the oil spill area must be removed from the ocean surface mesh. For the oil spill mesh a darker colour is used;

The integration of the MAVROS ROS package, that supports the bridge between ROS and the MAVLink protocol. This bridge allows the communication between computers running ROS and MAVLink enabled autopilots, like the PX4 flight stack. Based on the communication protocol between the PX4 and the application in ROS, we are able to communicate between the low level UAV control and the high level.

The usage of Iris UAV model, depicted in Figure 5.11, from the SITL Gazebo simulation, provided by the PX4 Developer Guide[90] that uses MAVROS MAVLink node to communicate with PX4 Autopilot Firmware. This UAV model is used to simulate the STORK I vehicle;

Integration of a undistorted camera model into the UAV with a plugin that publishes the camera feed into a ROS topic, to obtain a aerial perspective of an oil spill scenario.

Integration of an realistic model of the ASV ROAZ II, depicted in Figure 5.12, into the simulation world as a mesh in a Collada file (.dae format);

The author suggests the usage of the referred software, though later versions should work correctly, with some minor adjustments if required.

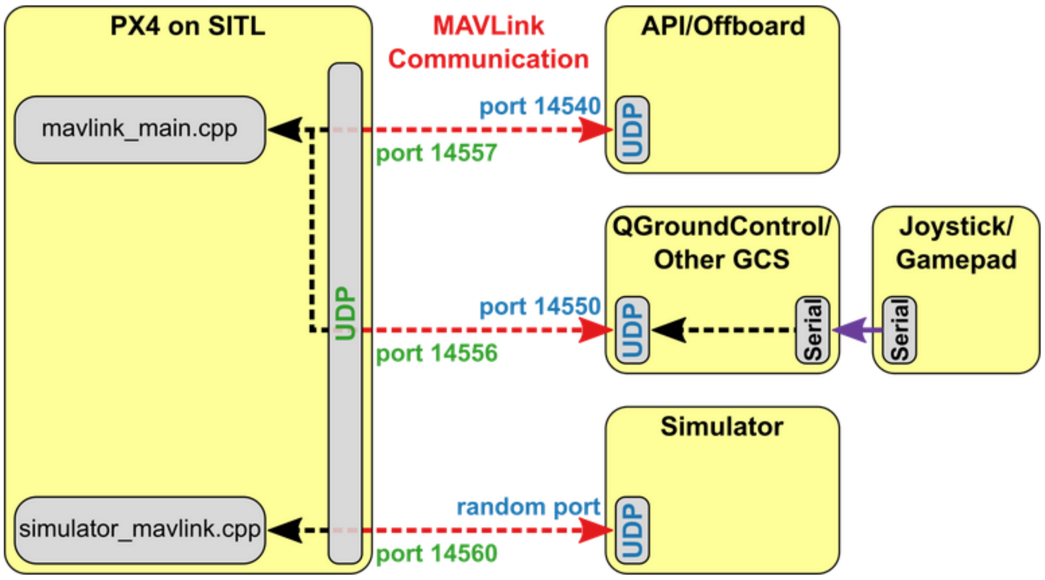


Figure 5.10: ROS/Gazebo integration with PX4.

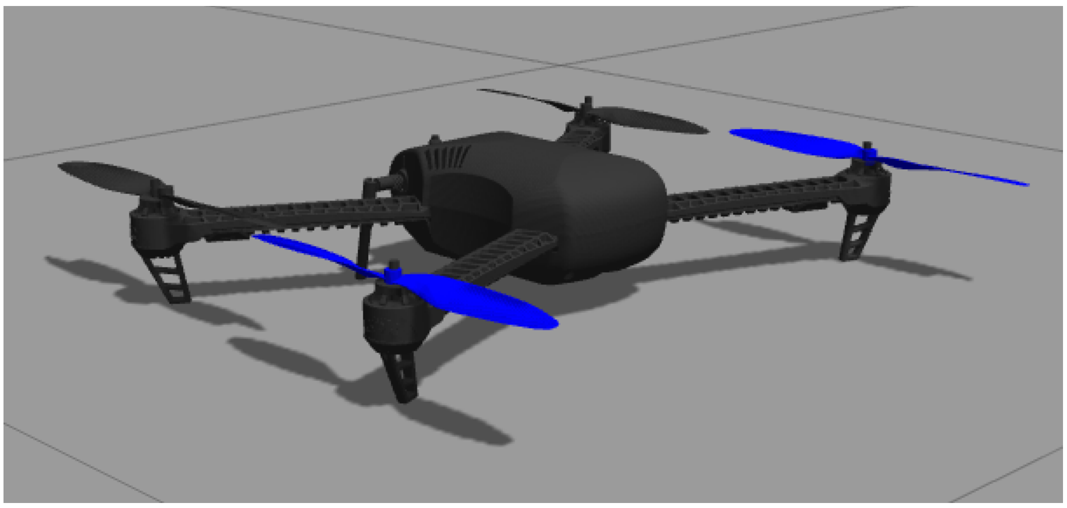


Figure 5.11: Model of the Iris UAV.

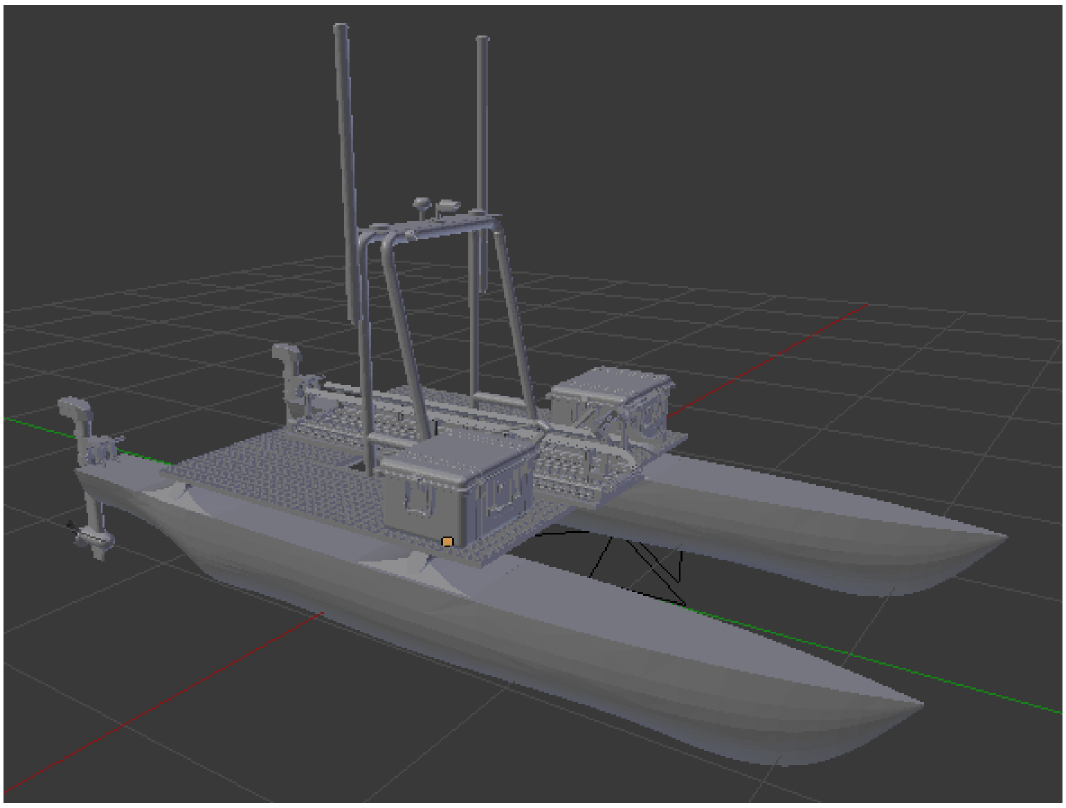


Figure 5.12: Model of the ROAZ II ASV.

5.3 Spill identification and real world position computation

In order to identify and obtain the position of the oil spill, in the real world, a node "*move_drone*" was created that forces the UAV model to perform an ascending trajectory to a higher position within the oil spill area. Ensuring that the entire affected area is captured by the field of view of the camera. Once it reaches that position, waits for the waypoint publication in the rostopic "*/drone_waypoints*", to initiate the trajectory.

Once the entire spill it's contained within its camera frame, a ROS node, named "*spillage_detection_node*", subscribes to the UAV camera feed, being published as a *rostopic*, "*camera/image_raw*". Based on that image, a *findContours* function from the Open Source Computer Vision Library (OpenCV)[1], with the correct threshold, obtained through several tests for this specific scenario, is applied in order to obtain the position of each point of the contour from the oil spill, in pixels from that frame, as depicted in Figure 5.13.

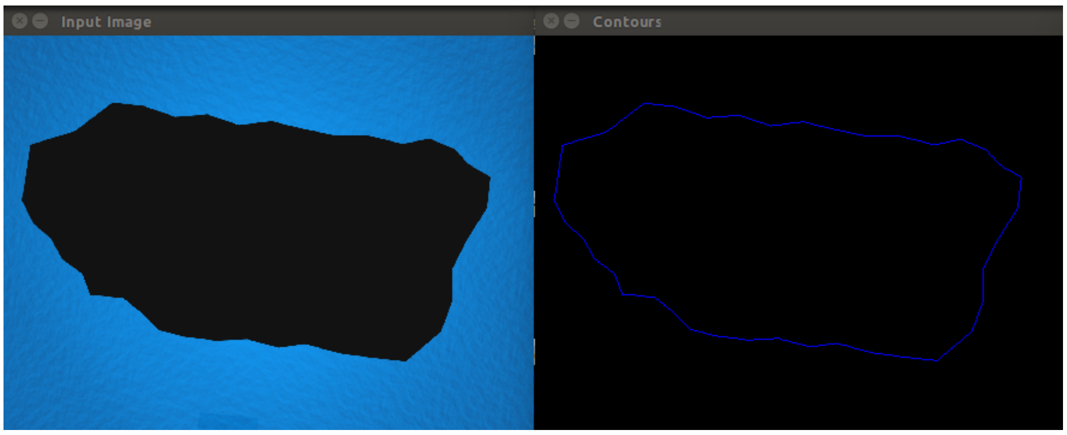


Figure 5.13: Input (left) and output (right) through OpenCV "findContours" function.

To obtain the projection of the contour points in the real world is necessary to go through a series of transformations. The first matrix necessary to obtain that projection is described in the equation 5.1 with K and corresponds to the camera Intrinsic Matrix, a 3 by 3 matrix where f_x and f_y stand for the focal length, in pixels, x_0 and y_0 for the principal point offset of the camera and s for the axis skew of the camera.

[1]https://docs.opencv.org/2.4/

$$K = \begin{bmatrix} f_x & s & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 568.238 & 0 & 644 & 0 \\ 0 & 568.238 & 482 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.1}$$

The rotation matrix between the camera and UAV body reference frames, R_c^b, is presented in the equation 5.2. This 3 by 3 matrix is constant throughout the entire manoeuvre, since the camera orientation does not change within the UAV body reference frame, and represent a *Roll*, *Pitch* and Yaw rotations of π, 0 and $-\pi/2$ radians, respectively.

$$R_c^b = Yaw_c^b(-\pi/2) * Pitch_c^b(0) * Roll_c^b(\pi) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5.2}$$

The transformation matrix between the camera and UAV body reference frames, P_c^b, is presented in the equation 5.3. This 4 by 4 matrix contains the rotation matrix between the camera and UAV body reference frames, R_c^b, and the displacement between the camera and UAV body reference frames, T_c^b, this translation is also constant throughout the entire manoeuvre.

$$P_c^b = \begin{bmatrix} & & & T_c^b\_x \\ & R_c^b & & T_c^b\_y \\ & & & T_c^b\_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

The rotation matrix between the UAV body and world reference frames, R_b^w, is presented in the equation 5.4. This 3 by 3 matrix represents the *Roll*, *Pitch* and Yaw rotations of the UAV in relation to the world referential. This rotations and the translation matrix T_b^w are obtained through the subscription to the rostopic "*/mavros/local_position/pose*", that estimates the UAV pose (through GPS + IMU), being published by the MAVROS node.

$$R_b^w = Yaw_b^w(\gamma) * Pitch_b^w(\beta) * Roll_b^w(\alpha) \tag{5.4}$$

The transformation matrix between the UAV body and world reference frames, P_b^w, is presented in the equation 5.5. This 4 by 4 matrix contains the rotation matrix between the UAV body and world reference frames, R_b^w, and the displacement between the UAV body and world reference frames, T_c^b.

$$P_b^w = \begin{bmatrix} & & & T_b^w\_x \\ & R_b^w & & T_b^w\_y \\ & & & T_b^w\_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

Considering the matrices previously defined it's possible to compute an auxiliary matrix, Ψ, that represents the position and attitude transformation matrix of the camera to the world referential. This matrix is formed by the multiplication of K by the inverse of P_c^b and by the inverse of P_b^w, and its represented in the equation 5.6. In this matrix the third column is removed to allow the multiplication in 5.8, this could only be done since the author is assuming that the ocean surface is a plan with $Z = 0$ and the contour points will be always on that plan.

$$\Psi = K * [P_c^b]^{-1} * [P_b^w]^{-1} = \begin{bmatrix} \Psi_1 & \Psi_2 & \Psi_3 & \Psi_4 \\ \Psi_5 & \Psi_6 & \Psi_7 & \Psi_8 \\ \Psi_9 & \Psi_{10} & \Psi_{11} & \Psi_{12} \end{bmatrix} = \begin{bmatrix} \Psi_1 & \Psi_2 & \Psi_4 \\ \Psi_5 & \Psi_6 & \Psi_8 \\ \Psi_9 & \Psi_{10} & \Psi_{12} \end{bmatrix} \tag{5.6}$$

The matrix with each contour x and y positions, in pixels, from the image frame, $Image\_pos$, is represented in the equation 5.7.

$$Image\_pos = \begin{bmatrix} Pixel_x & Pixel_y & 1 \end{bmatrix} \tag{5.7}$$

Lastly $World\_pos$ corresponds to a vector with the x and y positions of that contour point in real world coordinates. This operation, represented in the equation 5.8, must be applied for each contour point.

$$World\_pos = Image\_pos * [K * [P_c^b]^{-1} * [P_b^w]^{-1}]^{-1} = Image\_pos * \Psi^{-1} \tag{5.8}$$

5.4 UAV's trajectory computation in the simulation scenario

The same ROS node, "*spillage_detection_node*" is responsible for the computation of the optimal oil spill mitigation trajectory for the UAV. This trajectory is depicted, in red, in the Figure 5.14 and is obtained through an orderly following of the waypoints, depicted as red asterisks, by the UAV.

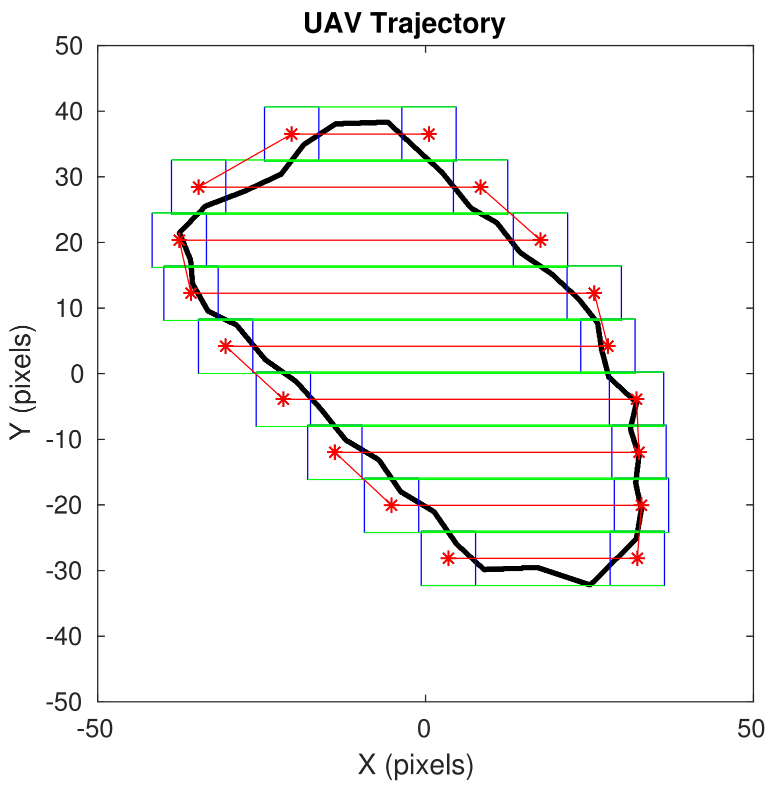


Figure 5.14: UAV trajectory computation.

For the simulation of the UAV manoeuvre, within the Gazebo Simulator, a node was created, "*move_drone*", this plugin, described before is not only responsible for sending the UAV to a higher position within the oil spill area for wider aerial perspective, but also for moving the UAV through the waypoints published into the rostopic "*/drone_waypoints*" once determined the trajectory.

5.5 ASV's trajectory computation in the simulation scenario

To achieve the control-law for the ASV manoeuvre, the previously described ROS node "*spillage_detection_node*", starts by subscribing to the ASV GPS position, through the rostopic "*roaz_gps*", to obtain a trajectory from the vehicle current position and by receiving the vehicle behavior (waypoint/maneuover) to maintain from the oil spill borders. With these values and the contour points real world positions, the node is capable of computing a mitigation trajectory for the ASV based on the three methods described bellow.

5.5.1 Method I: Artificial Potential Fields with 8 interchangeable goals

In the figure 5.15, the result of this first algorithm is presented. When the surface vehicle is in the area A1, is attracted towards the top central goal, but once it enters the area A2, is attracted towards the top right goal and so on, interchanging goals in a clockwise direction. The same algorithm is applied to a scenario with multiple different geometrical forms, in the figure 5.16.

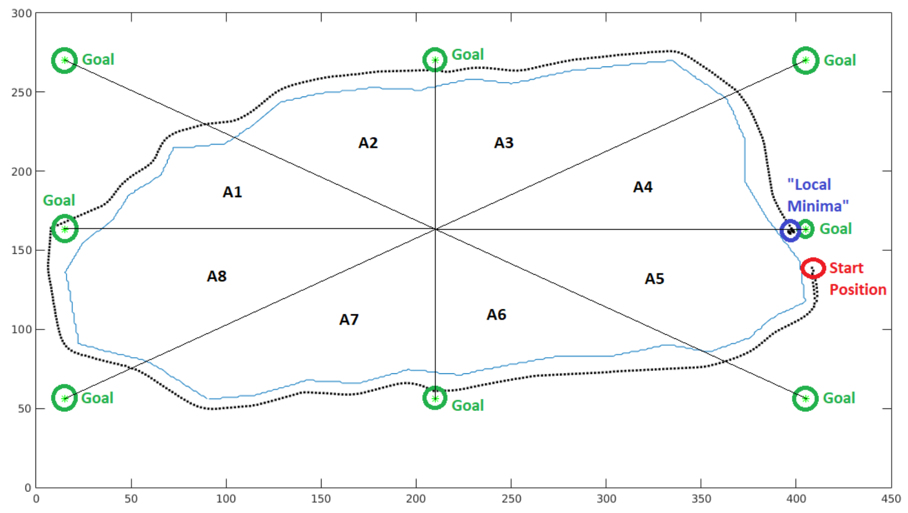


Figure 5.15: Control-law performance of Method I in a simulated oil spill scenario.

This algorithm provides a good trajectory, though it is highly dependable on the shape of the spill, as it is demonstrated on the figures 5.15 and 5.16, where the robot found a "*local minima*", from which it is unable to leave.

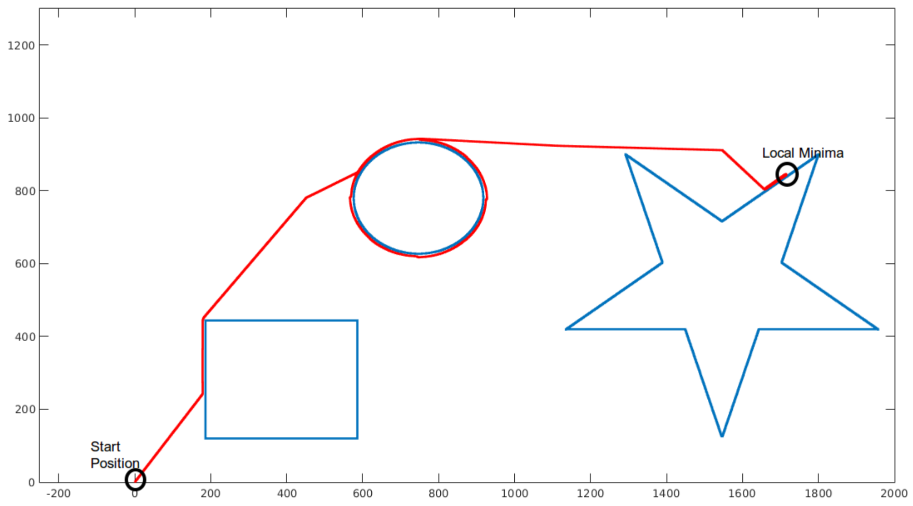


Figure 5.16: Control-law performance of Method I for a simulated scenario with different geometric forms.

5.5.2 Method II: Artificial Potential Fields with an extra Tangential Force

In the figures 5.17 and 5.18, the results of this second algorithm are presented and though it looked promising, in theory, is also affected by a "*local minima*".

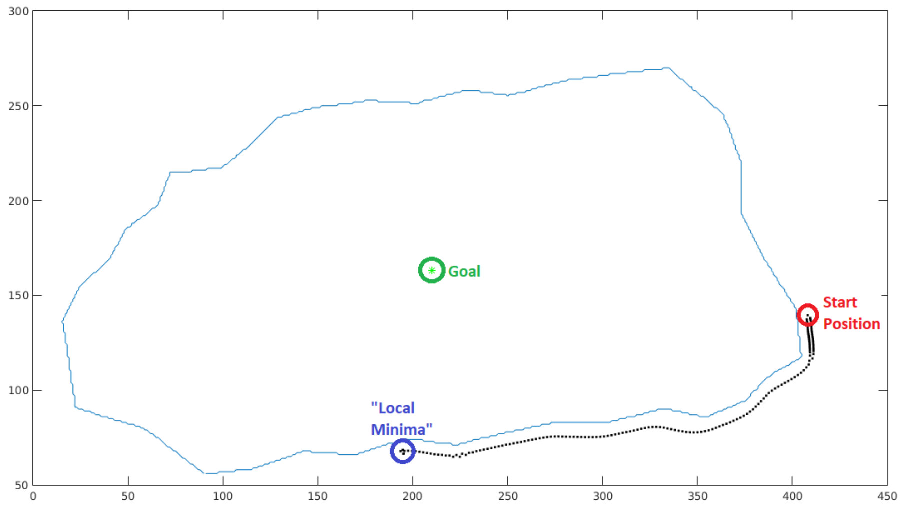


Figure 5.17: Control-law performance of Method II with $U_{tangential}$=3.2 in a simulated oil spill scenario.

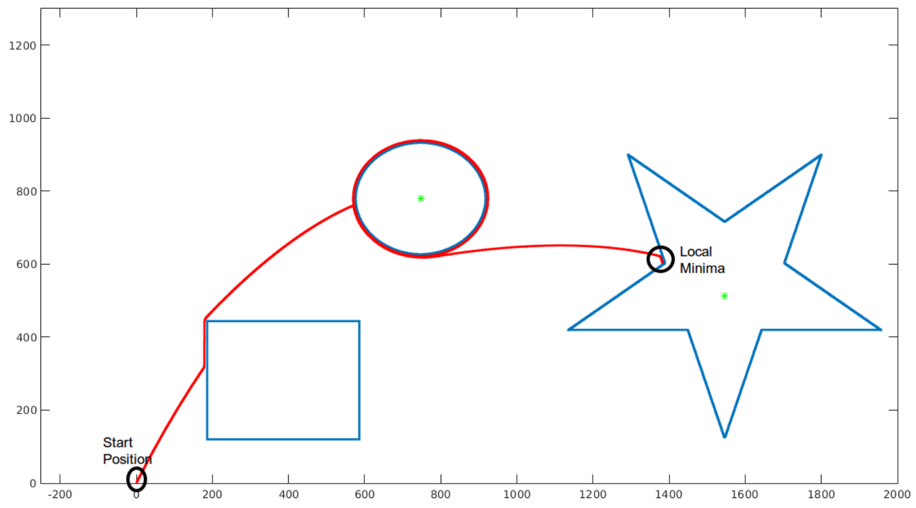


Figure 5.18: Control-law performance of Method II, with $U_{tangential}$=3.2, for a simulated scenario with different geometric forms.

The presence of "*local minimas*" on this algorithm, can be surpassed, in the oil spill scenario, by increasing the value of the tangential force. In the figure 5.17 that value was 3.2, if increased up to 16.2, the robot is already capable of contour the entire spill, as demonstrated in the figure 5.19. This raises another problem, with the increase of the tangential force, the trajectory tends to a circular motion, not respecting then, the main objective of this project, to follow closely the contour of an oil spill. Furthermore, there is no way to define the ideal tangential force value, without the previous knowledge of the entire shape of the obstacle.

If the tangential force is increased up to 30, in the scenario with multiple different geometric forms, the algorithm fails, since the tangential force surpasses the repulsion force created by the spill, moving the robot to the interior of the spill, as represented in the figure 5.20.

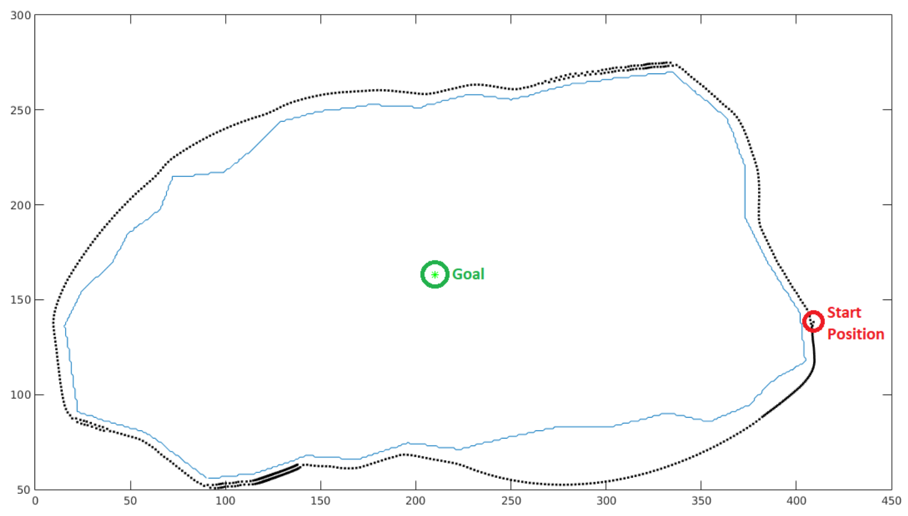


Figure 5.19: Control-law performance of Method II, with $U_{tangential}$=16.2 in a simulated oil spill scenario.

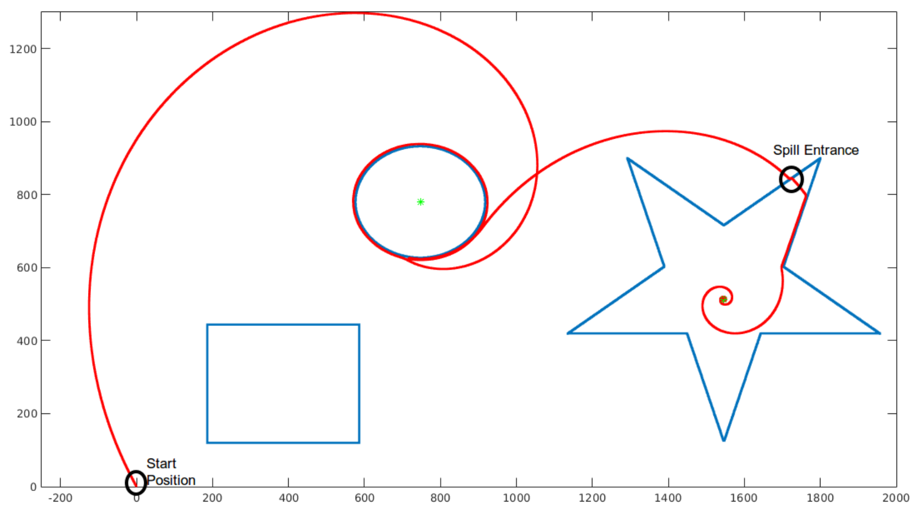


Figure 5.20: Control-law performance of Method II, with $U_{tangential}$=30, for a simulated scenario with different geometric forms.

5.5.3 Method III: Control Points through Normal Vectors with Artificial Potential Fields

The oil spill contour and the trajectory obtained with this algorithm are represented in the figures 5.21 and 5.22.

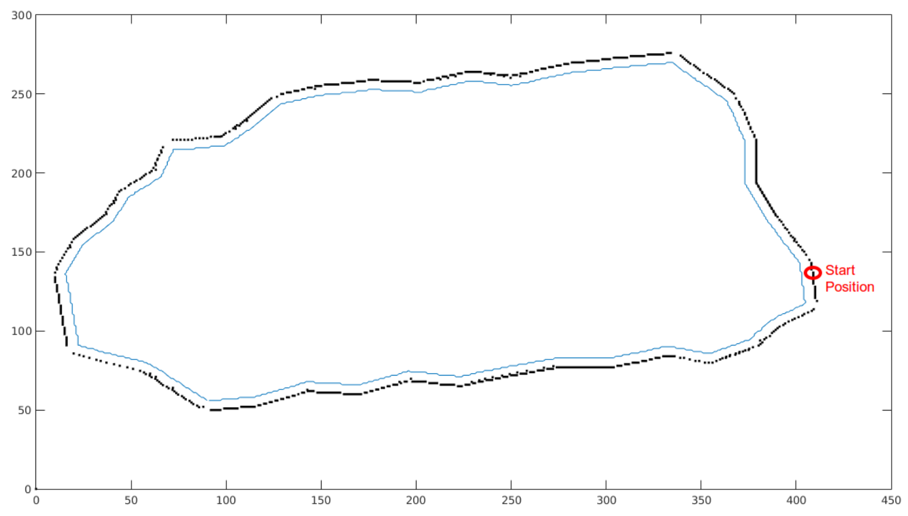


Figure 5.21: Control-law performance of Method III in a simulated oil spill scenario.

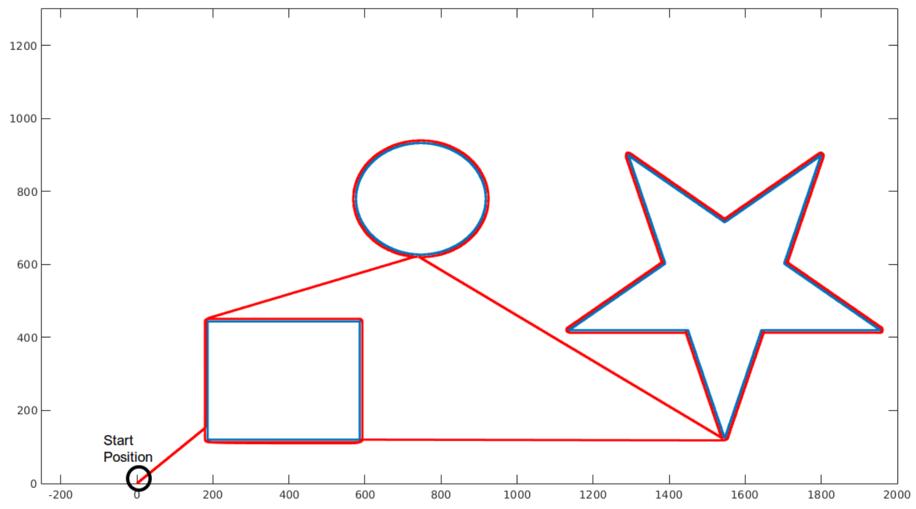


Figure 5.22: Control-law performance of Method III, for a simulated scenario with different geometric forms.

An overlap of the result trajectories for each method, at the oil spill scenario and at the scenario with the multiple geometrical forms, are represented in the figures 5.23 and 5.24, respectively.

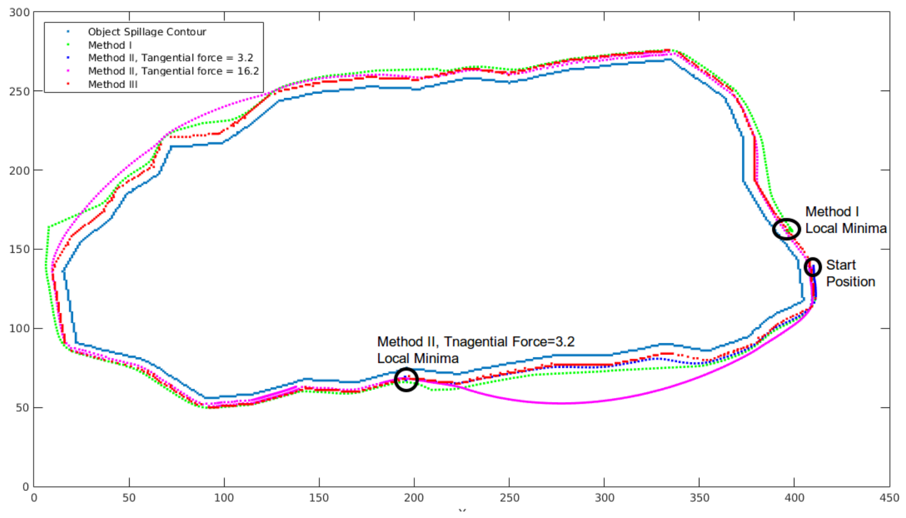


Figure 5.23: Trajectories overlap comparison in a simulated oil spill scenario.

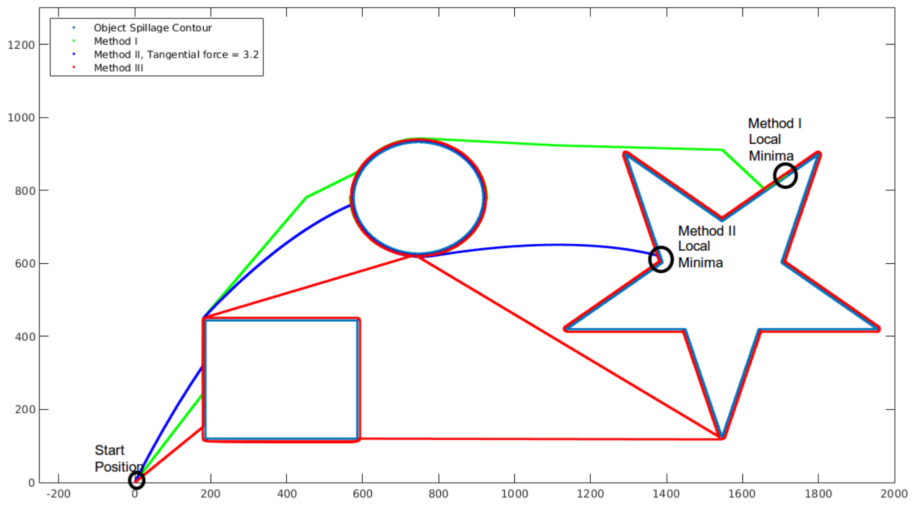


Figure 5.24: Trajectories overlap comparison, for a simulated scenario with different geometric forms.

Since two methods were able to complete the contour, in the oil spill scenario, figures 5.19 and 5.21, the table 5.1 was elaborated to compare the travelled distance performed by each method. In this table, the method III has a shorter travelled distance than the method II with the $U_{tangential} = 16.2$, which indicates a much closer contour following trajectory of the oil spill.

Table 5.1: Travelled distance performed by each method.

Method	Travelled distance (m)
I	Incomplete mission
II, $U_{tangential} = 3.2$	Incomplete mission
II, $U_{tangential} = 16.2$	1536
III	1238

For the scenario described in this dissertation, from the approaches tested, it became clear that, the one that produced the best trajectory, for the oil spill mitigation, was the Method III, that uses normal vectors to compute a list of control points that describe a new and enlarged contour, this list of control points is then passed through a Potential Field algorithm that uses them as successive goals. It is the easiest method to define a exact distance, to be maintained in relation to the oil spill boundaries, along the entire trajectory and does not present trajectory stops unlike the other algorithms that suffer from "*local minimas*".

For the simulation of the ASV manoeuvre within the Gazebo Simulator the node "*move_roaz*" was created, this node is responsible for moving the surface vehicle model through the waypoints published into the rostopic "*/surf_waypoints*" once determined the trajectory.

This page was intentionally left blank.

Chapter 6

Results

The results for this dissertation were obtained on April 23 and 24, in the preliminary tests of the project SpilLess that occurred in the Leixões Harbour in Porto, Portugal, depicted in Figure 6.1, where the oil spill was simulated on the Gazebo simulator scenario, on a known position from the real world and the algorithms for the cooperative and simultaneous manoeuvre between the UAV and the ASV were put to test. In this test for the generation of the surface vehicle trajectory, the method III, generation of control points through normal vectors and artificial potential field, was applied, since it provided the best simulated results.

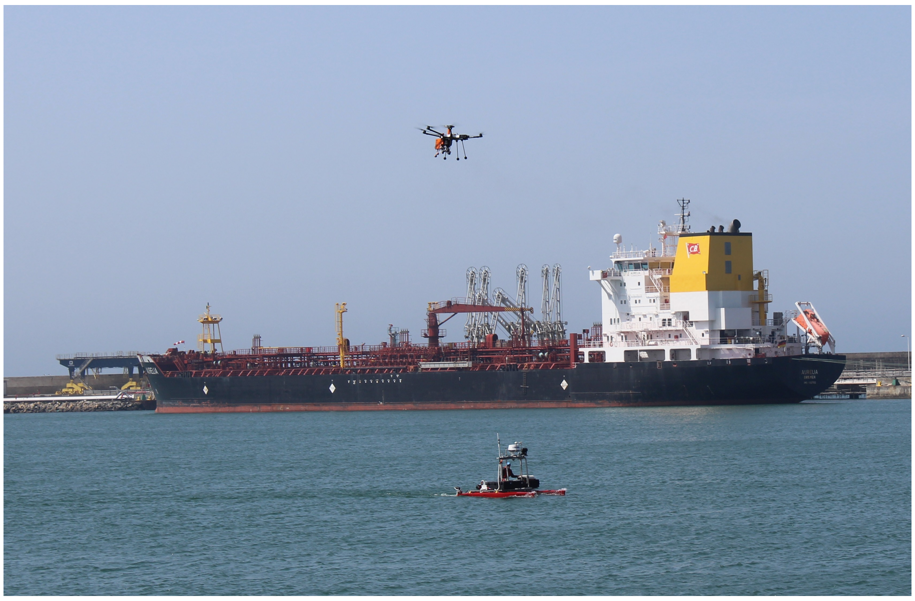


Figure 6.1: Experimental Field Tests at Leixões Harbour.

The position of both vehicles was represented in real time in a ROS tool for 3D visualisation, RViz, with a precise model of the oil spill included into the scenario for manoeuvre monitoring from land. This representation is depicted in Figure 6.2, where the ASV is represented in red, the UAV in green and the oil spill in black. In the figure is also possible to observe, in real time, the orientation of the ASV's sprinkler, represented as a three dimensional white arrow and the ASV waypoints generated using the Method III, represented as a two dimensional blue arrows.

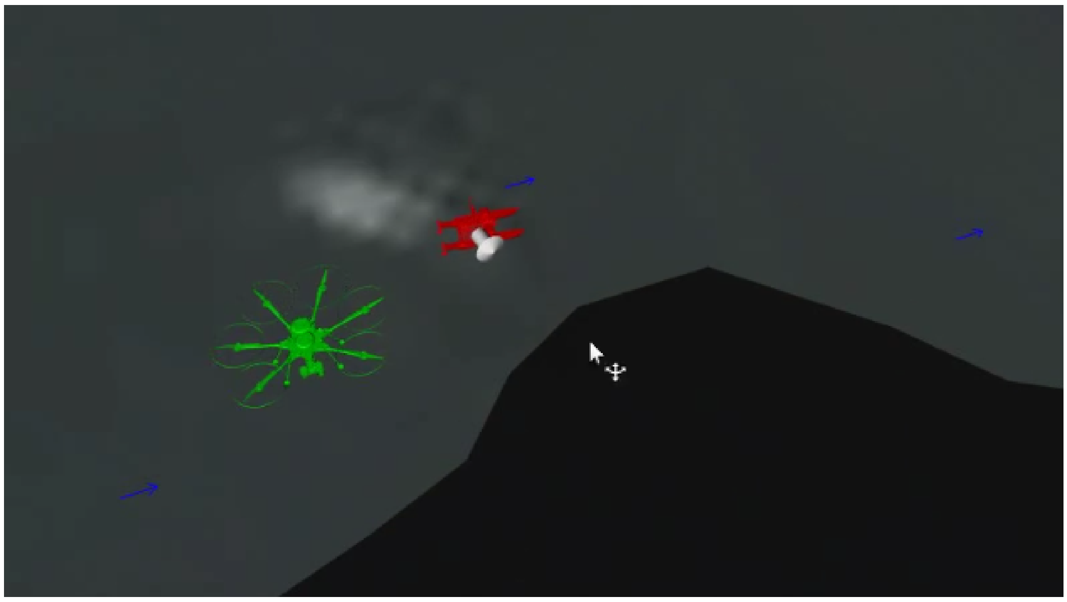


Figure 6.2: RViz representation of the scenario and vehicles position.

6.1 UAV's trajectory

Prior to the UAV's trajectory start, the container for the Lyophilized powder was loaded with a pink powder, as depicted in Figure 6.3, in order to visualise the release system in operation, this pink powder has a similar consistency to the Lyophilized powder. The powder releasing is noticeable in Figure 6.4.

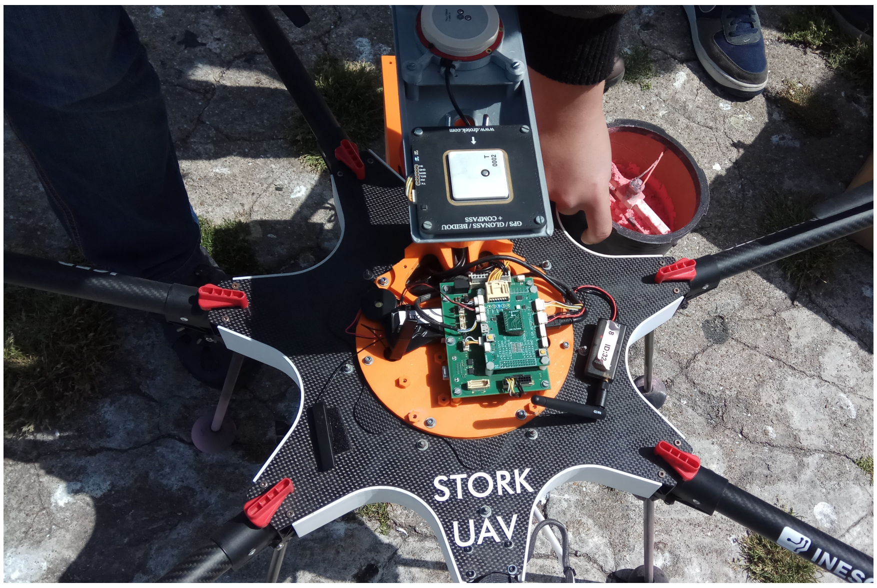


Figure 6.3: Powder loading into the UAV's container.

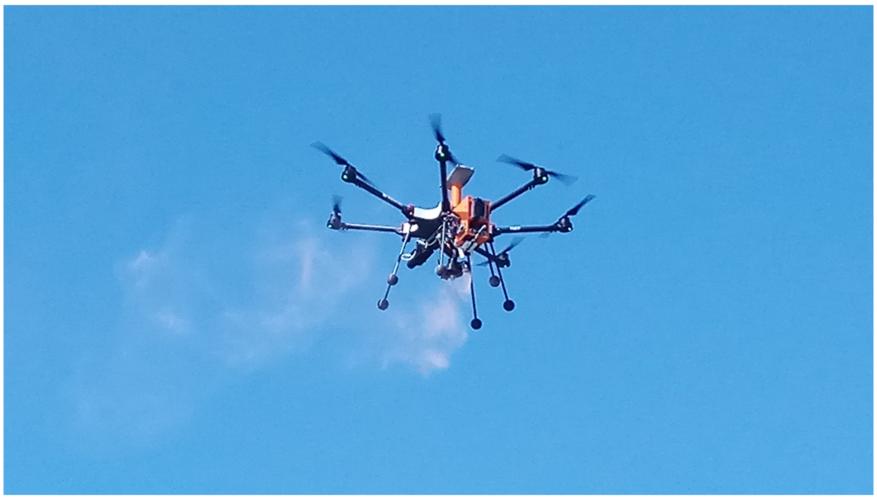


Figure 6.4: UAV's powder releasing system in action.

After the powder was loaded into the UAV, the trajectory waypoints were generated through the simulation environment and loaded into the UAV's mission planner. The computed trajectory was formed by 18 waypoints that describe an efficient "zigzag" manoeuvre for the oil spill mitigation. At that moment, the vehicle took of for an autonomous waypoint following flight.

From the data obtained from the PX4 autopilot logs, from the performed trajectory, it was possible to obtain the global estimated position from the UAV's on board GPS and IMU. The representation of that estimated trajectory is depicted in Figure 6.5. In the Figure 6.6 is possible to perceive in addition, other information like the intended trajectory, the estimated trajectory based only on the GPS and the trajectory waypoints. Both representations were obtained using the online software PX4 Flight Review[1].

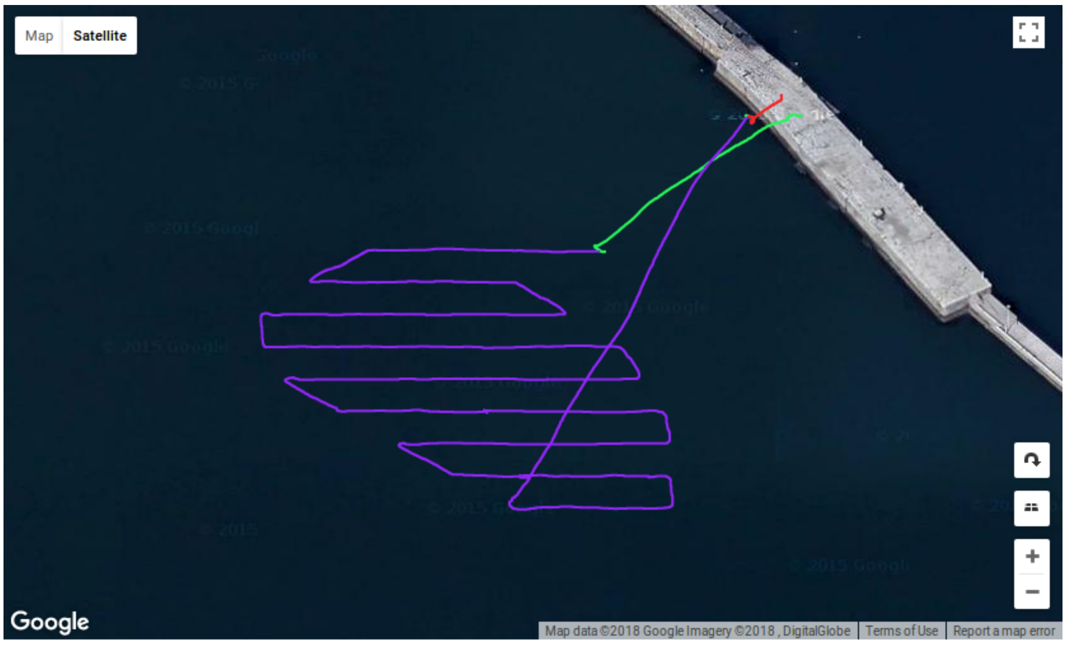


Figure 6.5: UAV's estimated trajectory representation.

Depicted in the Figure 6.7 is an overlap of the UAV's waypoints (intended trajectory), in green, and UAV's performed trajectory, in blue. For a more comprehensible representation, the detected oil spill contour was also included in the figure, in red. The representation was obtained using the online software GPS Visualizer[2].

[1]https://review.px4.io/
[2]http://www.gpsvisualizer.com/

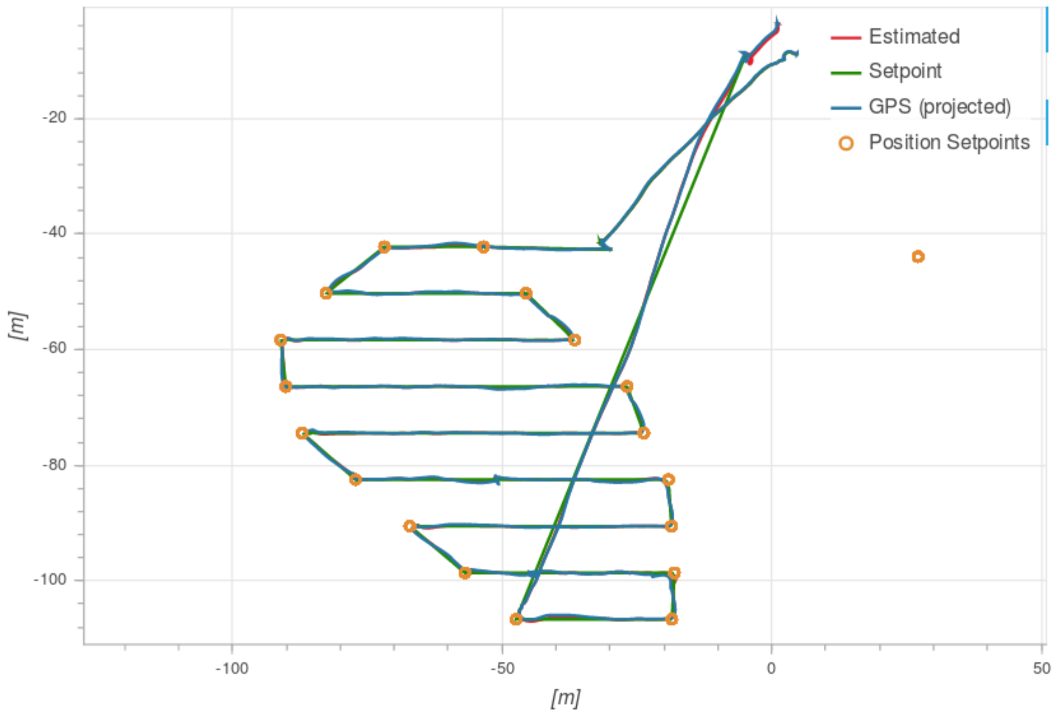


Figure 6.6: Representation of the data obtained from the UAV's PX4 autopilot.

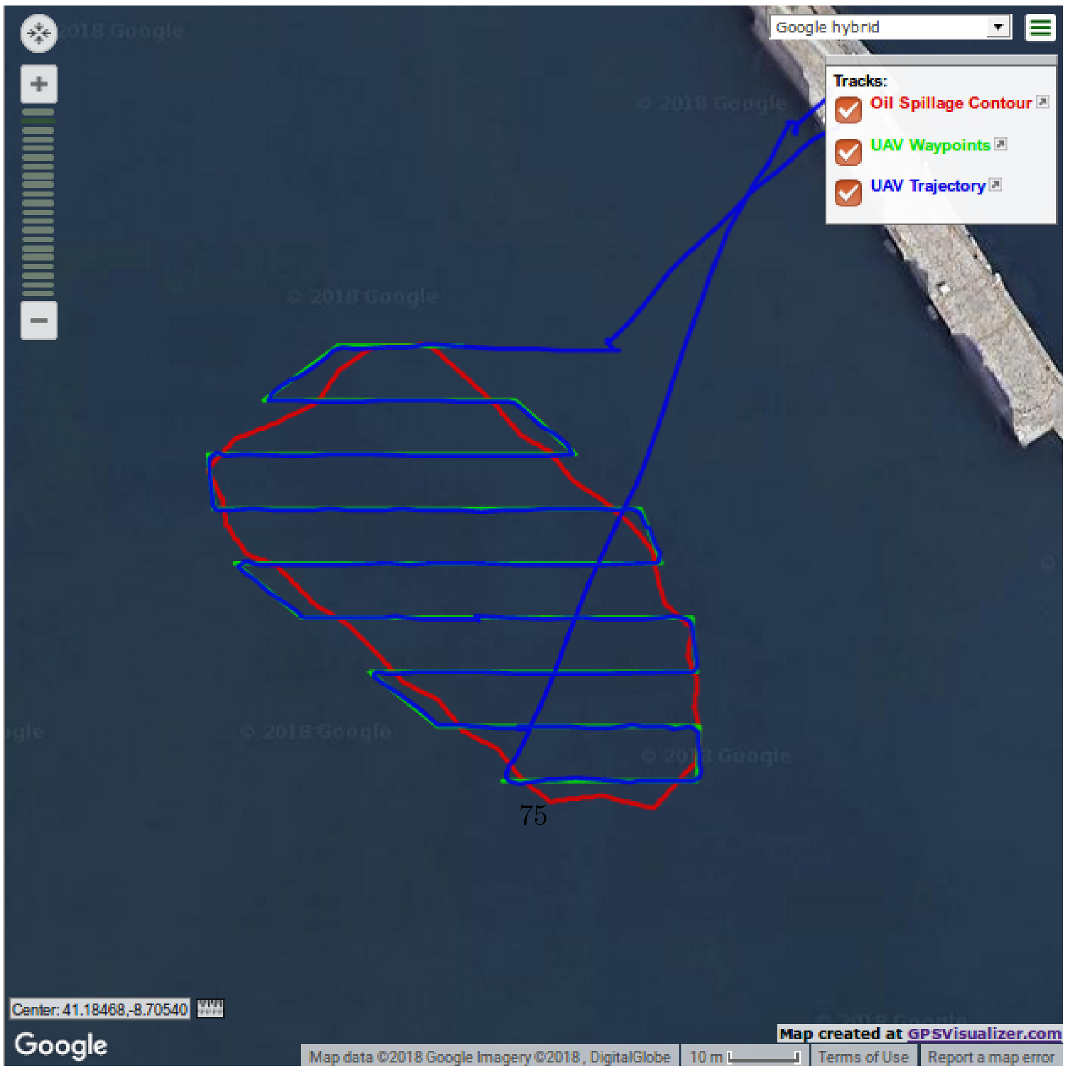

Google hybrid
Tracks:
Oil Spillage Contour
UAV Waypoints
UAV Trajectory
Center: 41.18468,-8.70540
Google
Map created at GPSVisualizer.com
Map data ©2018 Google Imagery ©2018 , DigitalGlobe
10 m
Terms of Use
Report a map error

Figure 6.7: UAV trajectory representation from the PX4 global position logs.

6.2 ASV's trajectory

The sprinkler implemented into the ASV is visible in action in the Figure 6.8. A Dynamixel servo motor was fixed to the ASV's sprinkler, to allow the sprinkler to to vary its angle of attack to the oil spill in a constant motion.

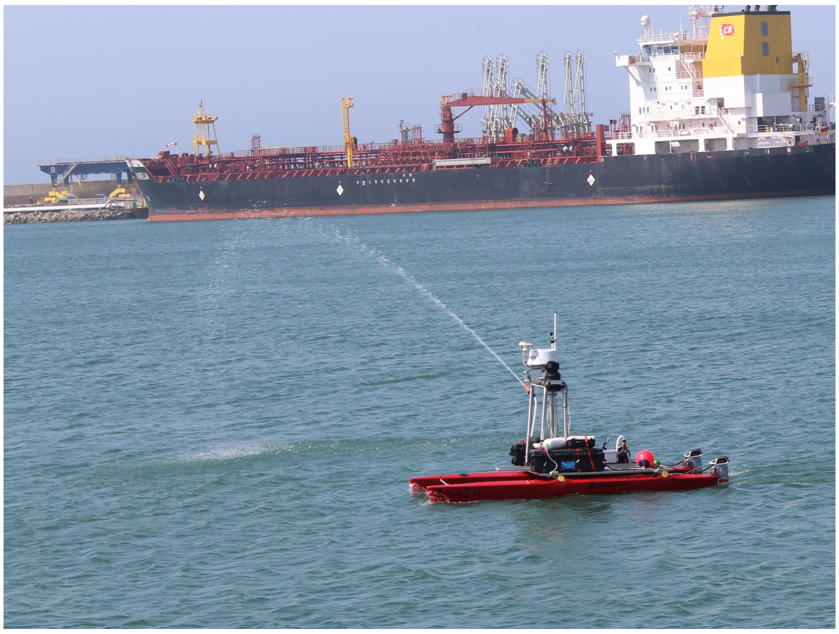


Figure 6.8: ASV sprinkler in action.

The ASV used for the experimental field tests, ROAZ II, has its own control base-station, depicted in Figure 6.9, from where is possible to manually control it, define and send waypoints for an autonomous waypoint following behaviour and the capability of monitor all the data provided by ROAZ II such as sensor data, vehicle position and trajectory, waypoints reached and next waypoint to be reached.

In the Figure 6.10 from a base station screenshot, is possible to identify the vehicle's current position by the red arrow, its performed trajectory by the red dots, the reached waypoints, from 1 to 8, as green triangles and the waypoint 9 still to be reached, in yellow. All these positions are presented on a realistic map. These 9 waypoints, represent the oil spill contour manoeuvre. The manoeuvre is represented by this low number of waypoints due too the minimal distance of 30 meters between each waypoint, this condition is imposed by the reduced turning angle of this specific surface vehicle. This distance between the waypoints can be easily altered in a parameter of the algorithm.

From the GPS attached to the ASV it was possible to represent the trajectories

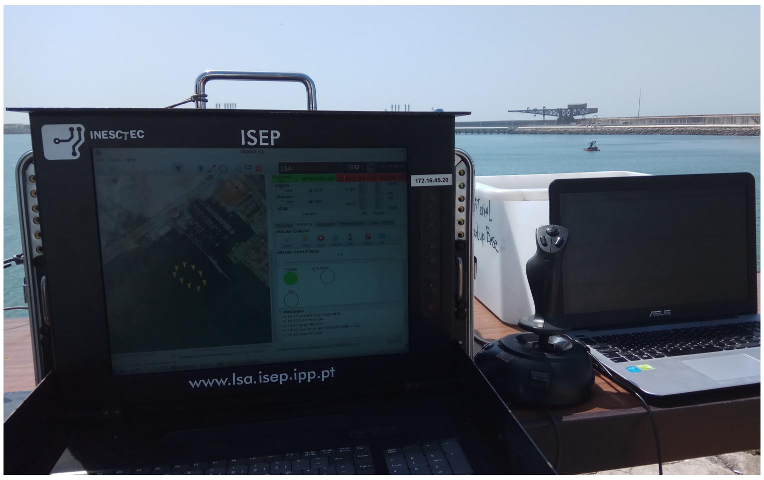


Figure 6.9: ASV ROAZ II field base station.

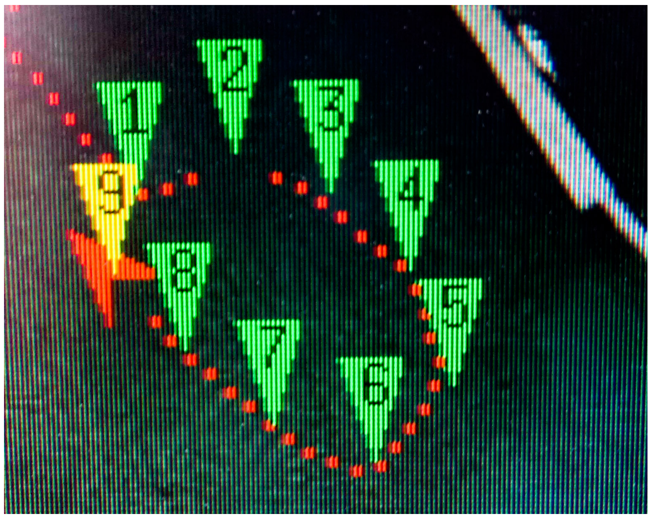


Figure 6.10: ASV trajectory representation in the basestation.

performed by ROAZ II. In the Figures 6.11 and 6.12, those trajectories are represented in blue. This manoeuvres were performed on two distinct days, 23 and 24 of April of 2018, respectively, with a constant acceptance radius of 10 meters. It is noticeable a larger drift from the intended trajectory in the Figure 6.12, this was due to harsh weather conditions, strong wind and current. Both representations were obtained using the online software GPS Visualizer.

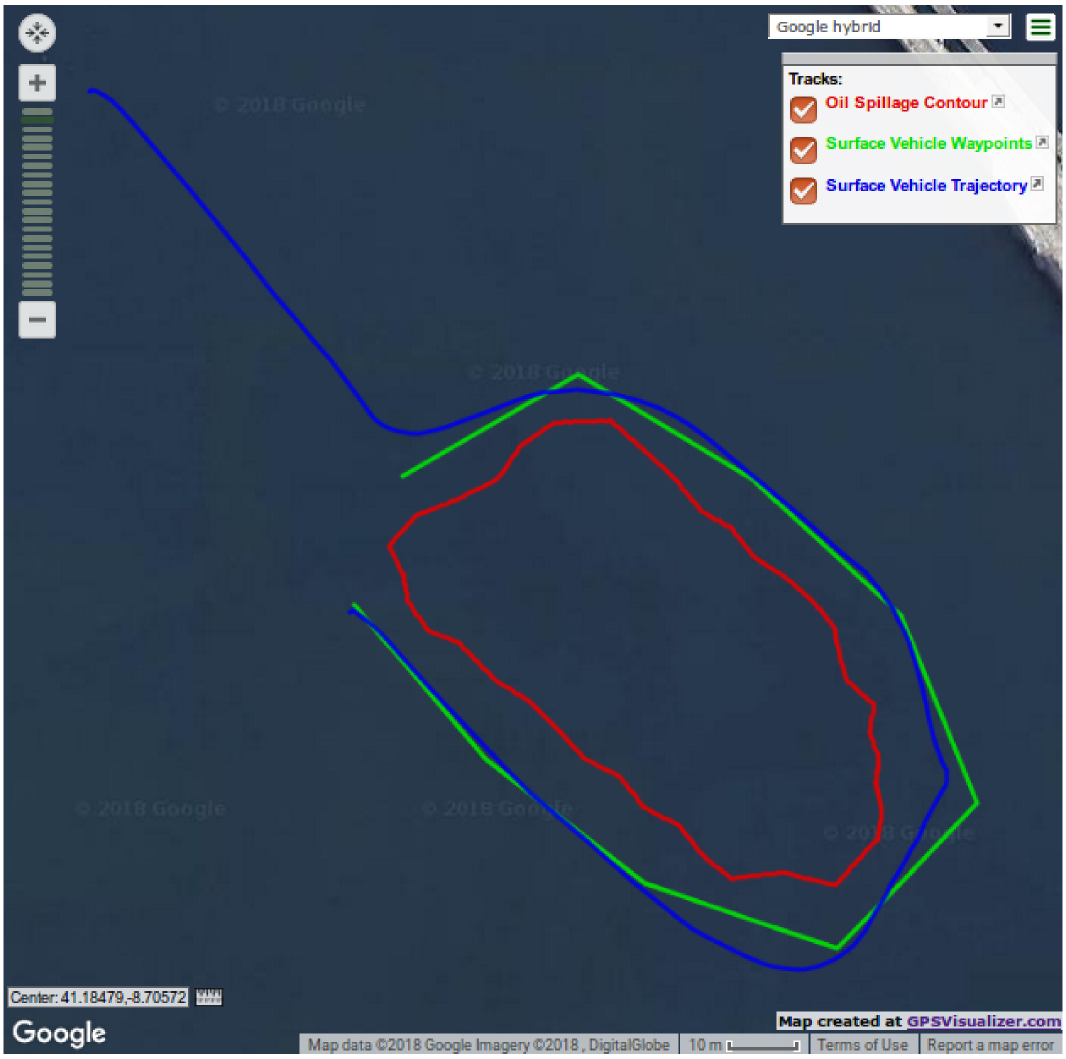

Google hybrid
Tracks:
Oil Spillage Contour
Surface Vehicle Waypoints
Surface Vehicle Trajectory
Center: 41.18479,-8.70572
Google
Map created at GPSVisualizer.com
Map data ©2018 Google Imagery ©2018 , DigitalGlobe
10 m
Terms of Use
Report a map error

Figure 6.11: ASV trajectory on April 23.

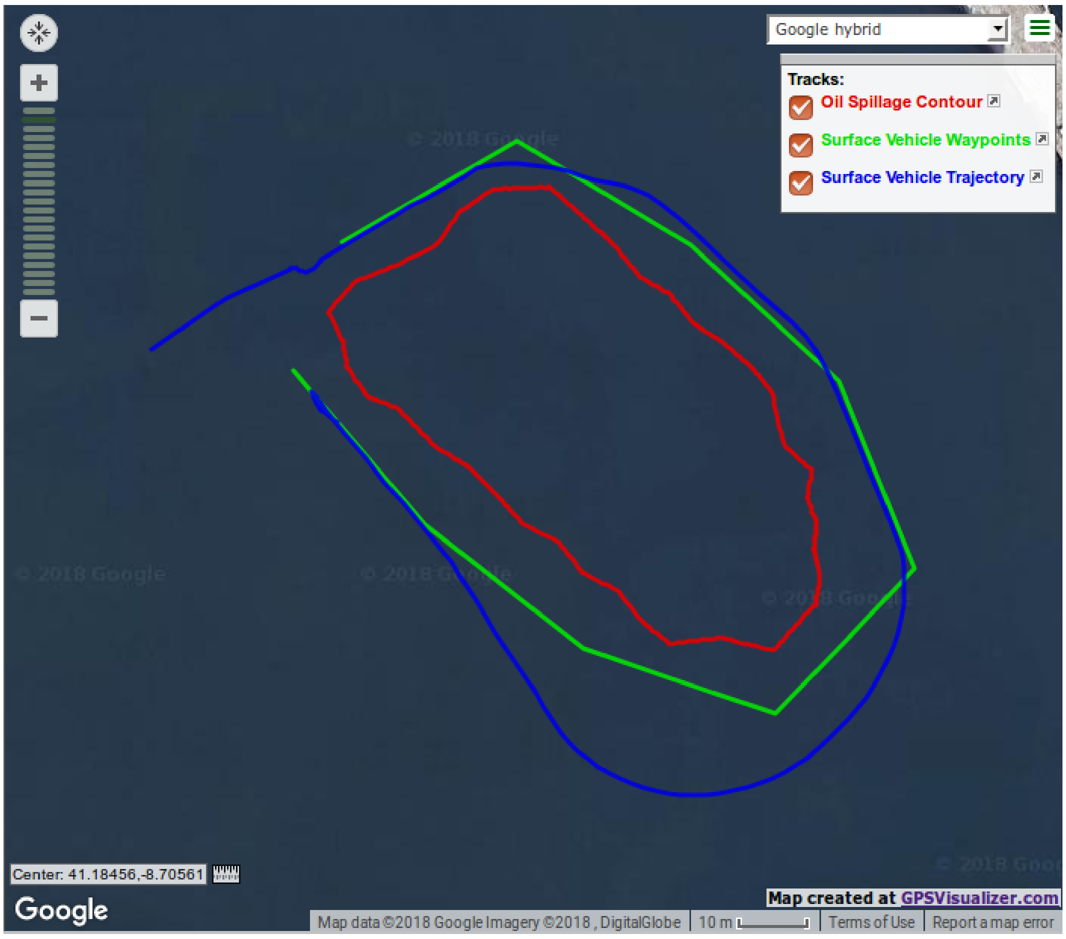

Google hybrid
Tracks:
Oil Spillage Contour
Surface Vehicle Waypoints
Surface Vehicle Trajectory
Center: 41.18456,-8.70561
Google
Map created at GPSVisualizer.com
Map data ©2018 Google Imagery ©2018 , DigitalGlobe
10 m
Terms of Use
Report a map error

Figure 6.12: ASV trajectory on April 24.

It is noticeable that the intended trajectory does not follow closely the oil spill borders during the entire manoeuvre, as explained before, this reduction of the number of trajectory waypoints was necessary due to the reduced turning angle of this specific surface vehicle.

This page was intentionally left blank.

Chapter 7

Conclusion and Future Work

In this dissertation the simulation and experimental results for the control-laws for the oil spill mitigation resorting to an UAV and an ASV are presented.

For the scenario previously described, from the approaches tested for the surface vehicle, it became clear that, the method able of obtain the best trajectory, for the oil spill mitigation, was the Method III, that uses normal vectors to compute a new and enlarged contour, the control points from that new contour are then passed through an Artificial Potential Field algorithm that uses them as successive goals. This method demonstrated to be the easiest to define a precise distance, to be maintained in relation to the oil spill boundaries, along the entire trajectory and did not present trajectory stops unlike the other algorithms that suffer from "*local minimas*".

However, in the real-world applications, the autonomous vehicles must carry out the clean-up tasks in more complicated scenarios, such as obstacle-rich environments that contain islands and other cleaning vessels. To mitigate these and other adversities, the following tasks are proposed as future work:

- Further improve the oil spill detection, discarding false positives from obstacles, personnel rescue teams or oil spill mitigation vessels;
- Control the displacement or the spread of the oil spill by the action of the wind/currents and its degradation from the mitigation process, either by realistic weathering models or through an aerial view of the environment during the entire process;
- Convert the UAV and ASV path planning methods into adaptive algorithms, capable of handling the displacement, spread or degradation of the oil spill;
- Modify the UAV path planning algorithm to ensure a mitigation action prioritisation over the areas where the ASV has already taken action, thereby ensuring a

more effective action.

- Continue improving the performance of the control law defined in this dissertation as Method III.
- Replace the 8 interchangeable goals defined in the Method I for N interchangeable goals, this N value keeps increasing while "*local minimas*" still exist in the planned trajectory.
- Increasing the ASV sprinkler's reach so that the vehicle do not need to follow the spill boundaries so closely.

A part of the work developed in this dissertation resulted in the publication of the paper "Control-law for Oil Spill Mitigation with an Autonomous Surface Vehicle" on the conference OCEANS'18 MTS/IEEE Kobe / Techno-Ocean 2018.

Bibliography

[1] Hani Safadi. Local path planning using virtual potential field. *McGill University School of Computer Science, Tech. Rep*, 2007.

[2] Clearpath Robotics et al. Ros 101: Intro to the robot operating system— robohub. *Robohub. org. Np*, 2017.

[3] RyuWoon Jung TaeHoon Lim YoonSeok Pyo, HanCheol Cho. *ROS Robot Programming.* ROBOTIS Co.,Ltd, 2017.

[4] Peter Burgherr. In-depth analysis of accidental oil spills from tankers in the context of global spill trends from all sources. *Journal of hazardous materials*, 140(1-2):245–256, 2007.

[5] Keisha Huijer. Trends in oil spills from tanker ships 1995-2004. *International Tanker Owners Pollution Federation (ITOPF), London*, 30, 2005.

[6] Xin Jin and Asok Ray. Navigation of autonomous vehicles for oil spill cleaning in dynamic and uncertain environments. *International Journal of Control*, 87(4):787–801, 2014.

[7] James G Speight and Karuna K Arjoon. *Bioremediation of petroleum and petroleum products.* John Wiley & Sons, 2012.

[8] Mark F Kirby and Robin J Law. Accidental spills at sea–risk, impact, mitigation and the need for co-ordinated post-incident monitoring. *Marine pollution bulletin*, 60(6):797–803, 2010.

[9] Sharon Gaudin. MIT builds swimming, oil-eating robots. *Computerworld, August*, 26, 2010.

[10] E. Gernez, C. M. Harada, R. Bootsman, Z. Chaczko, G. Levine, and P. Keen. Protei open source sailing drones: A platform for education in ocean exploration

and conservation. In *2012 International Conference on Information Technology Based Higher Education and Training (ITHET)*, pages 1–7, June 2012.

[11] A Pandey, S Pandey, and DR Parhi. Mobile robot navigation and obstacle avoidance techniques: A review. *Int Rob Auto J*, 2(3):00022, 2017.

[12] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, 61(12):1258–1276, 2013.

[13] Liang Yang, Juntong Qi, Jizhong Xiao, and Xia Yong. A literature review of uav 3d path planning. In *Intelligent Control and Automation (WCICA), 2014 11th World Congress on*, pages 2376–2381. IEEE, 2014.

[14] Tiago Fernandes. Planeamento de Trajetória para Operações de Busca e Salvamento com UAVs. ISEP, 2016.

[15] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[16] Shaojie Shen, Nathan Michael, and Vijay Kumar. Autonomous multi-floor indoor navigation with a computationally constrained mav. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 20–25. IEEE, 2011.

[17] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[18] Ivin Amri Musliman, Alias Abdul Rahman, Volker Coors, et al. Implementing 3d network analysis in 3d gis. *International archives of ISPRS*, 37(part B), 2008.

[19] Luca De Filippis, Giorgio Guglieri, and Fulvia Quagliotti. Path planning strategies for uavs in 3d environments. *Journal of Intelligent & Robotic Systems*, 65(1-4):247–264, 2012.

[20] Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard. A fully autonomous indoor quadrotor. *IEEE Transactions on Robotics*, 28(1):90–100, 2012.

[21] Sungsik Huh, David Hyunchul Shim, and Jonghyuk Kim. Integrated navigation system using camera and gimbaled laser scanner for indoor and outdoor autonomous flight of uavs. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3158–3163. IEEE, 2013.

[22] Neda Shahidi, Hadi Esmaeilzadeh, Marziye Abdollahi, and Caro Lucas. Memetic algorithm based path planning for a mobile robot. In *International Conference on Computational Intelligence*. Citeseer, 2004.

[23] Ellips Masehian and MR Amin-Naseri. A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. *Journal of Field Robotics*, 21(6):275–300, 2004.

[24] Ivan Maza and Anibal Ollero. Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6*, pages 221–230. Springer, 2007.

[25] M. Schwager, B. J. Julian, M. Angermann, and D. Rus. Eyes in the sky: Decentralized control for the deployment of robotic camera networks. *Proceedings of the IEEE*, 99(9):1541–1561, Sept 2011.

[26] H. K. Heidarsson and G. S. Sukhatme. Obstacle detection and avoidance for an autonomous surface vehicle using a profiling sonar. In *2011 IEEE International Conference on Robotics and Automation*, pages 731–736, May 2011.

[27] UCG Commandant. International regulations for prevention of collisions at sea, 1972 (72 colregs). *US Department of Transportation, US Coast Guard, COMMANDANT INSTRUCTION M*, 16672, 1999.

[28] S Campbell, Wasif Naeem, and George 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.

[29] Wasif Naeem, George W Irwin, and Aolei Yang. Colregs-based collision avoidance strategies for unmanned surface vehicles. *Mechatronics*, 22(6):669–678, 2012.

[30] Sable Campbell, Mamun Abu-Tair, and Wasif Naeem. An automatic colregs-compliant obstacle avoidance system for an unmanned surface vehicle. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 228(2):108–121, 2014.

[31] Liang Hu, Wasif Naeem, Eshan Rajabally, Graham Watson, Terry Mills, Zakirul Bhuiyan, and Ivor Salter. Colregs-compliant path planning for autonomous surface vehicles: A multiobjective optimization approach. *IFAC-PapersOnLine*, 50(1):13662–13667, 2017.

[32] Michael Benjamin, Joseph Curcio, and John Leonard. Navigation of unmanned marine vehicles in accordance with the rules of the road. Technical report, NAVAL SEA SYSTEMS COMMAND NEWPORT DIV RI, 2006.

[33] M. Pinto, B. Ferreira, H. Sobreira, A. Matos, and N. Cruz. Spline navigation and reactive collision avoidance with colregs for asvs. In *2013 OCEANS - San Diego*, pages 1–9, Sept 2013.

[34] Merv Fingas and Carl Brown. Review of oil spill remote sensing. *Marine pollution bulletin*, 83(1):9–23, 2014.

[35] Difeng Wang, Fang Gong, Delu Pan, Zengzhou Hao, and Qiankun Zhu. Introduction to the airborne marine surveillance platform and its application to water quality monitoring in china. *Acta Oceanologica Sinica*, 29(2):33–39, 2010.

[36] T Puestow, L Parsons, I Zakharov, N Cater, P Bobby, M Fuglem, G Parr, A Jayasiri, S Warren, and G Warbanski. Oil spill detection and mapping in low visibility and ice: Surface remote sensing. *Arctic Oil Spill Response Technology Joint Industry Programme (JIP)*, 2013.

[37] DF Dickins and JH Andersen. Evaluation of airborne remote sensing systems for oil in ice detection. *SINTEF Oil-in-Ice final report, Trondheim*, (28), 2010.

[38] GL Hover and JV Plourde. Evaluation of night capable sensors for the detection of oil on water. Technical report, COAST GUARD WASHINGTON DC OFFICE OF RESEARCH AND DEVELOPMENT, 1994.

[39] Carl E Brown, Richard Marois, Gregory E Myslicki, Mervin F Fingas, and Ron C Mackay. Remote detection of submerged orimulsion with a range-gated laser fluorosensor. In *International Oil Spill Conference*, volume 2003, pages 779–784. American Petroleum Institute, 2003.

[40] Deqing Liu, Xiaoning Luan, Feng Zhang, Jiucai Jin, Jinjia Guo, and Ronger Zheng. An usv-based laser fluorosensor for oil spill detection. In *Sensing Technology (ICST), 2016 10th International Conference on*, pages 1–4. IEEE, 2016.

[41] Pablo Marzialetti and Giovanni Laneve. Oil spill monitoring on water surfaces by radar l, c and x band sar imagery: A comparison of relevant characteristics. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 7715–7717. IEEE, 2016.

[42] G Aalet Mastin, JJ Manson, JD Bradley, RM Axline, and GL Hover. A comparative evaluation of sar and slar. Technical report, Sandia National Labs., Albuquerque, NM (United States), 1993.

[43] Nikolaos P Ventikos, Emmanouil Vergetis, Harilaos N Psaraftis, and George Triantafyllou. A high-level synthesis of oil spill response equipment and countermeasures. *Journal of hazardous materials*, 107(1-2):51–58, 2004.

[44] Elon Rimon and Daniel E Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on robotics and automation*, 8(5):501–518, 1992.

[45] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404. IEEE, 1991.

[46] J-O Kim and Pradeep K Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3):338–349, 1992.

[47] Hiroaki Seki, Yoshitsugu Kamiya, and Masatoshi Hikizu. Real-time obstacle avoidance using potential field for a nonholonomic vehicle. In *Factory Automation*. InTech, 2010.

[48] Vladimir J Lumelsky and Alexander A Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1):403–430, 1987.

[49] Ji Yeong Lee and Howie Choset. Sensor-based exploration for convex bodies: A new roadmap for a convex-shaped robot. *IEEE Transactions on Robotics*, 21(2):240–247, 2005.

[50] Zvi Shiller. Online suboptimal obstacle avoidance. *The International Journal of Robotics Research*, 19(5):480–497, 2000.

[51] Stefan Escaida Navarro, Stefan Koch, and Bjórn Hein. 3d contour following for a cylindrical end-effector using capacitive proximity sensors. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 82–89. IEEE, 2016.

[52] Urbano Nunes, Pedro Faia, and Anibal T de Almeida. Sensor-based 3-d autonomous contour-following control. In *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 1, pages 172–179. IEEE, 1994.

[53] Fumin Zhang, Alan O'Connor, Derek Luebke, and PS Krishnaprasad. Experimental study of curvature-based control laws for obstacle avoidance. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3849–3854. IEEE, 2004.

[54] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, April 2004.

[55] F. Abbasi, A. Mesbahi, and J. M. Velni. Coverage control of moving sensor networks with multiple regions of interest. In *2017 American Control Conference (ACC)*, pages 3587–3592, May 2017.

[56] S. Chen, C. Li, and S. Zhuo. A distributed coverage algorithm for multi-uav with average voronoi partition. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 1083–1086, Oct 2017.

[57] Mac Schwager, Daniela Rus, and Jean-Jacques Slotine. Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. *The International Journal of Robotics Research*, 30(3):371–383, 2011.

[58] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500–505. IEEE, 1985.

[59] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.

[60] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[61] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[62] Allen Miu. Lecture 7: Voronoi diagrams in computational geometry, September 2001.

[63] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

[64] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[65] Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote, and Séverin Lemaignan. Modular open robots simulation engine: Morse. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 46–51. IEEE, 2011.

[66] Jeff Craighead, Robin Murphy, Jenny Burke, and Brian Goldiez. A survey of commercial & open source unmanned vehicle simulators. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 852–857. IEEE, 2007.

[67] Adam Harris and James M Conrad. Survey of popular robotics simulators, frameworks, and toolkits. In *Southeastcon, 2011 Proceedings of IEEE*, pages 243–249. IEEE, 2011.

[68] Patricio Castillo-Pizarro, Tomás V Arredondo, and Miguel Torres-Torriti. Introductory survey to open-source mobile robot simulation software. In *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American*, pages 150–155. IEEE, 2010.

[69] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.

[70] Brian Gerkey, Richard T Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics*, volume 1, pages 317–323, 2003.

[71] Stefano Carpin, Mike Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. Usarsim: a robot simulator for research and education. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1400–1405. IEEE, 2007.

[72] Mario Prats, Javier Pérez, J Javier Fernández, and Pedro J Sanz. An open source tool for simulation and supervision of underwater intervention missions. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2577–2582. IEEE, 2012.

[73] E. Rohmer, S. P. N. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326, Nov 2013.

[74] Olivier Kermorgant. A dynamic simulator for underwater vehicle-manipulators. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 25–36. Springer, 2014.

[75] Kevin J DeMarco, Michael E West, and Ayanna M Howard. A simulator for underwater human-robot interaction scenarios. In *2013 OCEANS-San Diego*, pages 1–10. IEEE, 2013.

[76] Thomas Tosik and Erik Maehle. Mars: A simulation environment for marine robotics. In *Oceans-St. John's, 2014*, pages 1–7. IEEE, 2014.

[77] Musa Morena Marcusso Manhães, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*. IEEE, sep 2016.

[78] Andreas Birk, Gianluca Antonelli, Andrea Caiti, Giuseppe Casalino, Giovanni Indiveri, Antonio Pascoal, and Andrea Caffaz. The co 3 auvs (cooperative cognitive control for autonomous underwater vehicles) project: overview and current progresses. In *OCEANS 2011 IEEE-Spain*, pages 1–10. IEEE, 2011.

[79] Pedro J Sanz, Mario Prats, Pere Ridao, David Ribas, Gabriel Oliver, and Alberto Ortiz. Recent progress in the rauvi project: A reconfigurable autonomous underwater vehicle for intervention. In *Elmar, 2010 Proceedings*, pages 471–474. IEEE, 2010.

[80] Pedro J Sanz, Pere Ridao, Gabriel Oliver, Giuseppe Casalino, Yvan Petillot, Carlos Silvestre, Claudio Melchiorri, and Alessio Turetta. Trident an european project targeted to increase the autonomy levels for underwater intervention missions. In *Oceans-San Diego, 2013*, pages 1–10. IEEE, 2013.

[81] Daniel Cook, Andrew Vardy, and Ron Lewis. A survey of auv and robot simulators for multi-vehicle operations. In *Autonomous Underwater Vehicles (AUV), 2014 IEEE/OES*, pages 1–8. IEEE, 2014.

[82] J. J. Fernandez, J. Perez, A. Penalver, J. Sales, D. Fornas, and P. J. Sanz. Benchmarking using UWSim, Simurv and ROS: An autonomous free floating dredging intervention case study. *MTS/IEEE OCEANS 2015 - Genova: Discovering Sustainable Ocean Energy for a New World*, 2015.

[83] Jörg Kalwa, A Pascoal, Pere Ridao, A Birk, M Eichhorn, L Brignone, M Caccia, J Alvez, and RS Santos. The european r&d-project morph: Marine robotic systems of self-organizing, logically linked physical nodes. *IFAC Proceedings Volumes*, 45(5):349–354, 2012.

[84] David M Lane, Francesco Maurelli, Tom Larkworthy, Darwin Caldwell, Joaquim Salvi, Maria Fox, and Konstantinos Kyriakopoulos. Pandora: Persistent autonomy through learning, adaptation, observation and re-planning. *IFAC Proceedings Volumes*, 45(5):367–372, 2012.

[85] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. Yarp: yet another robot platform. *International Journal of Advanced Robotic Systems*, 3(1):8, 2006.

[86] Weijia Yao, Wei Dai, Junhao Xiao, Huimin Lu, and Zhiqiang Zheng. A simulation system based on ros and gazebo for robocup middle size league. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 54–59. IEEE, 2015.

[87] Thomio Watanabe, Gustavo Neves, Rômulo Cerqueira, Tiago Trocoli, Marco Reis, Sylvain Joyeux, and Jan Albiez. The rock-gazebo integration and a real-time auv simulation. In *Robotics Symposium (LARS) and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), 2015 12th Latin American*, pages 132–138. IEEE, 2015.

[88] Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote, and Séverin Lemaignan. Modular open robots simulation engine: MORSE. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 46–51, 2011.

[89] Haris Balta, Silvia Rossi, Salvatore Iengo, Bruno Siciliano, Alberto Finzi, and Geert De Cubber. Adaptive behavior-based control for robot navigation: A multi-robot case study. In *Information, Communication and Automation Technologies (ICAT), 2013 XXIV International Symposium on*, pages 1–7. IEEE, 2013.

[90] MAVROS (MAVLink on ROS) - PX4 Developer Guide. https://dev.px4.io/en/ros/mavros_installation.html. Accessed: 2018-06-16.