



Doctoral Thesis

ONLINE PATH PLANNING FOR AUTONOMOUS
UNDERWATER VEHICLES UNDER MOTION
CONSTRAINTS

JUAN DAVID HERNÁNDEZ VEGA

2017



Doctoral Thesis

ONLINE PATH PLANNING FOR AUTONOMOUS
UNDERWATER VEHICLES UNDER MOTION
CONSTRAINTS

JUAN DAVID HERNÁNDEZ VEGA

2017

Doctoral Program in Technology

Supervised by:
Dr. Marc Carreras

Thesis submitted to the University of Girona in fulfillment of the
requirements for the degree of Doctor of Philosophy

CERTIFICAT DE DIRECCIÓ DE TESI

Dr. Marc Carreras, del Departament d'Arquitectura i Tecnologia de Computadors de la Universitat de Girona,

DECLARO:

Que el treball titulat Online Path Planning for Autonomous Underwater Vehicles under Motion Constraints, que presenta Juan David Hernández Vega per a l'obtenció del títol de doctor, ha estat realitzat sota la meva direcció i que compleix els requisits per poder optar a Menció Internacional.

I, perquè així consti i tingui els efectes oportuns, signo aquest document.

Girona, Setembre 2017

Dr. Marc Carreras

ACKNOWLEDGMENTS

The work presented throughout this manuscript would not have been possible without the help of some incredible people. Their continuous support, from the beginning of this journey, till its end, helped me celebrate even the smallest achievements, as well as overcome the difficult moments any PhD thesis involves. I would like to give my gratitude to all of them.

En primer lugar me gustaría agradecer a mi familia en Armenia. A mi madre, por buscar y querer de mi siempre lo mejor; por su carácter y decisión para afrontar las dificultades de la vida; por sus sacrificios para ofrecerme las mejores oportunidades. A Agustín Escobar, por hacer de mi la persona y el ingeniero que soy, y por todos aquellos consejos que nunca olvidaré. Espero que este logro lo estés disfrutando desde el infinito. A mi hermana María Paula, por su alegría y su apoyo incondicional, por estar siempre ahí para mi sin importar la distancia. A Denisa, quien también ya hace parte de esta familia, gracias por apoyarme en los buenos y malos momentos de este doctorado, y por escuchar mis interminables conversaciones sobre mis experimentos y artículos.

También me gustaría agradecer a mi familia en Neiva. A mi padre, por sus llamadas y discusiones sobre la vida académica, por los buenos consejos y por escucharme en los momentos que he tenido que tomar decisiones importantes. A mis hermanos Daniela y Sergio Andrés, así como su madre Sonia Perdomo, gracias por la buena energía, y los buenos deseos. Han sido cortos y pocos los momentos que hemos podido compartir en los últimos años, pero gracias por estar siempre ahí.

Sin lugar a dudas, irme a Girona a hacer mi doctorado fue la mejor decisión profesional y experiencia personal que pude tener. Girona siempre será como mi segunda casa. Por esto me gustaría agradecer a mi tutor, quien desde un inicio me dio la posibilidad de preparar la propuesta de tesis con la que obtuve la beca de Colciencias, propuesta que se ha convertido ahora en una realidad. Gracias por el tiempo, las discusiones, las charlas de camino a Sant Feliu, y por siempre ser crítico y realista a la hora de definir los objetivos. Al mismo tiempo, me gustaría agradecer a Pere, por siempre estar allí para discutir temas adicionales de la tesis, los experimentos y los artículos. Siempre fue para mí como un co-tutor.

Un agradecimiento muy especial va para mis amigos y compañeros de trabajo en el CIRS y el Vicorob. Quisiera empezar con aquellos que estuvieron más cerca del desarrollo de esta tesis. Gracias a Enric por ser como un tutor y mentor durante mi primer año, por nuestras discusiones sobre path/motion planning, y por sus consejos para escribir artículos. A Eduard, Èric y Narcís quisiera agradecerles por siempre estar dispuestos a implementar, ajustar, mejorar la arquitectura del robot, o simplemente acompañarme durante los experimentos en el mar. A Carles y a Lluís, por tener los robots siempre a punto, solucionar los problemas técnicos, y por darme siempre consejos prácticos de mis experimentos. Gracias a mis compañeros de doctorado con los que compartí mis “deadlines”, viajes y competiciones, Arnau, Guillem, Albert, Jep, y Klemen. Gracias también a

los miembros seniors, Aggelos, Tali (en especial por darle revisiones adicionales a mis papers, abstracts, y pósters), Ricard, David y Nuno. Quisiera agradecer además al área administrativa del ViCOROB. Joseta, Mireia, Anna, Olga y Bego, que siempre solucionaron todos los problemas con la Universidad, las conferencias, los viajes y demás.

I would also like to thank my foreigner friends who have been part of ViCOROB. Francesco, Mojdeh and Guillaume, Sonia, Habib (and Sarah), Konstatin (and Amanda), Shihav, and Sharad- I will never forget our amazing multicultural dinners. I learned so much from all of you. And since the group was ever-changing, also its latest members: Dina, Khadidja, Klemen, Richa, and Patryk.

One of the most important and decisive experiences during my PhD was without a doubt my visit to Rice University in Houston. I would especially like to thank Lydia Kavraki and Mark Moll, my advisors there. Their time, dedication and contributions to this thesis are invaluable. I would also like to thank all the people at the Kavraki Lab: Morteza, Ryan, Didier, Dinler, Jayvee, Sarah, and Keliang. Your advice and comments during our meetings not only helped me in my work but also made my stay there an incredible experience.

Por último me gustaría agradecer a las fuentes de financiamiento de mis estudios, experimentos y publicaciones. Al Departamento Administrativo de Ciencia, Tecnología e Innovación (Colciencias), de Colombia, que me otorgó la beca de estudios doctorales. También me gustaría agradecer a la Unión Europea, a la Generalitat de Catalunya y al Gobierno Español, que por medio de proyectos financiaron parte de mi investigación.

PUBLICATIONS

The work developed in this thesis led to the following publications:

JOURNAL ARTICLES

[JFR'17] **J. D. Hernández**, E. Vidal, K. Istenič, M. Moll, M. Carreras, and L. E. Kavraki, "Online Path Planning for Unexplored Underwater Environments using AUVs that Operate under Motion Constraints," *Journal of Field Robotics*, (in preparation) 2017.

[SENSORS'16] **J. D. Hernández**, K. Istenič, N. Gracias, N. Palomeras, R. Campos, E. Vidal, R. García, and M. Carreras, "Autonomous Underwater Navigation and Optical Mapping in Unknown Natural Environments," *Sensors*, vol. 16, no. 8, p. 1174. July 2016.

PEER-REVIEWED CONFERENCES AND WORKSHOPS

[IROS'16] **J. D. Hernández**, M. Moll, E. Vidal, M. Carreras and L. E. Kavraki, "Planning Feasible and Safe Paths Online for Autonomous Underwater Vehicles in Unknown Environments," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), (Daejeon, Korea), pp. 1313–1320. October 2016.

[ROBOT'15] **J. D. Hernández**, K. Istenič, N. Gracias, R. García, P. Ridao, and M. Carreras, "Autonomous Seabed Inspection for Environmental Monitoring," ROBOT2015: Second Iberian Robotics Conference, (Lisbon, Portugal). Springer International Publishing, pp. 27–39. November 2015.

[ICRA'15] **J. D. Hernández**, E. Vidal, G. Vallicrosa, E. Galceran and M. Carreras, "Online Path Planning for Autonomous Underwater Vehicles in Unknown Environments," IEEE International Conference on Robotics and Automation (ICRA), (Seattle, USA), pp. 1152-1157. May 2015.

[NGCUV'15] **J. D. Hernández**, G. Vallicrosa, E. Vidal, E. Pairet, M. Carreras, and P. Ridao. "Online 3D Path Planning for Close-proximity Surveying with AUVs," IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV), Girona. April 2015.

OTHER CONFERENCES AND WORKSHOPS

[OCEANS'17] **J. D. Hernández**, E. Vidal, J. Greer, R. Piasco, P. Jausaud, M. Carreras, and R. García, "AUV Online Mission Replanning for Gap Filling and Target Inspection," MTS/IEEE OCEANS, (Aberdeen, Scotland). (to appear, 2017).

[AUV'16] M. Carreras, **J. D. Hernández**, E. Vidal, N. Palomeras, and P. Ridao, "Online Motion Planning for Underwater Inspection," IEEE/OES Autonomous Underwater Vehicles (AUV), (Tokyo, Japan), pp. 336-341. November 2016.

[OCEANS'15] **J. D. Hernández**, E. Vidal, G. Vallicrosa, E. Pairet, and M. Carreras. "Simultaneous Mapping and Planning for Autonomous Underwater Vehicles in Unknown Environments," IEEE OCEANS, (Genova, Italy). May 2015.

CONTRIBUTIONS TO OTHER WORKS

[RA-LETTERS'17] E. Vidal, **J. D. Hernández**, K. Istenič, and M. Carreras, "Online View Planning for Inspecting Unexplored Underwater Structures," IEEE Robotics and Automation Letters, vol. 2, no. 3, p. 1436 - 1443. February 2017.

[ICRA'18] E. Pairet, **J. D. Hernández**, M. Lahijanian, and M. Carreras, "Uncertainty-based Online Mapping and Motion Planning for Marine Robotics Guidance," IEEE International Conference on Robotics and Automation (ICRA), (in preparation) 2018.

LIST OF FIGURES

Figure 1	VICTOR 6000: a remotely operated vehicle (ROV) and its surface vessel.	1
Figure 2	Underwater glider and its sinusoidal-like vertical motion.	2
Figure 3	Sparus II: a torpedo-shaped and propeller-driven autonomous underwater vehicle (AUV).	3
Figure 4	The basic path/motion planning problem and the C-Space concept.	9
Figure 5	Reactive and sensor-based method Bug1.	10
Figure 6	Reactive and sensor-based method Bug2.	11
Figure 7	Reactive and sensor-based method Tangent Bug.	11
Figure 8	Grid-based method: coarse and fine grid.	12
Figure 9	Comparison between Dijkstra's algorithm and A*.	14
Figure 10	Example of potential field.	16
Figure 11	Example of a visibility graph.	17
Figure 12	Example of a voronoi diagram.	18
Figure 13	Example of a probabilistic roadmap (PRM) used to solve a start-to-goal query.	21
Figure 14	Example of a rapidly-exploring random tree (RRT).	24
Figure 15	Example of an asymptotic optimal rapidly-exploring random tree (RRT*).	25
Figure 16	Comparison between start-to-goal query solutions under geometric and differential constraints over a 2D workspace.	27
Figure 17	Coverage path planning (CPP) applications.	30
Figure 18	Sensor-based planning for AUVs under motion constraints.	31
Figure 19	FM-based planning for AUVs under motion constraints.	32
Figure 20	Combination of a grid-based planning method and a local planner to include the AUV motion constraints.	33
Figure 21	Top view of the Sparus II AUV, including the inertial and body-fixed frames, and the 2D vehicle state and control variables.	36
Figure 22	Control input selection for expanding an RRT under the differential equation.	38
Figure 23	Start-to-goal query solution by expanding an RRT algorithm that considers the differential constraints of a car-like system.	39
Figure 24	Examples of Dubins curves.	40
Figure 25	Start-to-goal query solution by expanding an RRT algorithm with Dubins curves.	41
Figure 26	RRT* algorithm reconnection using Dubins curves as steering function.	42

Figure 27	Comparison between start-to-goal query solutions obtained by an RRT and an RRT* algorithms over a 2D workspace.	43
Figure 28	Simulation of the Sparus II AUV attempting to follow a solution path calculated by an RRT* under geometric constraints.	44
Figure 29	Simulation of the Sparus II AUV attempting to follow a solution path calculated by an RRT and RRT* under motion constraints.	45
Figure 30	Perspective view of the Sparus II AUV, including the inertial and body-fixed frames, and the 6D vehicle state and control variables.	47
Figure 31	3D Start-to-goal query that requires a straight descent trajectory.	51
Figure 32	3D Start-to-goal query that requires forward and vertical motion.	52
Figure 33	Simulation of the Sparus II AUV attempting to follow a 3D solution path calculated by an RRT* under geometric constraints.	53
Figure 34	3D Start-to-goal query that requires circumnavigating while descending to reach the desired depth and orientation.	54
Figure 35	Simulation of the Sparus II AUV attempting to follow a 3D solution path calculated by an RRT* that uses the extended Dubins curves approach.	55
Figure 36	Simulation of the Sparus II AUV attempting to follow a 3D solution path in a high-relief environment. The path is calculated by an RRT* that uses the extended Dubins curves approach.	56
Figure 37	Framework for online AUV motion planning.	57
Figure 38	Harbor of Sant Feliu de Guíxols (Spain). Aerial view of the breakwater structure, and a representation of the area with an Octomap.	58
Figure 39	Collision checking using Octomaps.	59
Figure 40	A start-to-goal query solution using an RRT* algorithm with Dubins curves.	60
Figure 41	Sparus II AUV navigating between two obstacles and a visual representation of the risk zones around the vehicle.	61
Figure 42	Start-to-goal queries' solutions obtained by RRT* with Dubins curves. Optimization criteria comparison between path length and risk zones.	62
Figure 43	Sparus II AUV navigating between two obstacles and a visual representation of the risk vectors zones around the vehicle.	62
Figure 44	Extension of risk zones and risk vectors for 3D motions.	63
Figure 45	Opportunistic state checking. An important number of configurations are located in unexplored regions.	64

Figure 46	Sparus II AUV conducting a mission in the simulated breakwater structure using the pruning tree approach.	66
Figure 47	Reuse of the last best known solution.	67
Figure 48	Satellite image of the breakwater structure that is composed of a series of concrete blocks.	69
Figure 49	Start-to-goal queries' solutions using an RRT* algorithm with Dubins curves and risk functions. The solution paths maintain a safe distance from nearby obstacles.	70
Figure 50	Average computation time required by the planner when using different alternatives to estimate the risk associated with a path.	70
Figure 51	Sparus II solving a start-to-goal query in an equivalent virtual scenario of the breakwater structure without an initial map.	72
Figure 52	Incidence of <i>Opportunistic State Checking</i> approach when solving a start-to-goal query.	72
Figure 53	Comparison of the <i>anytime</i> computation approaches when solving start-to-goal queries without an initial map.	73
Figure 54	Comparison of the <i>anytime</i> computation approaches when solving start-to-goal queries without an initial map over a simulated natural-like environment.	74
Figure 55	Real-world trial to compare the <i>anytime</i> computation approaches when solving start-to-goal queries without an initial map. The test scenario was the breakwater structure.	75
Figure 56	Test scenario of an artificial marine structure.	77
Figure 57	Sparus II AUV is teleoperated at surface in the breakwater structure scenario.	78
Figure 58	Exteroceptive sensors configuration for the Sparus II AUV to conduct autonomous missions in the breakwater structure.	79
Figure 59	The Sparus II AUV using the path/motion planning framework to navigate through a breakwater structure without a preliminary map of the surroundings.	81
Figure 60	3D reconstruction built from images gathered along the autonomous mission presented in Fig. 59.	82
Figure 61	Test scenario of a natural marine formation.	83
Figure 62	Exteroceptive sensors configuration for the Sparus II AUV to conduct autonomous missions in the underwater canyon.	83
Figure 63	The Sparus II AUV using the path/motion planning framework to navigate through an underwater canyon without a preliminary map of the surroundings.	84
Figure 64	3D reconstruction built from images gathered along the autonomous mission presented in Fig. 63.	85

Figure 65	Comparison between the vehicle trajectory estimated by the DR system and the one estimated with the cameras' images.	86
Figure 66	Satellite image of the underwater caves complex "Coves de Cala Viuda".	87
Figure 67	The underwater caves complex "Coves de Cala Viuda". Meshed map constructed using the SNDC algorithm.	88
Figure 68	The underwater caves complex "Coves de Cala Viuda" added into UWSim as a 3D simulation environment.	88
Figure 69	Simulated mission in the caves complex. First and second start-to-goal queries.	89
Figure 70	Simulated mission in the caves complex. Third start-to-goal query.	90
Figure 71	Simulated mission in the caves complex. Fourth and fifth start-to-goal queries.	91
Figure 72	Simulated mission in the caves complex. Whole mission execution.	92
Figure 73	Main modules of the <i>backseat</i> controller.	94
Figure 74	AsterX AUV deployment from its mother ship.	95
Figure 75	Potential target and gaps detected after completing the initial survey with the AsterX AUV.	96
Figure 76	Path planned to further cover the potential target and gaps with the AsterX AUV.	96
Figure 77	The AsterX AUV approaches to further inspect a potential target.	97
Figure 78	The AsterX AUV completed the mission by inspecting the potential target and covering the gaps from the initial survey.	97
Figure 79	Sparus II AUV: top, bottom, lateral and 3D views, and its hardware elements.	105
Figure 80	AsterX AUV at sea surface and its lateral view with a description of the different hardware elements.	106
Figure 81	AsterX AUV software architecture.	107

LIST OF TABLES

Table 1	Path/motion planning methods.	34
Table 2	Comparison of the <i>anytime</i> computation approaches when solving start-to-goal queries without an initial map. The test scenario was a simulated version of a breakwater structure	74

Table 3	Comparison of the <i>anytime</i> computation approaches when solving start-to-goal queries without an initial map. The test scenario was a simulated natural-like environment.	75
---------	--	----

LIST OF ALGORITHMS

1	A*	14
2	Gradient Descent	16
3	Visibility Graph	18
4	PRM, preprocessing phase	20
5	sampleRRT	23
6	extendRRT	23
7	extendRRT*	25
8	extendRRT when dealing with motion constraints.	37
9	SampleAnytimeRRT*	65
10	incSolveStart2Goal($q_{\text{start}}, q_{\text{goal}}$)	68

ACRONYMS

C-Space	configuration space
ACA	Ariadne's clew algorithm
RPP	randomized path planner
PRM	probabilistic roadmap
RM	roadmap
RRT	rapidly-exploring random tree
RRT*	asymptotic optimal RRT
T-RRT	transition-based RRT
CL-RRT	closed-loop RRT
GVD	generalized Voronoi diagram
CPP	coverage path planning
AFP	artificial potential field
FADPRM	flexible anytime dynamic PRM
D*	dynamic A*
AD*	anytime dynamic A*
ARA*	anytime repairing A*
2D	2-dimensional
3D	3-dimensional
6D	6-dimensional
EST	expansive-spaces tree
FM	fast marching
OMPL	open motion planning library
FCL	flexible collision library
WP	waypoint
B-spline	basis spline
HLP	high-level planner
LLP	low-level planner
DFS	depth-first search

DOF	degrees of freedom
ROS	robot operating system
SLAM	simultaneous localization and mapping
UAV	unmanned aerial vehicle
GA	genetic algorithm
DR	dead-reckoning
AUV	autonomous underwater vehicle
UUV	unmanned underwater vehicle
ROV	remotely operated vehicle
DVL	doppler velocity log
IMU	inertial measurement unit
NED	north-east-depth
USBL	ultra-short baseline
CIRS	Underwater Vision and Robotics Research Center
ViCOROB	Computer Vision and Robotics Institute
COLA ₂	component oriented layer-based architecture for autonomy
UWSim	underwater simulator
Ifremer	French Research Institute for Exploitation of the Sea
ACE	automated control engine
ISE	International Submarine Engineering
SDNC	splats distance normalized cut

CONTENTS

1	INTRODUCTION	1
1.1	Motivation	1
1.2	Problem Statement	2
1.2.1	Navigating in Unexplored Environments with AUVs	3
1.2.2	Path/Motion Planning Problem for AUVs	4
1.3	Objectives of the Thesis	6
1.4	Thesis Outline and Contributions	6
2	STATE OF THE ART	8
2.1	Planning Collision-free Paths over the C-Space	8
2.2	Bug-based Methods	9
2.3	Search-based (Grid-based) Path/Motion Planning	12
2.3.1	Dijkstra's Algorithm	12
2.3.2	A*	13
2.3.3	Dynamic A* (D*)	15
2.4	Potential Fields	15
2.5	Roadmaps	16
2.5.1	Visibility Graphs	17
2.5.2	Generalized Voronoi Diagrams	17
2.6	Sampling-based Algorithms	18
2.6.1	Multiple-query Methods	19
2.6.2	Single-query Methods	21
2.6.3	Optimal Planning	23
2.7	Extensions and Applications	24
2.7.1	Planning Feasible Paths	26
2.7.2	Online Path/Motion Planning	27
2.8	Path/Motion Planning for AUVs	29
2.8.1	Start-to-goal Path/Motion Planning for AUVs	30
2.8.2	Online Motion Planning for AUVs through Unexplored Environments	31
2.9	Summary	33
3	PLANNING CONSTANT-DEPTH PATHS UNDER AUV MOTION CONSTRAINTS	35
3.1	2D First-Order Motion Model	35
3.2	Motion Planning by Using Differential Equations	36
3.3	Motion Planning by Using Dubins Curves	39
3.4	Results	41
3.4.1	Conducting Missions under Geometric Constraints	43
3.4.2	Conducting Missions under Motion Constraints	44
4	PLANNING 3D MOTIONS FOR TORPEDO-SHAPED AND PROPELLER-DRIVEN AUVS	46
4.1	3D First-Order Motion Model	46
4.1.1	Vertical Motion Constraints	47
4.1.2	Mathematical Formulation	48
4.2	Tree Expansion using Dubins Curves for 3D Motions	49
4.3	Results	50

4.3.1	Conducting 3D Missions under Geometric Constraints	51
4.3.2	Conducting 3D Missions under Motion Constraints .	52
5	FRAMEWORK FOR ONLINE MOTION PLANNING IN UNEXPLORED ENVIRONMENTS	57
5.1	Mission Handler	57
5.2	Module for Incremental and Online Mapping	58
5.3	Module for (Re)Planning Paths Online	59
5.3.1	Planning Safe Paths using Risk Functions	59
5.3.2	Opportunistic State Checking	63
5.3.3	Reuse of the Last Best Known Solution	64
5.4	Pipeline for Online Planning Feasible and Safe Paths	67
5.5	Results	68
5.5.1	Comparison of Risk Functions	69
5.5.2	Opportunistic State Checking and its Benefits	71
5.5.3	Anytime Computation for Solving Start-to-goal Queries in Unexplored Environments	71
6	RESULTS IN REAL-WORLD SCENARIOS	76
6.1	Planning AUV Paths in Artificial Marine Structures	76
6.2	Planning AUV Paths in Natural Marine Formations	80
6.3	Planning AUV Paths in Confined Natural Environments . .	86
6.4	Gap Filling and Potential Target Inspection	93
7	CONCLUSIONS	98
7.1	Summary of Completed Work	98
7.2	Review of Contributions	100
7.3	Future Work	101
Appendix		103
A	EXPERIMENTAL PLATFORMS	104
A.1	Sparus II AUV	104
A.2	AsterX AUV	106
BIBLIOGRAPHY		108

ABSTRACT

Since their beginning in the late 1950s, the capabilities and applications of autonomous underwater vehicles ([AUVs](#)) have continuously evolved. Their most common applications include imaging and inspecting different kinds of structures on the sea floor as well as collecting oceanographic information: biological, chemical, and even archaeological data.

Most of these applications require *a priori* information of the area or structure to be inspected, either to navigate at a safe and conservative altitude or to pre-calculate a survey path. However there are other applications where it's unlikely that such information is available (e.g., exploring confined natural environments like underwater caves). In these scenarios, [AUVs](#) must operate in unexplored and cluttered environments, and therefore are more exposed to collisions.

Although these [AUV](#) applications share some common requirements with other aerial and terrestrial robots (e.g., localization, mapping, vision, etc.), they are also different in significant ways. Navigating autonomously while conducting these type of tasks in underwater environments demands taking into account factors such as: the presence of external disturbances (currents), low-range visibility and limited navigation accuracy. Dealing with such constraints requires a path planner with online capabilities that can overcome the lack of environment information and the global position inaccuracy, especially when navigating in close proximity to nearby obstacles.

In this respect, this thesis presents an approach that endows an [AUV](#) with the capabilities to move through unexplored environments. To do so, it proposes a computational framework for planning feasible and safe paths online. This approach allows the vehicle to incrementally build a map of the surroundings, while simultaneously (re)plan a feasible path to a specified goal. To accomplish this, the framework takes into account motion constraints in planning feasible 2D and 3D paths, i.e., those that meet the vehicle's motion capabilities. It also incorporates a risk function to avoid navigating close to nearby obstacles.

To evaluate the proposed approach in different real-world scenarios, a series of trials were conducted with the Sparus II and the AsterX [AUVs](#), torpedo-shaped vehicles that performed autonomous missions. These experiments include simulated and in-water trials in different environments, such as artificial marine structures, natural marine structures, and confined natural environments.

RESUMEN

Desde sus inicios a finales de los años 50, las capacidades y aplicaciones de los vehículos autónomos submarinos o AUVs (por sus siglas en inglés) han estado bajo un continuo proceso de evolución. Las aplicaciones más comunes incluyen la obtención de imágenes e inspección de diferentes tipos de estructuras, tales como cascos de barcos, estructuras naturales en el fondo marino, así como la recolección de información oceanográfica como datos biológicos, químicos e incluso arqueológicos.

Muchas de estas aplicaciones requieren información *a priori* del área o estructura que va a ser inspeccionada, ya sea para navegar a una altitud segura o para precalcular un camino para realizar estudios, el cual puede ser corregido o modificado en tiempo real. Sin embargo, existen aplicaciones similares o nuevas, como la exploración de entornos naturales confinados (e.g., cuevas submarinas), donde dicha información puede no estar disponible. En estos escenarios, los AUVs deben operar en entornos desconocidos, y por lo tanto los AUVs están más expuestos a colisiones.

Aunque estas aplicaciones de AUVs comparten algunos requerimientos comunes con otras aplicaciones de robots aéreos y terrestres (e.g., localización, mapeo, visión, etc.), navegar autónomamente mientras se ejecutan este tipo de tareas en entornos submarinos difiere en ciertos factores, como la presencia de perturbaciones externas (corrientes), bajo rango de visibilidad y limitaciones en la precisión del sistema de navegación. Para poder abordar dichas restricciones se requiere un planificador de movimientos con capacidad de cómputo en tiempo real, el que contribuya a superar las limitaciones en la información del entorno y la falta de precisión de posicionamiento, en especial cuando se navega cerca de los obstáculos de su alrededor.

En este sentido, esta tesis presenta una método para dotar un AUV con la habilidad para moverse a través de entornos no explorados. Para ello, esta tesis propone un método para calcular en tiempo real caminos factibles y seguros. El método propuesto permite al vehículo construir incrementalmente un mapa del entorno, y al mismo tiempo replanificar el camino factible hacia la meta u objetivo establecido. Para lograr esto, es necesario considerar las restricciones de movimiento para planificar caminos 2D y 3D que sean factibles o realizables.

Para evaluar el método propuesto, se realizaron diferentes experimentos con los AUVs Sparus II y AsterX, ambos vehículos tipo torpedo que realizaron misiones de manera autónoma en diferentes escenarios. Estos experimentos incluyen pruebas en simulación y en el agua en diferentes entornos, tales como estructuras marinas artificiales, estructuras marinas naturales, así entornos naturales confinados.

RESUM

Des dels inicis a finals dels anys 50, les capacitats i aplicacions dels vehicles autònoms submarins o AUVs (per les seves sigles en anglès) han experimentat un procés d'evolució continu. Les aplicacions més comunes són l'obtenció d'imatges i inspecció de diferents tipus d'estructures, com per exemple, cascós de vaixells o estructures naturals en el fons marí, i l'adquisició d'informació oceanogràfica com dades biològiques, químiques i arqueològiques.

Moltes d'aquestes aplicacions requereixen informació *a priori* de l'àrea o estructura que es vol inspeccionar, ja sigui per navegar a una altitud segura o per calcular en avançat un camí que permeti realitzar els estudis, el qual pot ser corregit o modificat a temps real. No obstant, existeixen aplicacions similars o noves, com l'exploració d'entorns naturals confinats (e.g., coves submarines), on aquesta informació pot ser inexistente. En aquests casos, els AUVs han d'operar en entorns desconeguts, pel que estan més exposats a col·lisions.

Tot i que aquestes aplicacions de AUVs comparteixen alguns requeriments comuns amb altres aplicacions de robots aeris i terrestres (e.g., localització, mapeig, visió, etc.), navegar autònomament al mateix temps que s'executen aquest tipus de tasques en entorns submarins difereix en certs factors, com la presència de pertorbacions externes (corrents), baix rang de visibilitat i limitacions en la precisió del sistema de navegació. Per poder tractar les esmentades restriccions és necessari un planificador de moviments amb capacitat de processament en temps real, el que ajuda a superar les limitacions en la informació de l'entorn i la falta de precisió de posicionament, en especial quan es navega a prop dels obstacles presents en el seu voltant.

En aquest sentit, aquesta tesi presenta una alternativa per dotar un AUV amb l'habilitat de moure's a través d'entorns no explorats. Per aconseguir aquesta fita, aquesta tesi proposa un mètode per calcular en temps real camins factibles i segurs. El mètode proposat permet al vehicle construir de forma incremental un mapa de l'entorn, i al mateix temps replanificar un camí factible cap a l'objectiu establert. Per assolir això, el mètode proposat té en compte les restriccions de moviment del vehicle per planificar camins 2D i 3D que siguin factibles o realitzables.

Per avaluar el mètode proposat, s'han realitzat diferents experiments amb els AUVs Sparus II i l'AsterX, els quals són vehicles de tipus torpede que van realitzar missions de forma autònoma en diferents escenaris. Aquests experiments inclouen proves en simulació i a l'aigua en diferents entorns, tal i com estructures marines artificials, estructures marines naturals i entorns naturals confinats.

INTRODUCTION

1.1 MOTIVATION

The need to use robotic systems in order to overcome human beings' limitations has been a topic of an ever increasing interest in the last decades. Manipulator arms, for instance, were introduced in the industry to optimize production lines by systematically executing tasks with a bounded margin of error and high repeatability during long working hours. Recently, such an interest in robots has veered towards autonomous vehicles that conduct missions in complex and undiscovered environments, such as deep oceans, volcanoes and even other planets. Consequently, various sectors of the scientific community started to research this area, especially focused on developing robotic vehicles that are capable of conducting tasks where data is gathered autonomously in different environmental media, such as air, soil and water [28].

Marine scientists were probably the first ones capitalizing on the use of robotic vehicles in exploring unknown environments. Oceanographers, for instance, started using unmanned underwater vehicles (UUVs) to study deep marine environments and the seafloor [127]. Nowadays, UUVs are separated into two categories according to the level of human intervention required for their operation. A first group includes the so-called remotely operated vehicles (ROVs) which, despite being unmanned, have to be tethered to a support surface vessel from which they are powered and controlled by a human operator. Although ROVs are commonly associated with their use in inspecting and maintaining offshore oil and gas platforms [68], they have also contributed to different scientific applications such as environmental monitoring [79] and marine archeology [36]. Figure 1 shows the widely-known VICTOR 6000 ROV in a typical deployment maneuver from the mother ship, where the umbilical cable used to operate the vehicle can be clearly observed.

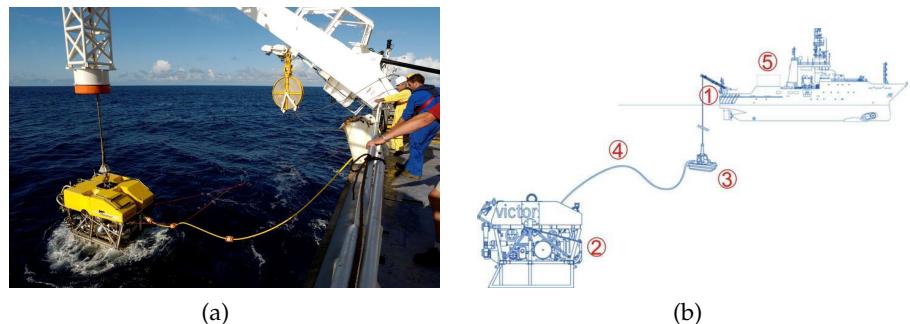


Figure 1: (a) The VICTOR 6000 ROV. (b) The ROV-vessel system: 1) a direct-winding winch, 2) the ROV, 3) the hard ballast, 4) the tether cable, and 5) the power/hydraulic units and control room. Image credit: Ifremer.

The second group gathers the autonomous underwater vehicles ([AUVs](#)), or [AUVs](#) that do not require being controlled by a human operator. This characteristic eliminates the need of continuously being in contact with a surface mother ship, thus permitting to overcome some major drawbacks of the former group, such as their high operative cost and limited working area. In most of the [AUV](#) applications, the vehicle follows a sequence of pre-calculated waypoints and uses its onboard sensors to gather oceanographic information of biological nature [43], chemical composition [126], and even archaeological data [11]. Often, they are also equipped with multibeam and imaging sonars that collect data that is used to build bathymetric maps (i.e., elevation maps of the seabed). These underwater surveys are normally conducted in a previously explored area so that the vehicle navigates at a constant and safe altitude from the seafloor.

Furthermore, [AUVs](#) can also be divided into two main subcategories: buoyancy-driven and propeller-driven. The former category corresponds to the underwater gliders, or [AUVs](#) that are capable of modifying its buoyancy in order to generate sinusoidal-like vertical motion that, together with their wings, also permit horizontal (forward and lateral) motion. The trajectories followed by these vehicles also include periodic ascents to the surface to obtain GPS fixes, thus improving its navigation estimation (see Fig. 2). This propulsion technique results in a low-consumption but also less maneuverable approach. This is commonly used in long-term studies over open sea areas, in which a single mission can last from days to weeks.

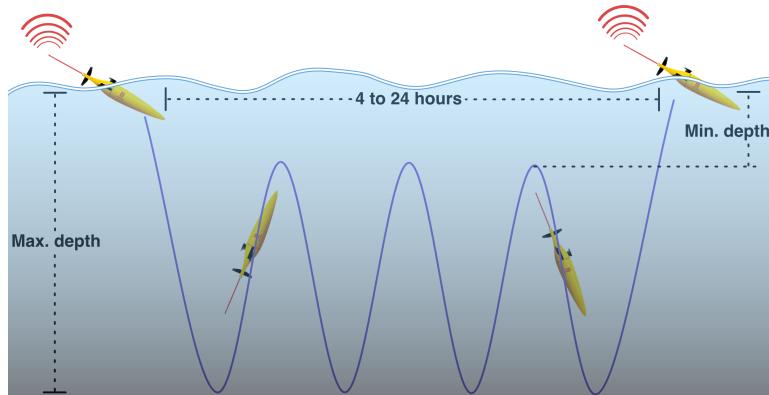


Figure 2: Underwater glider, or [AUV](#) with the capability of modifying its buoyancy to generate sinusoidal-like vertical motion. The vehicle dives for several hours to gather data, and surfaces periodically to obtain GPS fixes thus improving its own navigation estimation.

The propeller-driven [AUVs](#), on the other hand, correspond to those that use propellers or thrusters to generate 3-dimensional ([3D](#)) motions. The work presented throughout this thesis is dedicated to this latter kind of vehicles. Furthermore, it has been mainly validated with the Sparus II, a torpedo-shaped [AUV](#) (see Fig. 3).

1.2 PROBLEM STATEMENT

Recent [AUV](#) applications include imaging and inspecting different kinds of structures such as in-water ship hulls [53, 58] and natural structures on



Figure 3: Sparus II AUV, a torpedo-shaped and propeller-driven underwater vehicle, which is equipped with two back thrusters for horizontal motion, and one independent thruster for vertical motion.

the sea floor [42]. These applications share a common characteristic: they require *a priori* information of the area or structure to be inspected. Therefore it is needed to either navigate at a safe and conservative altitude [11, 43] or to pre-calculate a survey path that may be corrected or reshaped online [42, 53]. However, there are similar or even potentially new applications, such as exploring confined natural environments (e.g., underwater caves) [91], where such information might not be available. In these scenarios, the AUV must operate in unexplored (unknown), cluttered and dynamic environments, and therefore are more exposed to collisions.

Although the aforementioned AUV applications share some common requirements with aerial and terrestrial robots (e.g., localization, mapping, vision, etc.), navigating autonomously while conducting such type of tasks in an underwater environment presents different challenges. These are caused by factors specific to this environment, such as the presence of external disturbances (currents), low-range visibility and limited navigation accuracy. Dealing with such constraints requires a path planner with on-line computation capabilities, which contribute in overcoming the lack of surroundings information and the global position inaccuracy, especially when navigating in close proximity to nearby obstacles.

1.2.1 Navigating in Unexplored Environments with AUVs

Even with the most recent technological advances, conducting both tele-operated and autonomous missions in underwater environments still represents a challenge with different scientific aspects to be solved. One important restriction faced by UUVs is that they operate in GPS-denied environments, which limits their capability to accurately calculate their position with respect to an inertial reference frame. In order to cope with this, dead-reckoning (DR) techniques are used to estimate the relative position with respect to an initial position, which is normally defined from GPS fixes obtained when the vehicle is at surface. This approach suffices in many survey AUV applications. However, its limitations rely on highly precise odometry and low-level or negligible disturbances. When these conditions are not met, it can rapidly lead to errors in distance estimation, which make this approach an unsafe alternative when navigating and ex-

ploring surroundings in close proximity to nearby obstacles, even when the environment is known and a map is available.

Another alternative is the of use of an ultra-short baseline (**USBL**), which is a method of underwater acoustic positioning. An **USBL** system is composed of a transceiver mounted on a surface vessel and a transponder mounted on the **AUV**. This system permits calculating the range (distance) and the angle of the subsea vehicle with respect to the surface vehicle. Furthermore, given that the transceiver is at surface (i.e., it can get GPS fixes), it is possible to calculate a more accurate absolute position of the **AUV**. The major drawback of this alternative is the mobility constraint imposed by the transceiver, similar to what occurs with a **ROV** and its mother ship. This also prevents its use in confined natural environments such as an underwater caves complex, which results in a critical limitation for the new applications intended for **AUVs**.

Therefore, when planning collision-free **AUV** paths, the difficulty to pre-explore and build a map of the area or structure of interest is an additional challenge to be considered together with the navigation inaccuracy. In order to tackle these problems, this thesis proposes a framework that endows an **AUV** with the capability to map an undiscovered environment, while planning collision-free paths to navigate throughout it. Furthermore, although the framework considers specific characteristics of the motion planning problem for **AUVs**, the proposed approach may also be applied for other types of autonomous vehicles.

1.2.2 Path/Motion Planning Problem for AUVs

1.2.2.1 Overview of the Basic Robot Path/Motion Planning

In general, path/motion planning methods can be divided into two categories according to their application: coverage planning and start-to-goal planning. The former category, commonly referred to as coverage path planning (**CPP**), gathers those methods that seek to determine a collision-free path that a robot must follow in order to pass over all points of an area or volume of interest. A detailed review of available techniques that address this problem can be found in [39].

The second category, i.e., start-to-goal motion planning, consists in finding valid (collision-free) paths from a start configuration to a goal configuration in the configuration space (**C-Space**), which is the space of all the possible robot configurations [86]. There are different computational algorithms that attempt to solve this task. Some of them find an analytic solution, but their applicability is limited to simple problems. Another group of algorithms, such as Dijkstra's [23], A* [44] and D* [112], search throughout discretization of the **C-Space**. However, in problems involving high-dimensional **C-Spaces**, these exhaustive search methods suffer from scalability issues.

One alternative in dealing with this is to use the so-called sampling-based methods. These latter ones construct a partial representation by taking samples at random from the **C-Space** and checking them for collisions [20, 75], instead of fully describing the **C-Space**. In contrast with exhaustive search methods, these are not complete (i.e., they can not report

whether a solution exists), but often find a solution for complex problems where other algorithms fail.

There are additional characteristics or capabilities that should be considered when solving path/motion planning tasks for AUVs. Some of which will be mentioned in the following sections.

1.2.2.2 Planning under Geometric and Differential (Motion) Constraints

When calculating a valid path, the planner must deal with different types of constraints, some of which may be related only to the C-Space, while others may also include the vehicle motion capabilities. Depending on which constraints are considered, the planning problem was categorized in one of two possible groups: geometric path planning and motion planning [20]. The former group assumes that the robotic system can move instantaneously in any direction (i.e., system kinematics and dynamics are negligible), thus reducing the problem to geometric constraints that are solved by checking for collisions over the C-Space.

However, the motion constraints associated with the vehicle are often significant, and therefore the robot is not capable of following a simple geometric path. This means that, when calculating collision-free paths, it is necessary to use a vehicle motion model that includes differential constraints, also called kinodynamic constraints. Such a group of planning problems is commonly called motion planning or *kinodynamic motion planning* (this latter term was introduced by Donald et al. in 1993 [26]). An example of such planning problems is the one related to a non-holonomic torpedo-shaped AUV, as the ones used in this thesis (see Figure 3).

As recently the terms path planning and motion planning are used interchangeably [114], [75], throughout this thesis, we refer to the either problem as path/motion planning

1.2.2.3 3D Path/Motion Planning

In some AUV applications the vehicle either navigates at constant depth or attempts to keep reference values of altitude. In those cases, and specially in areas with no or little relief, it is possible to simplify the motion model to one that assumes the vertical position variation as constant or negligible. Nonetheless, when such conditions are not met, it is required to use more complete motion equations, which in the simplest case implies adding an additional state variable for the vertical motion, or in more complex scenarios may require to use a 6-dimensional (6D) state space, i.e., one that incorporates the position and the orientation of the vehicle in a 3D workspace, where each state or configuration is $\mathbf{q} = [x, y, z, \phi, \theta, \psi]$.

1.2.2.4 Online Path/Motion Planning

Another important aspect to be considered when planning motions for AUVs, especially when navigating in unexplored environments, is the necessity of calculating the paths (motions) online. To deal with such computation constraints, the planner must be capable of providing a solution at *any time* and of correcting (reshaping) it according to the updated environment information gathered along the mission execution. This characteristic

implies that although it is possible that the solution provided at *any time* may not be the most optimal one, any available time left will be spent on improving the existing solution path. This means that the next delivered solution is better than the previous one or at least equally good. Finally, it is also important to correctly balance the computation load to coexist with other functional modules of the AUV such as navigation, control, etc.

1.3 OBJECTIVES OF THE THESIS

Once the motivation and the most relevant aspects of the problem addressed in this thesis have been established, the main objective of this work can be stated as follows:

To endow an autonomous underwater vehicle (AUV) with the capability to incrementally map an unexplored environment, while planning online collision-free paths; such paths should be 3-dimensional (3D), safe, and feasible (doable) according to the vehicle's motion constraints. This will contribute a further step towards better and more reliable AUVs for the new and potential applications.

This objective can be separated into the following specific objectives:

Review of the Path/Motion Planning Literature: To conduct a review of the most relevant path/motion planning methods, which have been used for different robotic systems including terrestrial, aerial, and, especially, marine vehicles. This will allow identifying the most appropriate approaches for meeting the aforementioned constraints for the intended applications.

Planning Feasible Motions for AUVs: To propose a motion planning method that takes into account the AUV's motion constraints for both 2D and 3D movements. This will permit generating more feasible motions according to the AUV's capabilities. Furthermore, such motions will be more likely to be followed by the vehicle, thus minimizing unexpected maneuvers.

Online Mapping and Path/Motion Planning for AUVs: To propose a framework that allows an AUV to safely navigate through initially unexplored environments. This implies that the vehicle must be capable of incrementally mapping the surroundings while, simultaneously and online, planning safe and feasible paths.

Simulation and In-water Validation: To extensively test and validate the proposed approach in different simulated and real-world scenarios.

1.4 THESIS OUTLINE AND CONTRIBUTIONS

This manuscript is organized in a way that presents the incremental and progressive development of this thesis. In order to contextualize and better understand the real impact of such developments, Chapter 2 firstly makes a review of the most relevant methods, techniques, and applications that compose the state of the art, not only in what has to do with path/motion

planning, but also in relation with the approaches currently applied with **AUVs**. Then, based on the new and potential applications intended for **AUVs** that were mentioned above, and considering their requirements, the main contributions of this thesis are gathered and distributed throughout this document as follows:

- 1) The incorporation of motion constraints into the **AUV** path planner, which acts as a mechanism to avoid generating unfeasible paths, thus reducing the number of unexpected maneuvers. This includes those constraints required for constant depth missions that only take into account 2-dimensional (**2D**) horizontal motions, which are discussed in Chapter 3. But it includes as well more challenging and variable depth missions that require analysing and combining the vehicle turning and ascending/descending limitations, which are addressed in Chapter 4.
- 2) An efficient approach for approximating the risk associated with a path. There are situations in which considering the vehicle motion constraints does not avoid risky maneuvers, especially when navigating in close proximity to nearby obstacles. In such cases, the **AUV** path/motion planner must lead the vehicle to navigate at a safe and conservative distance from its surroundings, yet without discarding any possible solution. Different strategies to estimate the risk associated with a vehicle configuration, and therefore with a path, are presented and compared in Chapter 5.
- 3) The capability of an **AUV** to efficiently (re)plan feasible and safe paths, i.e., under motion constraints and taking into consideration the associated risk, in scenarios where the environment is incrementally discovered. This clearly means that the path/motion planner has to operate under online computation constraints, which can only be done by combining existing approaches and newly proposed strategies, such as *opportunistic state checking and reuse of the last best known solution*. The complete online planning approach will be presented in Chapter 5.
- 4) A framework that allows an **AUV** to incrementally map an environment undiscovered previously, while simultaneously (re)planning a path that must be followed by the vehicle to safely cross and explore the surroundings. This framework integrates all previous stages and characteristics, in order to finally endow the **AUV** with the required capabilities for the intended applications, and will also be presented in Chapter 5.
- 5) An extensive validation of the proposed framework is presented in Chapter 6. This includes simulated and in-water trials in different real-world scenarios, where an **AUV** had to navigate through initially unexplored environments, while gathering different kind of information such as acoustic and optical data. These tests were conducted with two different **AUVs**: the Sparus II from the University of Girona, and the AsterX from the French Research Institute for Exploitation of the Sea (**Ifremer**)¹. The results include planning safe and feasible paths to move through both artificial and natural marine structures, as well as the autonomous survey replanning for gap filling and target inspection.

Finally, concluding remarks and directions for further research are given and discussed in Chapter 7.

¹ Technical details about both vehicles can be found in Appendix A.

2

STATE OF THE ART

Although the major research efforts in this thesis were focused on developing, extending and using path/motion planning methods for AUV applications, their validation with an AUV in real-world scenarios involved other fields of study. These included but were not limited to: navigation, perception, mapping and control. An extensive presentation of the state-of-the-art background of all these areas would not only prove to be lengthy and diffuse, but would also be out of the scope of this work. Nonetheless, this chapter is dedicated to contextualizing the contribution of this thesis. In order to do so, it firstly provides a survey of the techniques available for solving specifically path/motion planning problems. Secondly, it discusses the most common extensions used when the vehicle motion capabilities have to be taken into consideration or when dealing with online computation constraints. Finally, it presents a review of the algorithms and extensions that have been used with AUVs particularly.

2.1 PLANNING COLLISION-FREE PATHS OVER THE C-SPACE

Even though first works on path/motion planning appeared in the late 60's [99], it was only until the 80's, when Lozano-Perez introduced the concept of the *configuration space* [86–88], that this field of study became active. The configuration space (or *C-Space*) establishes the set of all possible *configurations* that a robot can adopt when executing tasks in the workspace. While the workspace, \mathcal{W} , is typically defined as \mathbb{R}^n (where $n = 2, 3$ for 2D and 3D motion, respectively), the *C-Space*, \mathcal{C} , depends on the robot motion capabilities. For example, a robot considered to be a point that moves in a plane (i.e., $\mathcal{W} = \mathbb{R}^2$) requires two coordinates in order to specify its *configuration*, q , which is equal to $[x, y]^T$, thus $q \in \mathcal{C} = \mathbb{R}^2$.

However, if the robot is considered to be a rigid body that moves in a plane (i.e., in the same \mathcal{W}), it requires three coordinates; these would describe not only its position, but also its orientation, therefore its *configuration* is now $q = [x, y, \psi]^T$, which is commonly expressed as $q \in \mathcal{C} = \text{SE}(2) = \mathbb{R}^2 \times \text{SO}(2) = \mathbb{R}^2 \times \mathcal{S}$. A similar scenario occurs with a single rigid-body robot that operates in 3D workspaces, in which $q = [x, y, z]$, so that $q \in \mathcal{C} = \mathbb{R}^3$ when only the robot's position is considered, or $q = [x, y, z, \phi, \theta, \psi]$, so that $q \in \mathcal{C} = \text{SE}(3) = \mathbb{R}^3 \times \text{SO}(3)$ when considering both position and orientation. Finally, when the robot is an articulated rigid body system, for instance a manipulator arm, a given *configuration* is described by the values of a set of n generalized coordinates $q = q_1, \dots, q_n$, corresponding to each of the robotic arm degrees of freedom (*DOFs*).

The solution to a simple path/motion planning problem, which requires connecting a start and a goal configuration, q_{start} and q_{goal} , is a continuous path $p : [0, 1] \rightarrow \mathcal{C}$, such that $p(0) = q_{\text{start}}$ and $p(1) = q_{\text{goal}}$. However, a robot generally conducts tasks in environments that contain

obstacles, which normally are to be avoided. For this reason, the *C-Space* is subdivided into *free space* ($\mathcal{C}_{\text{free}}$) and the *obstacle region* (\mathcal{C}_{obs}), meaning that $\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}}$. Therefore, a collision-free path is defined as a continuous path $p : [0, 1] \rightarrow \mathcal{C}_{\text{free}}$. This concept is illustrated in Figure 4.

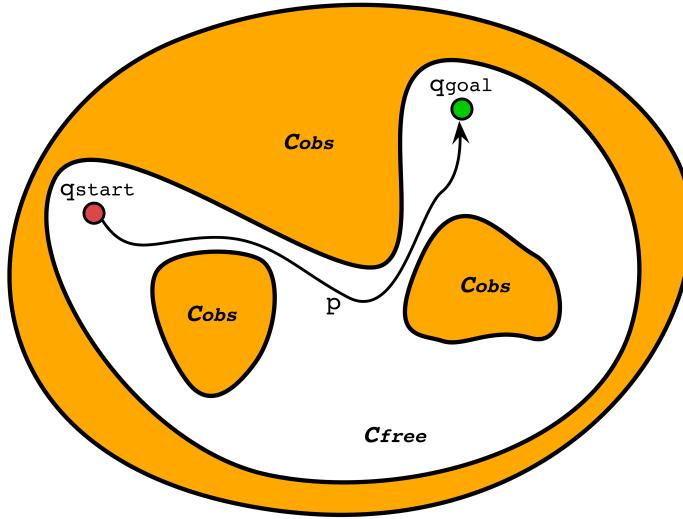


Figure 4: The basic path/motion planning problem seeks to connect a start configuration (q_{start}) and goal configuration (q_{goal}) with a continuous collision-free path $p : [0, 1] \rightarrow \mathcal{C}_{\text{free}}$.

2.2 BUG-BASED METHODS

Bug-based methods represent one of the earliest reactive and sensor-based path planning approaches, in which a mobile robot moves in the plane towards a global goal, while having a limited (local) knowledge of the environment. The algorithms included in this group are based on two basic behaviors: go straight towards the goal and follow the obstacles' boundary. Lumelsky and Stepanov presented the first of these algorithms, known as Bug1 and Bug2, that mainly rely on tactile or zero range sensors to perceive the obstacles [89]. Later, Kamon et al. introduced the Tangent Bug algorithm, which is an extension that uses non-zero range sensors [60]. All of them are classified as complete algorithms, which means that they find a solution when one is possible or, otherwise, they report when there is no solution. Furthermore, they assume the robot is a point that moves in a 2D workspace, and that it is capable of knowing its position and calculating its distance to the goal.

Bug1 [89] is the most basic algorithm that uses the aforementioned behaviors. With this method, the robot moves from the start position (q_{start}) towards the desired goal position (q_{goal}) until it reaches the goal or until it finds an obstacle (detected with its tactile sensors). When the latter situation occurs, the robot stores its current position and marks it as the *hit point* (H_i). Then it changes its motion mode (behavior) to completely circumnavigate the obstacle. While the vehicle follows the obstacle boundary, it continuously calculates the distance to the goal in order to determine the position that corresponds to the minimum possible distance. This is

marked as the *leave point* (L_i). Once the robot reaches the *hit point* again (i.e., after it has traveled all the obstacle contour), it goes back to the *leave point* and there changes its behavior, once again, in order to move towards the goal. This procedure is repeated until reaching the specified goal. Figure 5 depicts an example of this algorithm solving a start-to-goal query.

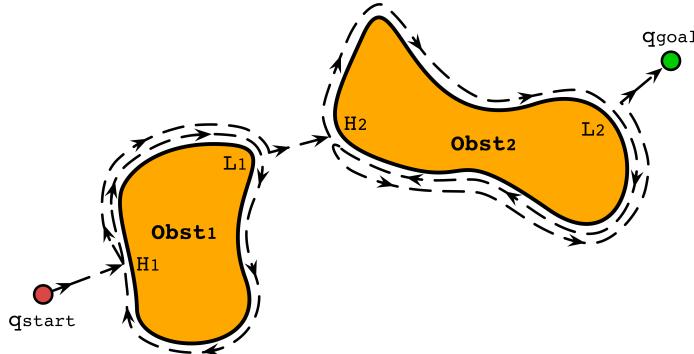


Figure 5: Reactive and sensor-based method Bug1

Bug2 [89] attempts to be an improved version of the previously explained algorithm. It establishes a straight line, sometimes referred as the *m-line* [20], which connects the start and goal positions. With this algorithm, the robot starts moving towards the desired goal by following the *m-line* until it reaches the goal or finds an obstacle. As occurs with Bug1, if the robot is dealing with an obstacle, it first marks that position as the *hit point* (H_i) and then changes its behavior to circumnavigate the obstacle. However, with Bug2 the robot does not necessarily travel the entire obstacle boundary, but instead stops when reaching another point in the *m-line*; if such a point is closer to the goal than the previous *hit point*, the robot marks this position as *leave point* (L_i) and, from there, continues following the *m-line* towards the goal. This procedure is repeated until reaching the specified goal. Even though Bug2 does not require to completely circumnavigate the obstacles, there are environments where the robots do require to travel the entire boundary; in these cases Bug2 may generate longer paths than those calculated by Bug1, and therefore is less efficient. Figure 6 depicts an example of Bug2 solving start-to-goal queries in both simple and complex scenarios.

Finally, the Tangent Bug algorithm [60] was proposed as an improved alternative for Bug1 and Bug2. In this version, the robot is assumed to be equipped with a non-zero range sensor, which permits detecting in advance not only the obstacles, but also their continuous boundaries. This way, when the robot is navigating towards the goal and faces an obstacle, it can determine the discontinuities in the boundaries or the limits of the perceived area, which are marked as *endpoints*. Then, in order to avoid the obstacle, the robot checks which of the *endpoints* minimizes the distance to the goal and starts moving towards it. The procedure is repeated until the obstacle is no longer perceived, at which point the robot can continue moving towards the goal following a straight line. Figure 7 depicts an example of Tangent Bug solving a start-to-goal query.

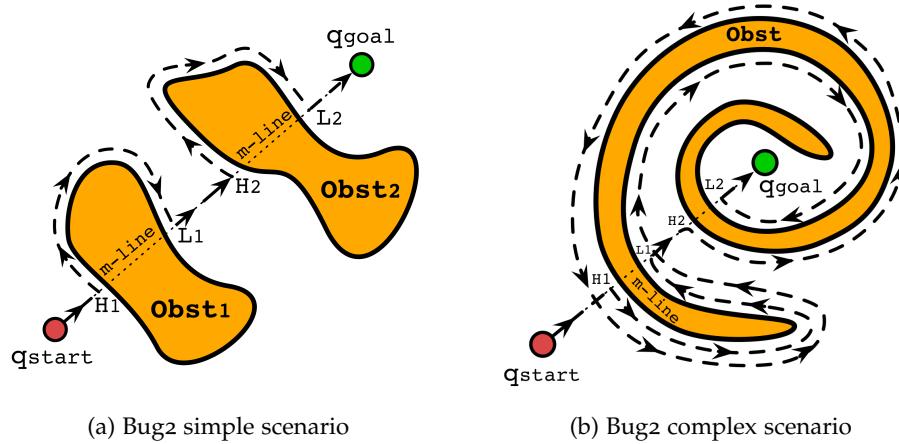


Figure 6: Reactive and sensor-based method Bug2

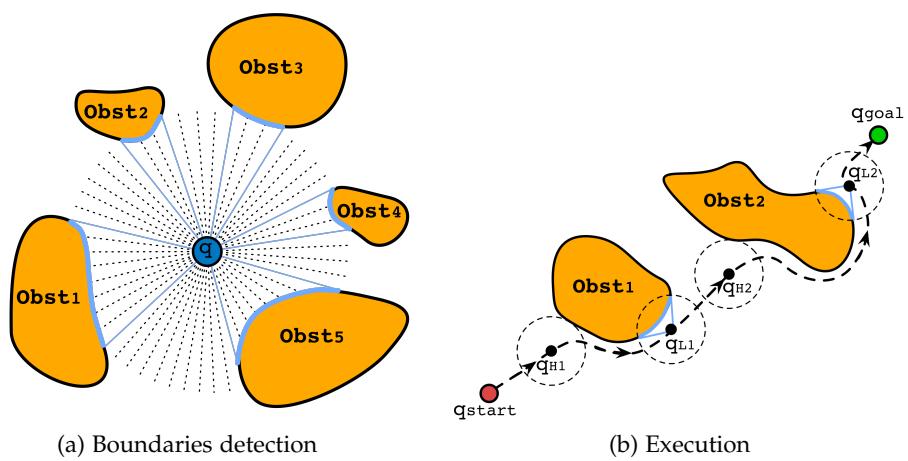


Figure 7: Reactive and sensor-based method Tangent Bug

2.3 SEARCH-BASED (GRID-BASED) PATH/MOTION PLANNING

Search-based planning is a path/motion planning approach that utilizes graph search methods to find collision-free paths over a discrete version of \mathcal{C} . For doing so, these methods overlay a grid on \mathcal{C} and assume that each collision-free configuration corresponds to a point on the grid, which is why they are also called grid-based methods. Over that grid, the robot is allowed to move from one point to any other adjacent point as long as the line between them is proved to be collision-free (i.e., is contained within $\mathcal{C}_{\text{free}}$). Hence, using this approach to solve, for instance, a start-to-goal query requires coping with two problems: how to correctly discretize \mathcal{C} to establish the grid, and how to search a path from the start point to the goal point (configuration) over such a grid.

For the first problem, it is important to correctly define the grid resolution, which mainly depends on the environment and the problem's requirements. Coarser grids, for example, will permit faster searches, but may fail to find paths when dealing with narrow passages in $\mathcal{C}_{\text{free}}$ (see Figure 8). Finer grids, on the other hand, will allow solving queries in more complex scenarios, but may be computationally too expensive for online applications. Once the grid is established, there are different methods to calculate an optimal path, some of which are described in this section.

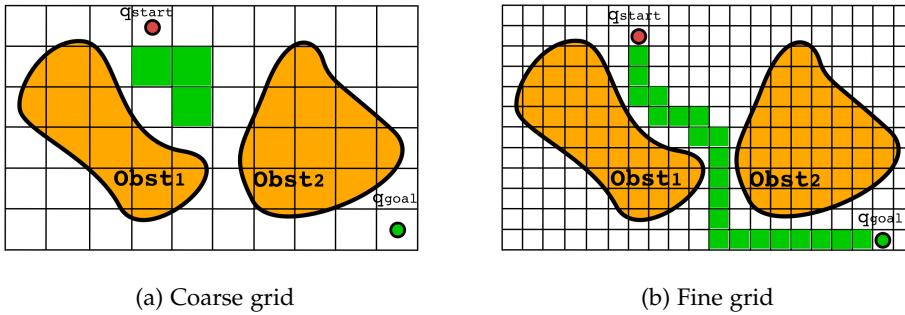


Figure 8: Grid-based method solving a start-to-goal query. (a) Although coarser grids decrease the computing time, they may fail when dealing with narrow passages where finer grids could succeed (b).

2.3.1 Dijkstra's Algorithm

Presented in 1959, Dijkstra's algorithm [23] is one of the first widely known methods used to find a path in a graph from one node to another. The algorithm uses a weighted graph Ω^1 to determine which path that connects two nodes has the lowest cost. While this method has been used in different areas and applications such as network routing protocols, in robotics it has been applied to path/motion planning problems. There, the associated edge cost may correspond to different optimization metrics such as distance, energy, time, etc., that can be considered when calculating the optimal path for a start-to-goal query.

¹ A weighted graph is one in which numerical values, or weights, are assigned to its nodes and edges.

In order to find the optimal path, the algorithm iteratively executes the following steps until reaching the goal (q_{goal}): 1) Setting an initial cost to all nodes, specifically zero to the initial one (q_{start}) and infinity to all others. 2) Establishing q_{start} as the current node and marking the rest as unvisited. 3) Calculating the cost for the current node's unvisited neighbors (equal to the current node's cost plus the edge to the neighbor cost), as well as updating any previously calculated node cost if the new value is lower. For example, if node B cost was x when current node was A, but the new cost is y when current node is C, and $y < x$, then node B cost will be now y . 4) After calculating the cost of all its unvisited neighbors, current node is marked as visited. 5) If the q_{goal} has been marked as visited, the algorithm has finished, otherwise, the new current node will be the unvisited node with the lowest cost, and will be processed as indicated from step 3). Finally, the least cost path can be obtained with backtracking. Figure 9a presents an example of executing this method.

2.3.2 A^*

In 1968, Peter Hart et al. described an extension of Dijkstra's algorithm called A^* [44], which incorporates a heuristic that permits estimating the cost of paths from any node of the graph to the goal. Because of this characteristic, A^* is considered an informed search algorithm; this means that it always attempts to find the path by firstly evaluating those nodes with the minimum estimated cost according to the heuristic. As occurs with Dijkstra's algorithm, A^* starts from a weighted graph Ω , in which each node represents a different configuration contained in \mathcal{C}_{free} and the edges correspond to collision-free paths between configurations. Then, it builds a search tree, rooted at q_{start} , by expanding different paths, one step at a time, until one of them ends at the desired q_{goal} . The main difference with respect to Dijkstra's algorithm is the order in which each partial path is expanded. In the case of A^* , it expands the node q_i that minimizes the total cost $f(q_i) = g(q_i) + h(q_i)$, which combines the cost required to reach the node from the start configuration $g(q_i)$ with the estimated heuristic cost from the node to the goal $h(q_i)$.

While both Dijkstra's algorithm and A^* can generate optimal solutions, the latter can result in a more efficient search by reducing the number of nodes required to be visited in order to determine the solution path (see Figure 9b). However, it is important to know that an incorrect heuristic will also provide a valid solution, but it may be suboptimal. For this reason and in order to produce an optimal path, the heuristic has to be optimistic, or admissible, which means that the estimated cost to the goal has to be lower or equal to the real cost [20]. Finally, it is also important to note that in case no heuristic is provided, i.e., $h(q_i) = 0$, A^* behaves as Dijkstra's algorithm. Algorithm 1 presents the pseudocode for A^* and also provides a general idea for Dijkstra's algorithm explained in the previous section.

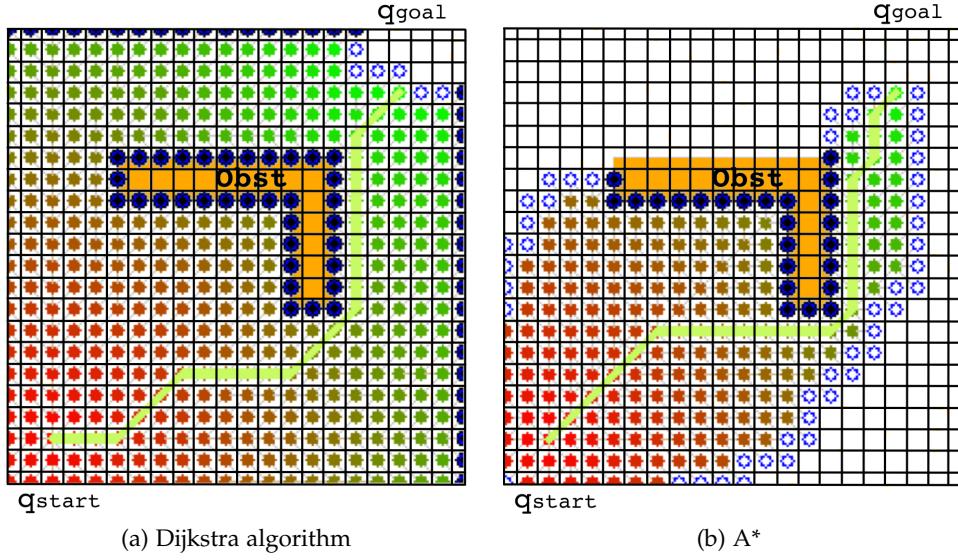


Figure 9: Grid-based methods solving a start-to-goal query. (a) Dijkstra's algorithm requires an exhaustive search to determine the shortest path to the goal. (b) A* requires exploring less cells since it uses a heuristic to guide the search. Image credit: modified from Wikipedia.

Algorithm 1 : A*

Input :

q_{start}: Start configuration.
 q_{goal}: Goal configuration.
 \mathcal{Q} : Graph of configurations q, such that $q \in \mathcal{C}_{\text{free}}$

```

1 begin
2   forall  $q \in \mathcal{Q}$  do
3      $g(q) = \infty$ 
4    $g(q_{\text{start}}) = 0$ 
5   UNVISITED = PriorityQueue()
6   UNVISITED.insert( $q_{\text{start}}, g(q_{\text{start}}) + h(q_{\text{start}}, q_{\text{goal}})$ )
7   while  $\arg \min_{q \in \text{UNVISITED}} (g(q) + h(q, q_{\text{goal}})) \neq q_{\text{goal}}$  do
8      $q \leftarrow \text{UNVISITED.popWithMinCost}()$ 
9     forall  $q' \in \mathcal{Q}.\text{Neighbors}(q)$  do
10       if  $g(q') > g(q) + c(q, q')$  then
11          $g(q') = g(q) + c(q, q')$ 
12         UNVISITED.insert( $q', g(q') + h(q', q_{\text{goal}})$ )

```

2.3.3 Dynamic A* (D^*)

The aforementioned search-based algorithms, at least in their original versions, are intended for planning paths in static environments. Nonetheless, there are situations, especially in mobile robotics applications, in which elements of the environment may change. For those cases, Anthony Stentz proposed the dynamic A* (D^*) [112, 113], an incremental search-based algorithm that plans collision-free paths using a similar strategy as A*. The difference is that it also allows to replan according to changes observed in the surroundings while the robot follows the path to the goal. Its most important characteristic is that it locally repairs the path, which is more efficient than invoking multiple times A* to find a new valid path. Contrary to Dijkstra's algorithm and A*, that both search paths from the start to the goal configuration, D^* expands nodes by searching backwards from the goal until the node to be expanded coincides with the start configuration, at which time the search is concluded. At the moment, D^* and some of its variants are probably the most used search-based methods in mobile robotics. Some of those extensions and applications will be presented in Section 2.7.

2.4 POTENTIAL FIELDS

Even though search-based methods have proved to be successful in 2D and 3D workspaces for some terrestrial, aerial and even aquatic robotic applications, those exhaustive methods suffer from scalability issues in problems involving high-dimensional configuration spaces. Another important drawback is the necessity of establishing a grid over $\mathcal{C}_{\text{free}}$, which means discretizing the C-Space, thus limiting the possible and available solutions according to the chosen resolution. An alternative approach is the use of potential functions.

Originally proposed by Oussama Khatib in 1985 [69, 70], potential functions, also known as *potential fields*, constitute a reactive approach for path-/motion planning, which attempts to guide a robot from an initial configuration to a goal configuration while avoiding obstacles. A *potential field* basically defines a real-valued function $U : \mathbb{R}^m \rightarrow \mathbb{R}$, which is composed of an attractive component $U_a(q)$ that pulls the robot towards the goal, and a repulsive component $U_r(q)$ that pushes the robot away from the obstacles. This function can be viewed as the total energy $U(q) = U_a(q) + U_r(q)$, which means that the total force applied by the potential field to the robot is defined as the negative of the vector gradient, i.e., $f(q) = -\nabla U(q)$, where $\nabla U(q) = DU(q)^T = \left[\frac{\partial U}{\partial q_1}(q), \dots, \frac{\partial U}{\partial q_m}(q) \right]^T$. In other words, the gradient $\nabla U(q)$ establishes the force required at any $q \in \mathcal{C}$, in order to guide the robot throughout a collision-free path towards the goal. Figure 10 presents an example of the two components of a potential field and the total potential field.

Once the potential function and the corresponding gradient have been established, the solution path to a start-to-goal query can be incrementally obtained by using the gradient descent, which is a widely-known algorithm in solving optimization problems. Having q_{start} as the initial

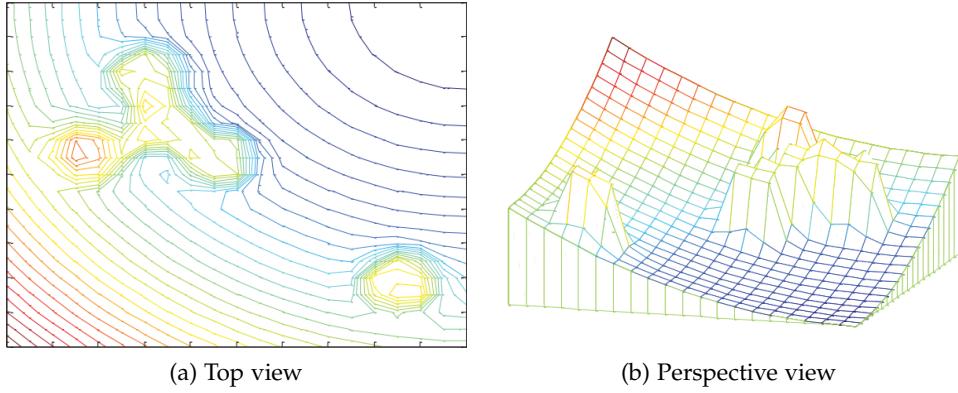


Figure 10: Example of potential field. The obstacles can be observed as a repulsive component of the whole potential. Image credit: Liu et al. [85].

configuration, the algorithm iteratively calculates a new configuration by moving in the direction opposite to the gradient until the gradient is zero. The pseudocode for this method is presented in Algorithm 2.

Algorithm 2 : Gradient Descent

Input :

q_{start} : Start configuration.

$\nabla U(q)$: Gradient of the potential field.

```

1 begin
2   q(0) = qstart
3   i = 0
4   while ∇U(q(i)) ≠ 0 do
5     q(i + 1) = q(i) - ∇U(q(i))
6     i = i + 1

```

However, the gradient descent algorithm using potential functions as explained above does not guarantee finding or converging to a solution for a start-to-goal query. This happens because it may reach a local minimum of $U(q)$ that may not correspond to q_{goal} . To deal with such situations, Barraquand and Latombe proposed the randomized path planner ([RPP](#)) [4, 5], which is an algorithm that makes use of the gradient descent together with random walks and backtracking in order to avoid issues related to local minima. However, it is important to note that [RPP](#) effectiveness is highly dependent on parameter tuning. Another important aspect to highlight is that [RPP](#) was one of the first methods to use a stochastic or random approach for path/motion planning; algorithms with this characteristic will be discussed more in detail in Section [2.6](#).

2.5 ROADMAPS

So far, the methods and algorithms presented above attempt to solve a single start-to-goal query, which means they calculate a path that connects a start configuration and a goal configuration. For doing so, they incrementally search a path towards the goal while avoiding, at the same time, collisions with the obstacles. Nonetheless, there are some applications in

which the path/motion planner is intended to solve more than one query. For those cases, it makes sense to have a map that contains the information about all feasible routes, and that can also be used more than once to solve multiple start-to-goal queries. There are different alternatives to define a map that can be used for path/motion planning, including topological, geometric, and grid-based representations.

This section reviews methods that use a class of topological maps called *roadmaps* [16, 80]. A roadmap (\mathcal{RM}) is defined as a subset of the *C-Space* that results from the union of curves, in which any q_{start} and q_{goal} contained in $\mathcal{C}_{\text{free}}$ can be connected by a path that meets the following properties [20]: 1) **Accessibility** - there is a path from $q_{\text{start}} \in \mathcal{C}_{\text{free}}$ to some $q'_{\text{start}} \in \mathcal{RM}$. 2) **Departability** - there is a path from some $q'_{\text{goal}} \in \mathcal{RM}$ to $q_{\text{goal}} \in \mathcal{C}_{\text{free}}$. 3) **Connectivity** - there is a path in \mathcal{RM} that connects q'_{start} and q'_{goal} .

2.5.1 Visibility Graphs

Visibility graphs are one of the alternatives in defining a roadmap. Assuming a 2D *C-Space* with polygonal obstacles, the set of the visibility graph nodes (v_i) is composed of q_{start} , q_{goal} , and all the vertices of the obstacles. Its edges, e_{ij} , are straight-line segments that can connect any possible combination of two nodes v_i and v_j , without colliding with the obstacles ($e_{ij} \in \mathcal{C}_{\text{free}}$). Once the visibility graph is fully defined, the solution path can be obtained by conducting any graph-based search method, such as those explained in Section 2.3. While Figure 11 depicts an example of a visibility graph and a start-to-goal query solution over it, Algorithm 3 presents the pseudocode to build a visibility graph.

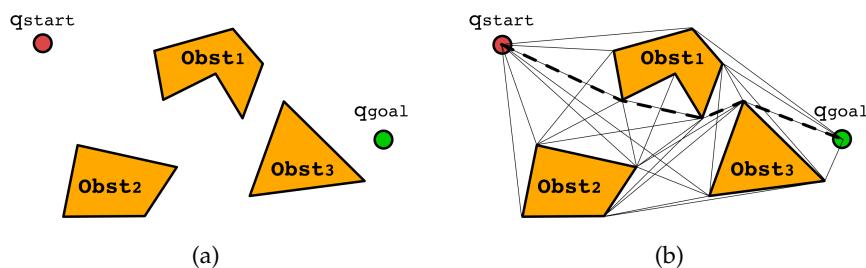


Figure 11: Example of a visibility graph. (a) A 2D *C-Space* that contains polygonal obstacles, a start configuration, and a goal configuration. (b) A roadmap is built by connecting any possible combination of two vertices without colliding with the obstacles. The start-to-goal query is solved using a graph search method.

2.5.2 Generalized Voronoi Diagrams

Another alternative to build a roadmap for path/motion planning is the generalized Voronoi diagram (*GVD*). The *GVD* is defined for a set of points called *sites*; given a particular site, the set of points closest to it is called a *Voronoi region*. Finally, the Voronoi diagram is the set of points that are

Algorithm 3 : Visibility Graph

Input : q_{start} : Start configuration. q_{goal} : Goal configuration.World: n Obstacles.**Output :**Roadmap (\mathcal{RM}): Visibility Graph(VG) = (V, E) .

```

1 begin
2    $V = \{\}$ 
3    $E = \{\}$ 
4   for  $i = 1 : n$  do
5      $V.addNodes(Obst(i).getVertices())$ 
6   for  $i = 1 : V.getNumNodes()$  do
7     for  $j = 1 : V.getNumNodes() and j \neq i$  do
8        $v_i \leftarrow V(i)$ 
9        $v_j \leftarrow V(j)$ 
10       $e_{ij} \leftarrow \text{findStraightLine}(v_i, v_j)$ 
11      if  $\text{isCollisionFree}(e_{ij})$  then
12         $E.addEdge(e_{ij})$ 

```

equidistant to at least two sites [3]. In path/motion planning applications, the **GVD** defines the sets of points equidistant to at least two obstacles. This means that the sites are the center of the obstacles to be avoided, and the edges correspond to the possible channels that maximize the distance to the obstacles [20]. Figure 12 displays an example of a planar **GVD**.

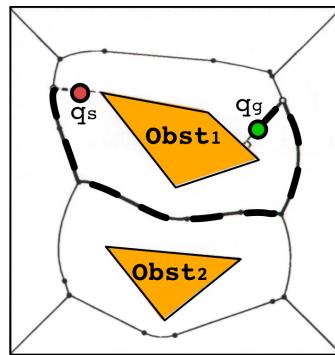


Figure 12: Example of a Voronoi diagram

2.6 SAMPLING-BASED ALGORITHMS

Some of the methods presented in previous sections, such as potential fields, roadmaps, and cell decomposition, require an explicit representation of \mathcal{C}_{free} in order to find a collision-free path from a start configuration to a goal configuration. Nonetheless, there are some situations in which building such a representation is not possible or is computationally intractable, for instance when dealing with high-dimensional configuration

spaces. For those cases, sampling-based methods have been demonstrated to be effective.

Firstly developed during the 90's, *sampling-based algorithms* are an alternative approach that aims to create a sampled (discrete) representation of **C-Space** that captures the connectivity of the regions of $\mathcal{C}_{\text{free}}$. These methods exploit the fact that, while explicitly building $\mathcal{C}_{\text{free}}$ is expensive, checking a given configuration for collisions can be done quickly. To build such a discrete representation, they firstly generate random samples q_{r_i} from **C-Space**, which are checked for collision in order to ensure that $q_{r_i} \in \mathcal{C}_{\text{free}}$. Secondly, they interconnect the random and collision-free configurations, thus establishing different routes (paths) to solve single or multiple start-to-goal queries. This two-stage (sampling and connecting) approach that uses a collision checking routine to validate samples, also allows generalizing the path/motion planning problem. This is done by separating the algorithm from the specific geometric representation of the environment.

However, there is one important characteristic to be considered when using sampling-based algorithms. Contrary to the methods mentioned previously, sampling-based algorithms weaken the completeness guarantee, which means that they are not capable of notifying if a solution exists. Nonetheless, given that many of these methods are based on random sampling, which is dense with probability one [75], this also implies that with enough points (samples), if a solution exists then the probability of finding it converges to one. In other words, if the algorithm runs for a sufficient amount of time, it will find a solution if there is one. This property is called *probabilistic completeness* [6, 65, 67].

These characteristics have led sampling-based algorithms to be considered the state-of-the-art approach for solving various path/motion planning problems. Albeit there are several methods and variants used nowadays, it is possible to identify those that, at the time, were pioneers and the most relevant ones. This section presents such methods and classifies them according to their capability to solve single or multiple start-to-goal queries.

2.6.1 Multiple-query Methods

As it was explained in Section 2.5, roadmaps are data structures that contain all feasible routes that can be used more than once to solve multiple start-to-goal queries. Likewise, there is a sampling-based method called probabilistic roadmap (**PRM**) that creates a graph attempting to represent the connectivity of $\mathcal{C}_{\text{free}}$. **PRM** was developed simultaneously at Stanford [64, 66] and Utrecht [100], and was jointly presented in 1996 [67].

Nowadays, **PRM** is one of the most representative sampling-based algorithms. It is mainly composed of a *preprocessing phase* and a *query phase*. During the first phase, the algorithm builds a roadmap, or undirected graph $G = (V, E)$. The set of nodes (V) contains n collision-free samples q_{r_i} (i.e., $q_{r_i} \in \mathcal{C}_{\text{free}}$) that are randomly obtained from an uniform distri-

bution². The set of edges (E) corresponds to the collision-free paths from each node q_{r_i} to its k closest nodes q_{r_j} ; the connecting paths are calculated by a local planner that checks them for collisions. Algorithm 4 presents the pseudocode for the *preprocessing phase*.

Algorithm 4 : PRM, preprocessing phase

Input :

n : Number of nodes of the roadmap.

k : Number of closest nodes to attempt connection.

\mathcal{C} : [C-Space](#)

Output :

Probabilistic roadmap (PRM): $G = (V, E)$.

```

1 begin
2    $V = \{\}$ 
3    $E = \{\}$ 
4   while  $|V| < n$  do
5      $q_{rand} = \mathcal{C}.generateRandomConf()$ 
6     if  $\mathcal{C}.isCollisionFree(q_{rand})$  then
7        $V.addNode(q_{rand})$ 
8       for  $q_{r_i} \in V.getNodes()$  do
9         for  $q_{r_j} \in \mathcal{C}.getClosestNodes(q_{r_i}, k)$  do
10           $e_{ij} \leftarrow findPath(q_{r_i}, q_{r_j})$ 
11          if  $\mathcal{C}.isCollisionFree(e_{ij})$  then
12             $E.addEdge(e_{ij})$ 

```

Once the roadmap (graph) has been built, the *query phase* attempts to find a collision-free path between the provided q_{start} and q_{goal} . To do so, [PRM](#) tries to connect q_{start} and q_{goal} to their k closest nodes in the graph G . If the connections are successful, a search-based method (e.g., A*, Dijkstra, etc.) attempts to find the shortest path over the graph. If the connections for q_{start} and q_{goal} to the graph are not possible, or if the search-based algorithm fails to find a solution path for the query, it does not necessarily mean that a solution does not exist. As explained above, most sampling-based methods, including [PRM](#), are probabilistic complete, which means that more time may be required to find a solution, if there is one. In this case, it would imply that the number of nodes n have to be increased, as that will generate a more dense probabilistic roadmap.

An important number of variants and extensions have been proposed to deal with different situations in which the originally proposed [PRM](#) may fail [20], [75]. A typical example includes a workspace that creates a [C-Space](#) with narrow passages. In such a case, a common approach would be oversampling the regions of interest (e.g., narrow passages), thus increasing the probability of finding a solution path. Other extensions that have served as a base for the work developed throughout this thesis, will be discussed

² Uniform distribution is the basic form to obtain the random samples, at least in the basic version of the algorithm. Other distributions have been used in some extensions of the original method.

in Section 2.7. Finally, Figure 13 depicts PRM solving a start-to-goal query in a 2D scenario for a point-like robot.

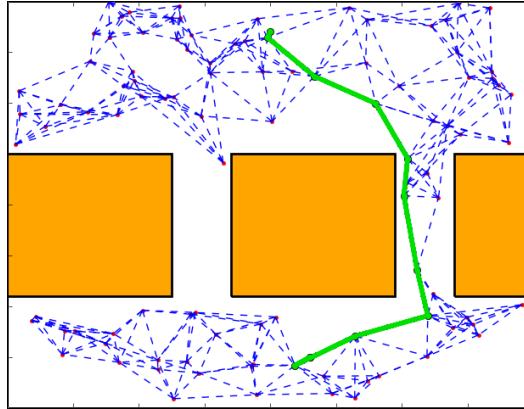


Figure 13: Example of a probabilistic roadmap (PRM) used to solve a start-to-goal query. The PRM was built with 100 random samples, which were attempted to connect to their 6 nearest neighbors.

2.6.2 Single-query Methods

There is another group of sampling-based algorithms that is devoted to solving a single start-to-goal query. Such methods conduct an incremental search, which in most cases is achieved by expanding a tree of randomly sampled configurations. The search is generally biased to finding a solution path for one particular start-to-goal query, rather than attempting to completely represent the connectivity of $\mathcal{C}_{\text{free}}$. These methods are *unidirectional* when a single tree is expanded from q_{start} until reaching q_{goal} or vice versa. Or, they are *bidirectional* if two trees are expanded, one from q_{start} and other from q_{goal} , until both trees meet at a common point. Choosing q_{start} or q_{goal} as the tree's root depends on the specific problem and scenario, since there can be situations in which expanding from the goal is easier (less constrained) than doing it from the start configuration. There are different sampling-based single-query methods, however, this section presents a brief review of those considered the most relevant.

2.6.2.1 Randomized Path Planner (RPP)

Section 2.4 presented an approach for guiding a robot from its current position towards the goal position, by following the negative gradient of an artificial potential field. As mentioned there, one of the main disadvantages of this approach is that the vehicle can get trapped in a local minimum that may not correspond to the specified goal. In order to deal with such situations, Barraquand and Latombe proposed in 1990 the randomized path planner (RPP) [4, 5], which uses the strategy of potential fields, but also incorporates random walks to escape local minima. RPP is commonly acknowledged as the first randomized algorithm. Albeit RPP has proved to be successful in many applications, it may fail when coping with narrow passages.

2.6.2.2 The Ariadne's Clew Algorithm (ACA)

Presented in 1993, Ariadne's clew algorithm ([ACA](#)) builds a tree from q_{start} by interleaving the *exploration* of the [C-Space](#) and the *search* of a connection between the tree and q_{goal} [10, 94]. During the exploration, the algorithm places a random collision-free configuration as far as possible from the others, therefore guaranteeing resolution completeness. New configurations are selected by using genetic optimization methods. These correspond to those from which a connection to q_{goal} is attempted. The main drawback of this approach is that the exploration of the [C-Space](#) is computationally expensive and also requires some parameter tuning.

2.6.2.3 Expansive-Spaces Tree (EST)

Proposed by David Hsu et al. in 1997, expansive-spaces tree ([EST](#)) [54–57] is a single-query method that incrementally builds a tree over the [C-Space](#) by interleaving its *construction* and *expansion*. In contrast to what occurs with [PRM](#) that computes a roadmap attempting to represent the whole \mathcal{C}_{free} , [EST](#) tries to sample the region of \mathcal{C} that is relevant in order to solve the specific start-to-goal query. To do so, during the *construction* phase the algorithm selects the node q to be extended in a way that prioritizes less explored regions, and then it randomly samples a configuration q_{rand} around q . Then, using a local planner, [EST](#) calculates the path that connects q and q_{rand} . In case that both q_{rand} and the calculated path are proved collision-free, they will be added to the tree, thus *expanding* it.

2.6.2.4 Rapidly-exploring Random Tree (RRT)

Within the group of sampling-based single-query algorithms, there is one method that is considered the state-of-the-art, which has been extended, modified and applied to a wide range of applications. This method is known as rapidly-exploring random tree ([RRT](#)) and it was firstly presented by Steven LaValle in 1998 [74]. [RRT](#) is a tree-based algorithm that has different properties such as rapid exploration of the [C-Space](#), probabilistic completeness, ease of implementation, just to mention some. In 1999, LaValle and Kuffner formally presented the [RRT](#) as a path/motion planning method capable of dealing with both geometric and motion constraints. They also proposed a greedy approach to decrease the time required to find a solution by interleaving a random growing of the tree with a biased growing towards the goal [76–78]. They later proposed a bidirectional version called RRT-Connect that extends the basic concept by constructing two [RRT](#)s towards each other [72].

Similar to [ACA](#) and [EST](#), the basic [RRT](#) algorithm is mainly composed of two main procedures, *sample* and *extend*. Algorithm 5 presents the first of them, where the tree is incrementally built until a stop condition occurs; such a condition can either be finding a feasible path that reaches the goal close enough, or that a maximum number of iterations has been completed. In each iteration, the [RRT](#) algorithm attempts to extend the tree towards a randomly sampled configuration q_{rand} . For doing so, the second procedure described in Algorithm 6 finds q_{near} , which is the nearest configuration to q_{rand} in the tree (line 2). Then, a local planner calcu-

lates a path of length δ from q_{near} towards q_{rand} (line 3); if the path is proved collision-free, the algorithm generates a new configuration q_{new} , which together with the calculated path are added to the tree (lines 5-6). A typical growth process of an RRT algorithm can be clearly observed in Figure 14a, where no goal has been specified and the tree attempts to explore uniformly the C-Space. Figure 14b depicts the RRT algorithm solving a start-to-goal query.

Algorithm 5 : sampleRRT

Input :

q_{start} : Start configuration.

q_{goal} : Goal configuration.

\mathcal{C} : C-Space.

Output :

Rapidly-exploring Random Tree (RRT): $T = (V, E)$.

1 **begin**

2 $V = \{\}$

3 $E = \{\}$

4 $V.addNode(q_{start})$

5 **while** not stopCondition($T, goal$) **do**

6 $q_{rand} = \mathcal{C}.generateRandomConf()$

7 extendRRT(T, q_{rand})

Algorithm 6 : extendRRT

Input :

T : an RRT.

q_{rand} : configuration towards which RRT will be extended.

\mathcal{C} : C-Space.

Output :

Result after attempting to extend.

1 **begin**

2 $q_{near} \leftarrow T.findNearestNeighbor(q_{rand})$

3 $q_{new}, collision \leftarrow calcNewConf(q_{near}, q_{rand}, \delta)$

4 **if** $collision = \text{FALSE}$ **then**

5 $V.addNode(q_{new})$

6 $E.addEdge(q_{near}, q_{new})$

7 **return** ADVANCED

8 **else**

9 **return** TRAPPED

2.6.3 Optimal Planning

At least in their original formulation, sampling-based algorithms do not guarantee that the calculated path is optimal with respect to a specified cost function. However, more recent contributions have attempted to cope with this situation. In 2010, Jaillet et al. presented the transition-based

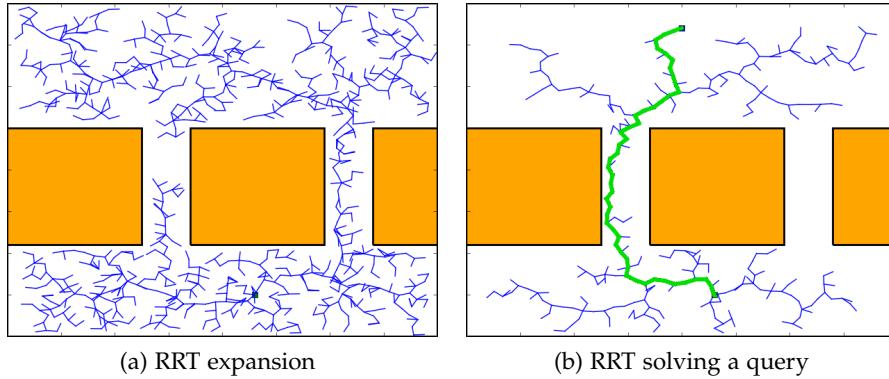


Figure 14: Example of a rapidly-exploring random tree (RRT) in a 2D workspace.

(a) The uniform and rapid exploration of the C-Space is one of the main characteristics. (b) The expansion of the tree stops once a solution has been found.

RRT ([T-RRT](#)), which calculates a low-cost path that follows valleys and saddle points in a costmap established over the [C-Space](#) [59]. To achieve this, it verifies the quality of the path by using the minimal work criterion. Its key principle is that positive variations of the cost function can be seen as forces acting against motion, and thus producing mechanical work. Contrasted to a standard [RRT](#) algorithm implementation, an additional transition validation is performed, which accepts or rejects new potential configurations before adding them into the solution tree.

Nonetheless, the state-of-the-art sampling-based method for calculating optimal paths is the asymptotic optimal RRT ([RRT*](#)). In 2010, Karaman and Frazzoli firstly introduced the [RRT*](#) algorithm and its concept of asymptotic optimality. This property states that the total cost of the solution, measured by a user-defined function, decreases as the number of samples increases [61]. In this approach, new configurations are connected to the closest and best configuration, i.e., the one that guarantees a minimum cost. Furthermore, an additional step of sample reconnection allows improving costs to surrounding configurations (see Algorithm 7). This concept was later extended by the same authors to the [PRM](#) in [62], where they formally presented the [RRT*](#) and [PRM*](#) algorithms. Similar extensions have been done to other RRT-based algorithms as [T-RRT](#) and its respective T-RRT* version [22]. A typical growth process of an [RRT*](#) algorithm can be clearly observed in Fig. 15.

2.7 EXTENSIONS AND APPLICATIONS

Most of the aforementioned methods have been widely used in real-world applications not only with manipulator arms, but also with different aerial, terrestrial and aquatic robotic systems. For doing so, some of these methods have been extended and adapted according to the specific applications' requirements. Considering the problem and objectives stated for this thesis in Chapter 1, this section presents a brief review of the most relevant extensions and applications of path/motion planning algorithms, which

Algorithm 7 : extendRRT***Input :**

T: tree of collision-free configurations.

 q_{rand} : state towards which the tree will be extended. \mathcal{C} : C-Space.**Output :**

Result after attempting to extend.

```

1 begin
2    $q_{near} \leftarrow T.\text{findNearestNeighbor}(q_{rand})$ 
3    $q_{new}, \text{collision} \leftarrow \text{calcNewConf}(q_{near}, q_{rand}, \delta)$ 
4   if  $\text{collision} = \text{FALSE}$  then
5      $\text{addNewNode}(T, q_{new})$ 
6      $Q_{near} \leftarrow \text{findNearestNeighbors}(T, q_{new})$ 
7      $q_{min\_cost} \leftarrow \text{findMinCost}(T, Q_{near}, q_{new})$ 
8      $\text{addNewEdge}(T, q_{min\_cost}, q_{new})$ 
9      $\text{reconnectNearNeighbors}(T, Q_{near}, q_{new})$ 
10    return ADVANCED
11   else
12     return TRAPPED

```

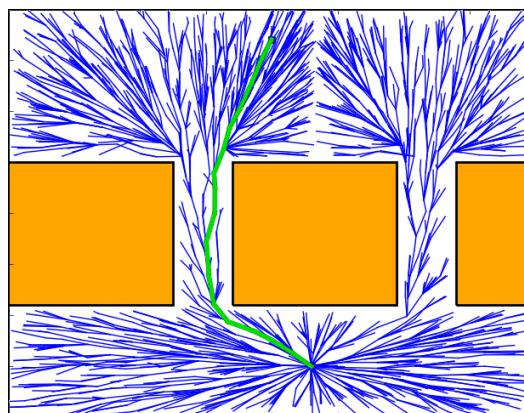


Figure 15: Example of an asymptotic optimal rapidly-exploring random tree (RRT*) in a 2D workspace. The RRT* preserves the uniform and rapid exploration of the C-Space as the standard RRT; however, it can be observed how the additional reconnection step reshapes the tree branches. The expansion of the tree does not stop once a solution has been found. Instead, it keeps improving the solution until either a maximum number of iterations or a maximum time has been reached.

have permitted conducting autonomous missions under both motion and online computation constraints.

2.7.1 Planning Feasible Paths

As explained in Section 1.2.2.2, planning under motion (differential) constraints has to do with considering the limits of the feasible system's maneuvers. These are described by a set of differential equations, generally expressed as $\dot{q} = f(q, u)$ (where q and \dot{q} are the system state and its first derivative, respectively, and u is the control input). A large body of research has been dedicated to the development and improvement of different approaches attempting to solve planning problems that include this kind of constraints.

When Donald et al. formally introduced the problem of kinodynamic motion planning³, they proposed the use of dynamic programming to find the shortest path in a directed graph using depth-first search (DFS). In this case, the graph represents a discretization of $\mathcal{C}_{\text{free}}$; the edges correspond to trajectory segments obtained after applying an acceleration a (bounded according to the system's capabilities) for a period of time τ (determined by the algorithm) [26].

Other variants of grid-based methods, such as A*, have been also used with similar strategies for discretizing $\mathcal{C}_{\text{free}}$. In terrestrial vehicles, for example, the most remarkable contributions were a result of the DARPA Grand Challenge [119]⁴. In one of those works, Likhachev and Ferguson used a multi-resolution lattice state space (a $\mathcal{C}_{\text{free}}$ discretization), where states represent configurations, and connections between them represent feasible paths (i.e., those that consider kinematic constraints). Then, one A* variant (called AD*) would find paths over the lattice [81]. Similarly, Dolgov et al. presented an approach in which $\mathcal{C}_{\text{free}}$ is also discretized and paths are found by running another A* variant (Hybrid-State A*). This is guided by two heuristics, one that considers the vehicle's non-holonomic constraints and a second one that computes the Euclidean distance. This approach also connects the states (configurations) by restricting the control inputs according to the feasible (doable) maneuvers of a car-like system, which are expressed by non-holonomic (kinematic) constraints [24, 25].

In what concerns sampling-based methods, EST and RRT algorithms were initially conceived to cope with motion constraints [57, 78]. In their original versions, the differential equation of motion is used to generate new nodes (states) during the *expansion* of the tree (e.g., in Algorithm 6, line 3)⁵. Figure 16 depicts two sampling-based algorithms: RRT* and RRT in the process of solving a start-to-goal query in a 2D workspace under both geometric and motion constraints.

³ Kinodynamic motion planning alternatively refers to motion planning under motion or differential constraints.

⁴ The DARPA Grand Challenge is a competition of autonomous vehicles, funded by the Defense Advanced Research Projects Agency (DARPA).

⁵ While tree and graph nodes are generally referred to as configurations in geometric path planning, in kinodynamic motion planning they are commonly called states. However, this latter term has been indistinctly used by some authors to refer to them in either case.

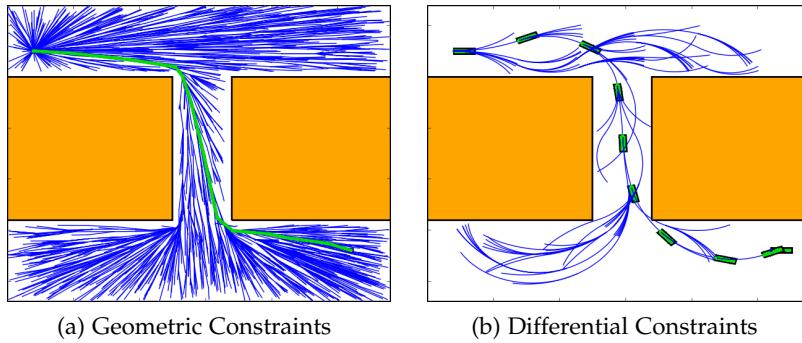


Figure 16: Start-to-goal query solution in a 2D workspace, where obstacles appear in orange, free space in white, and the tree of collision-free configurations in blue. The solution (in green) is calculated by (a) an RRT* algorithm under geometric constraints for a point-like system ($\mathcal{C} = \mathbb{R}^2$), and (b) an RRT algorithm under differential constraints for a car-like system ($\mathcal{C} = \mathbb{R}^2 \times \mathcal{S}$).

There are several successful applications based on this approach. In terrestrial vehicles, one of the most relevant application was presented by Kuwata et al. during The DARPA Urban Challenge. They proposed an approach known as closed-loop RRT (CL-RRT), which not only considers the vehicle’s motion model, but also includes the controller’s dynamic behavior [73]. Another interesting application was done with aerial systems, in which Müller et al. proposed to use an A* for globally finding the collision-free path which, at the same time, guides an RRT algorithm expanded under differential constraints. This latter guarantees finding paths that meet the system’s motion constraints [97].

2.7.2 Online Path/Motion Planning

As discussed in Chapter 1, potential and new applications for AUVs involve navigating unknown or undiscovered environments that require endowing the vehicles with the capability of (re)planning online while, at the same time, they explore the environment. There are different extensions that can contribute to achieving this requirement, however this section focuses on two of them. The first is the *anytime* computation, a characteristic incorporated to different kind of algorithms, including both search-based and sampling-based methods. The second refers to *lazy collision checking*, a strategy specifically used with sampling-based methods. These two characteristics have served as a basis for the proposed approach of this thesis.

2.7.2.1 Anytime Planning Algorithms

In some situations finding a definite path to solve a start-to-goal query in a finite and deterministic period of time is not possible. It may occur either because of the complexity of the task (i.e., more computation time is required) or because vehicles deal with partially known or dynamic environments. In either case, a common approach is to use an *anytime* algorithm that is capable of providing the best partial solution when the

available time is over [131],[21]. The most relevant and well-known planning algorithms have been extended based on this strategy.

Likhachev et al. have studied search-based algorithms, including their extensions for *anytime* computation. They presented the anytime repairing A* (**ARA***), a variant that rapidly calculates a suboptimal path to the goal using a loose bound, which is later tightened to progressively improve the path [82]. While this approach obtains a fast solution, it also permits improving it if additional time is available. Nonetheless, this approach is useful when full and accurate information about the environment is available, otherwise it may require recalculating the whole path if the environment changes. For those cases, i.e., when coping with dynamic environments, it is better to use an incremental search-based method as **D***, which allows locally repairing (replanning) the path when new information of the environment is provided (see Section 2.3.3). However, in its original version, **D*** lacks the *anytime* property. In order to improve **D***'s characteristics, Likhachev et al. also presented the anytime dynamic A* (**AD***). This is a variant that not only replans if required, but also improves simultaneously the available solution path [83]. A detailed discussion of these *anytime* search-based algorithms is provided in [84]. Some applications for vehicles that not only require online/*anytime* computation, but also navigate under motion constraints are also presented in [81].

In the case of sampling-based methods there are also different extensions for *anytime* computation. Belghith et. al, for example, proposed the flexible anytime dynamic PRM (**FADPRM**), which is an approach that combines both: a standard **PRM** for constructing the roadmap and an **AD*** for finding a solution path. This latter one extends the original **PRM** by permitting not only *anytime* calculation, but also a progressive improvement of the resulting path. An important characteristic of this approach is the possibility to establish zones with different values of desirability that allow the sampling strategy to be influenced in order to generate less awkward (inefficient and not smooth) paths [8]. A similar approach presented by van den Berg et al. uses **PRM** to represent the static portion of **C-Space** and an **AD*** to deal with the dynamic elements [9].

On the other hand, and because of its incremental nature, randomized tree-based methods, such as **RRT** algorithm, are more commonly used for applications in partly known or dynamic environments, where online and *anytime* computation is required. Ferguson et al. proposed an anytime RRT-based algorithm that calculates an initial path and its cost using the standard **RRT**, thus ensuring that a first solution is found in the shortest time possible. Then, a modified **RRT** algorithm iteratively generates a series of new solutions that are guaranteed to have a lower cost. The algorithm executes until a stop condition is reached, e.g., when the best available solution path needs to be provided [34, 35]. Other RRT variants as **RRT*** were also formulated as *anytime* algorithms [63].

2.7.2.2 Lazy Collision Checking

Even though **PRM** was originally intended for multi-query applications, Bohlin and Kavraki presented a modified version known as Lazy PRM, which minimizes the execution time by reducing the quantity of collision

checking callbacks when solving a specific (single) query [12, 13]. This variant builds a roadmap just as a standard PRM does, but it assumes that all the nodes (configurations) and edges (paths between configurations) are collision-free, leaving the collision detection to the final stage, i.e., when it must find the shortest path between an initial and a final configuration. If a collision is found, the associated nodes and edges are discarded (eliminated) and a new short path is calculated. The authors also suggested an optional *enhancing roadmap* step when collisions are indeed detected. This consists in including more samples around the discarded nodes.

This strategy of delaying the collision detection is known as *lazy collision checking*, and has been used in different applications, especially those with online computation requirements. Bekris and Kavraki, for instance, proposed and validated a tree-based planning framework for terrestrial vehicles, where the states validity is only checked once a path between the start and the goal configuration has been found [7]. Another example is the one presented by Vahrenkamp et al., where this strategy was used to speed up the motions calculation for humanoid robots (with many degrees of freedom, i.e., high dimensional C-Space) [121].

Finally, it is important to note that *lazy collision checking* has served as a base for one of the extensions proposed and used in this thesis, which will be explained in detail in Chapter 5.

2.8 PATH / MOTION PLANNING FOR AUVS

An important aspect to consider when comparing the path/motion planning approaches for AUVs is their application. Based on this, the different contributions can be classified into two main categories. The first group gathers those applications that require coverage path planning (CPP) techniques, which are commonly applied to guide AUVs over survey tasks. The most common examples within this group include coverage missions used for creating detailed bathymetric maps of the seabed [33, 38, 40, 41], detecting potential targets (such as underwater mines [111, 128]), and inspecting artificial structures (such as in-water ship hulls [29, 30, 48, 53]), as well as natural marine formations [37, 42].

A common characteristic in all these approaches is that the planner is provided with preliminary information of the target area or structure (see Fig. 17). This may include its location and shape. Based on this, the CPP algorithm defines a survey path that, in some cases, is reshaped or refined online according to the data obtained during the mission execution. This characteristic implies that most of the computation is done offline, i.e., before conducting the mission. Most recent work presented by Vidal et al. proposes a novel approach to conduct inspection tasks without preliminary information of the target. Results are, however, still limited to 2D motions (at a constant depth) [122].

The second group of AUV applications, on the other hand, gathers those that are focused on safely and efficiently guiding the vehicle from one initial position to a specified goal. For doing so, different strategies, such as those explained throughout previous sections, have been applied to underwater vehicles. In fact, these start-to-goal methods are commonly used as

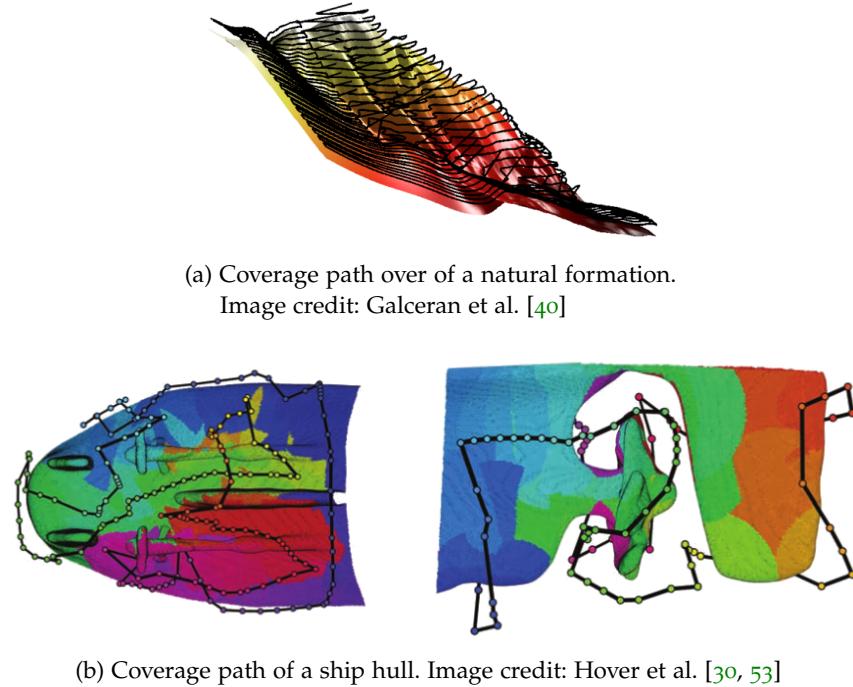


Figure 17: Coverage path planning (CPP) algorithms are normally used to plan routes to inspect different kind of structures. In most of these applications the planner has preliminary information. This may include a 2D/3D map of the area or structure to be inspected.

low-level motion planners required for the aforementioned coverage applications. The work presented throughout this thesis seeks to make contributions to this group of start-to-goal applications, therefore it is important to identify the main characteristics between the different approaches used.

2.8.1 Start-to-goal Path/Motion Planning for AUVs

One characteristic to consider in a start-to-goal planner for AUVs, and yet not an obvious one, is the capability of conducting 2D and 3D motions. Although AUVs operate in 3D workspaces, in a significant number of applications the vehicles navigate either at a constant altitude or at a constant depth [95, 102, 104, 110], thus simplifying considerably the motion planning problem. There are, however, some contributions that have presented alternatives in modelling and planning 3D (and therefore 2D) AUV motions.

From the approaches that address 3D motions, the available contributions have made use of different approaches such as potential fields [107, 124], genetic algorithms [1, 49, 116], as well as sensor-based [52, 130], grid-based [18, 71, 107], and sampling-based methods [14, 95, 104]. However, in some of these situations, the vehicle operates in open sea areas, where they do not usually have to deal with obstacles, narrow passages, or high-relief environments. This kind of constraints corresponds to the new and potential AUV applications presented in Chapter 1.

2.8.2 Online Motion Planning for AUVs through Unexplored Environments

New AUV applications require a path/motion planner to safely guide the vehicle through unexplored and challenging environments. This implies meeting online computation limitations while, at the same time, considering the vehicle's motion capabilities. Little research on this area, especially for underwater vehicles, has been addressed. In what concerns to generating AUV feasible paths, i.e., those that meet the motion constraints, a first group includes those approaches that use sensor-based methods either to navigate through unknown underwater environments [130], or to follow the terrain shape of a given bathymetric map [52] (see Fig. 18). In both cases, the vehicle is assumed to be equipped with a forward-looking sonar to reactively avoid collisions. Such maneuvers are calculated by a local planner that meets the AUV's kinematics or dynamics. An important characteristic of this kind of approach is the lack of global knowledge of the environment (i.e., a map is not incrementally built), which can cause the vehicle to get trapped in complex scenarios.

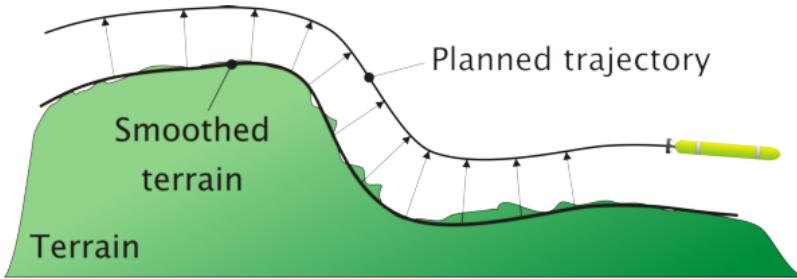


Figure 18: Sensor-based planning for AUVs under motion constraints. The vehicle trajectory can be either preplanned or incrementally calculated to maintain a desired distance from the terrain. The resulting trajectory is fit to satisfy curvature constraints that approximate the AUV motion constraints. Image credit: Houts et al. [51, 52].

A similar reactive approach establishes a set of inequality constraints that describes the obstacles as convex regions contained in the C-Space [102]. The initial configuration is treated as the starting point of a nonlinear search, where the goal configuration is assumed to be a unique global minimum of the objective function. The start-to-goal query is then solved as an optimization problem, in which a local planner takes into account the vehicles' constraints. This strategy was one of the first online obstacle avoidance approaches for underwater vehicles; it used a real-world dataset of acoustic images obtained by a ROV equipped with a multibeam forward-looking sonar. Its validity was demonstrated by guiding a simulated ROV. However, the capability of simultaneous mapping (detection) and planning online was not established.

A formulation that represents obstacles as convex regions has likewise been used either to represent obstacles detected online, which triggers collision avoidance maneuvers [106], or to approximate the terrain shape that must be followed by the vehicle [98]. In both cases, the low-level controller attempts to generate feasible trajectories by using the AUV kinematic equations and spline-based interpolation techniques, respectively. However, the

main drawback of these approaches is the difficulty in creating a convex representation of complex obstacles.

Another common strategy used in some of the aforementioned methods, is trying to get as close as possible to a path that can be followed by an AUV. Pêtrès et al., for instance, proposed a fast marching (FM)-based approach to find collision-free paths, which are smoothed by a cost function that contains kinematic and curvature constraints [103] (see Fig. 19). Likewise, another example based on genetic algorithms (GAs) finds a valid route to the goal by using basis spline (B-spline) curves, thus seeking to generate more feasible trajectories for AUVs [19].

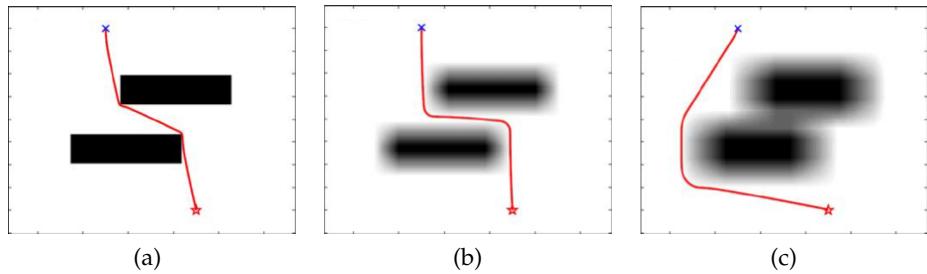


Figure 19: Start-to-goal query that is solved by a FM-based method. The solution path is smoothed by a cost function to approximate the AUV motion constraints. (a) The initial optimal path. (b) The effect after smoothing the path. (c) The use of the cost function can merge the obstacles, thus discarding possible solutions. Image credit: Pêtrès et al. [103].

Unlike the previous group of sensor-based approaches, the second group of path planning methods for motion-constrained AUVs uses grid-based approaches. Sequeira and Ribeiro, for example, presented a two-layer framework that is composed of a high-level planner (HLP) and a low-level planner (LLP) [107]. The HLP creates a visibility graph using the information of the known obstacles, sea currents, and specified waypoints of the mission. Furthermore, the energy required to move between the graph nodes corresponds to the edge weights. The global and optimal geometric route to the goal is then found by Dijkstra's algorithm. Finally, in order to calculate the vehicle's maneuvers between the different solution segments, the LLP uses an artificial potential field (AFP). This way the total artificial force includes a 3D double integrator that takes into account the AUV motion constraints. There are other similar two-layer approaches, where the global path planning problem is tackled with a grid-based method, and the local motion planning deals with the AUV constraints [2] (see Fig. 20). The main disadvantage of line of works is that the grid-based layer generally requires *a priori* information of the environment, e.g., a navigation map.

A third group of recent contributions includes those that use sampling-based planning algorithms. The most common approach builds a tree of collision-free configurations, which are obtained by integrating the differential equation that describes the AUV dynamic behavior [14, 45, 117]. This strategy was briefly introduced in Section 2.7.1, but its use with a torpedo-shaped AUV will be explained in more detail in Chapter 3.

Finally, there is another concept for generating feasible paths. It consists in utilizing the Dubins curves [27], which establishes a set of six maneu-

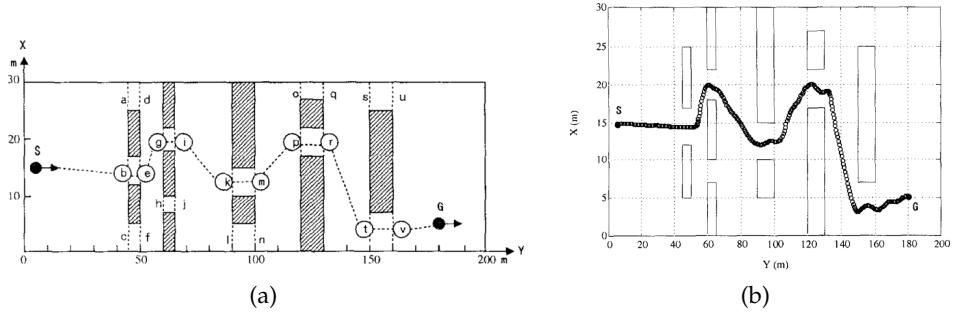


Figure 20: (a) Global motion planning solved by a grid-based method, (b) which is adjusted by a local planner that takes into consideration the AUV motion constraints. Image credit: Arinaga et al. [2].

vers (RSR, RSL, LSR, LSL, RLR, LRL, where R states for Right, L for Left, and S for straight) to connect two configurations $q_i, q_j \in \mathcal{C} = \text{SE}(2)$. These curves correspond to time-optimal trajectories for car-like vehicles. Albeit they have been used for trajectory generation [17, 125], they are also useful in the context of this thesis.

Another characteristic required for the new AUV applications is the capability of mapping and planning safe paths, simultaneously and online, as the environment is incrementally explored. Apart from the already explained work by Petillot et al., which in fact does not demonstrate online mapping and planning capacity [102], Maki et al. proposed an online motion planning method that uses landmarks to guide an AUV. Nonetheless, their approach does not permit replanning and, furthermore, results were obtained in a controlled environment (i.e., in a water tank) [90].

2.9 SUMMARY

This chapter has presented an extensive review of the most common path/-motion planning approaches, and those that have been used with underwater vehicles. Table 1 presents a summary of such methods and their characteristics. It is important to bear in mind that the intended applications for AUVs, discussed in Chapter 1, must meet online computation constraints in order to deal with unexplored environments. This implies that time complexity and anytime computation are important requirements. The following chapters will present the extension of some of these approaches, and their successful use in some of the intended applications introduced in Chapter 1.

Method	Completeness	Optimality	Time Complexity	Anytime
Search-based				
Dijkstra	Yes ^a	Yes	$O(n^2)$ ^a	No
A*	Yes	Yes	$O(b^d)$ ^b	No ^b
Potential fields				
Potent. Funct.	No ^c	Locally	$O(1)$	Yes
Roadmaps				
Visibility Graphs	Yes	Yes ^d	$O(n^2)$ ^d	No
Sampling-based				
PRM + Dijkstra	Prob. Compl.	Asymp. Opt.	$O(n^2)$ ^e	No
RRT	Prob. Compl.	No	$O(n \log n)$ ^e	Yes
RRT*	Prob. Compl.	Asymp. Opt.	$O(n \log n)$ ^e	Yes

Table 1: Path/motion planning methods. ^a Complete for bounded [C-Space](#), where n is the number of nodes in the graph. ^b The number of nodes expanded is exponential in the depth of the solution, where b is the branching factor (the average number of successors per state); ARA* provides a mechanism for anytime computation by reusing previous solutions. ^c requires full knowledge of the [C-Space](#), suffers from local minima. ^d Optimal with respect to the traveled distance; where n is the number of points defining obstacles. ^e Number of sampled configurations.

PLANNING CONSTANT-DEPTH PATHS UNDER AUV MOTION CONSTRAINTS

Recent and potential applications for AUVs mentioned in Chapter 1 establish most of the motion planner requirements, such as online computation, motion in 3D workspaces, and navigation along unexplored environments in close-proximity to nearby obstacles. All these constraints can be met with sampling-based planning methods. Particularly for navigating in close-proximity, one desired characteristic is being able to calculate feasible motions that take into account the AUV capabilities. This allows minimizing unexpected vehicle trajectories when attempting to follow the calculated path. In this respect, this chapter firstly explains the equations of motion used to approximate the 2D (at a constant depth) vehicle behavior for motion-planning purposes. Secondly, it explains two different approaches to integrate such equations into the motion planner. Lastly, it presents and discusses the results obtained with both approaches in a simulated environment.

3.1 2D FIRST-ORDER MOTION MODEL

As occurs with any mechanical system, AUV motions can be formulated with kinematic and dynamic models. The former ones describe the geometry of motion by relating the system positions and velocities. Dynamic models, on the other hand, not only include the system kinematics, but also take into account the forces and torques that generate the motions. This latter kind of model is generally employed for designing robust controllers, however, their high computational cost makes them an inappropriate approach for the scenarios proposed in this thesis, which require an online (re)planning behavior. The associated overhead is especially true when using sampling-based planning methods, where all configurations are generated by evolving the system, i.e., performing numerical integration of the equation of motion. Alternatively, a kinematic model provides a less accurate approximation for the vehicle's motion constraints, but it also allows additional computation time that can be dedicated to finding better collision-free paths.

In order to establish which kinematic model must be used, it is necessary to understand the vehicle motion capabilities and the possible test scenarios. In some AUV applications, a first valid approximation is to assume that the vehicle navigates at a constant depth (see Sec. 2.8). Furthermore, if the vehicle is a torpedo-shaped AUV, usually, it will only be able to change its direction of motion by moving forward/backward. In the case of the Sparus II AUV, the vehicle is equipped with two back thrusters that allow it to spin over. However, it is still not capable of conducting lateral motion. Therefore, it is correct to affirm that the vehicle is subject to a motion constraint along its y-axis (see Fig. 21).

As explained in Chapter 1, motion or differential constraints can be expressed as a set of differential equations in the general form $\dot{q} = f(q, u)$, where q and \dot{q} are the system state and its first derivative, respectively, and u is the control input. Having this in mind, and considering the aforementioned constraints when navigating at a constant depth, a non-holonomic and torpedo-shaped AUV can be represented as a simple car-like vehicle, with a first-order motion model defined as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v \cos(\psi) \\ v \sin(\psi) \\ w \end{bmatrix}, \quad (1)$$

where $q = [x, y, \psi]^T$ corresponds to the system state that includes its 2D position and orientation with respect to an inertial reference frame, and $\dot{q} = [\dot{x}, \dot{y}, \dot{\psi}]^T$ is the first time derivative that depends on the state itself and the control inputs, i.e., linear/surge speed (v) and turning rate (w). From Eq. (1), it can be concluded that the C-Space of a torpedo-shaped AUV is $\mathcal{C} = \text{SE}(2) = \mathbb{R}^2 \times \text{SO}(2) = \mathbb{R}^2 \times \mathcal{S}$. Figure 21 depicts the inertial frame, the body-fixed frame, and the different variables (positions and velocities) contained in (1).

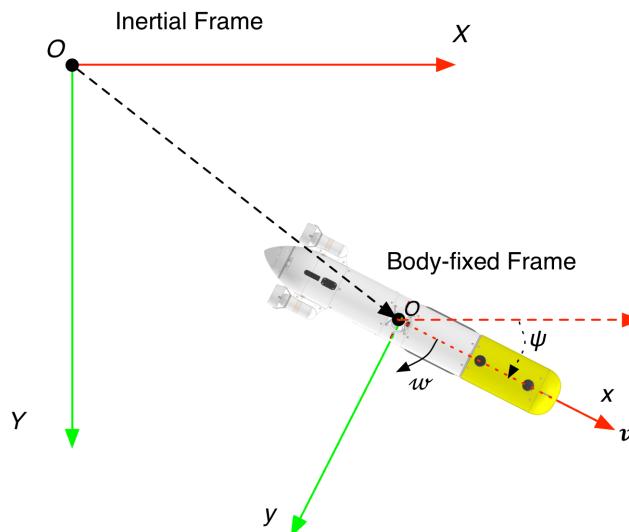


Figure 21: Top view of the Sparus II AUV, including the inertial and body-fixed frames, and the 2D vehicle state and control variables.

Once the constraints have been established through the corresponding differential equations, the next step is to define an appropriate strategy to generate feasible motions that meet such constraints. The following sections present two different approaches, which use sampling-based algorithms to calculate such kind of motions for a torpedo-shaped AUV that navigates at a constant depth.

3.2 MOTION PLANNING BY USING DIFFERENTIAL EQUATIONS

Sampling-based algorithms, especially the RRT [78] and its variants, have proved to efficiently explore the C-Space while taking all the motion con-

straints into account. Furthermore, there are other characteristics such as the incremental search behavior, which allows these methods to continuously (re)plan the solution path as the vehicle moves through unexplored environments (see Chapter 5). All this together makes an RRT-based algorithm the appropriate approach for the intended AUV applications.

As explained in Section 2.6.2.4, the RRT algorithm builds a single tree that is rooted at an initial state (configuration), which is incrementally expanded towards uniformly randomly *sampling* states¹. This procedure is the same as the one initially presented in Algorithm 5, being this a particular case where a random state corresponds to $q_{rand} \in SE(2)$, as explained before. In order to *extend* the tree towards q_{rand} while meeting the vehicle's motion constraints, new collision-free motions (states) are obtained by integrating the vehicle's equation of motion, such as Eq. (1). In such a case, instead of using Algorithm 6, the tree expansion can be rewritten as shown in Algorithm 8. This procedure firstly requires finding the state q_{near} , that is the nearest to q_{rand} (line 2). This can be done by using a weighted metric to calculate the distance between configurations. Such a metric combines the translation component, calculated as the Euclidean distance, and the orientation component, calculated as the smallest orientation difference [20, 75].

Algorithm 8 : extendRRT (when dealing with motion constraints)

Input :

T: tree of collision-free configurations.

 q_{rand} : configuration towards which the tree will be extended. \mathcal{C} : C-Space.**Output :**

Result after attempting to extend.

```

1 begin
2   qnear ← T.findNearestNeighbor(qrand)
3   unear_to_rand ← findInput(T, qnear, qrand)
4   qnew, collision ← calcNewState(qnear, unear_to_rand, Δt)
5   if collision = FALSE then
6     V.addNode(qnew)
7     E.addEdge(qnear, qnew)

```

Once the nearest motion has been found in the tree, the next step is to calculate the control input $u_{near_to_rand} = [v, w]$ that has to be applied in order to change the vehicle state from q_{near} towards q_{rand} (line 3). With an RRT algorithm under geometric constraints the tree is iteratively expanded a geometric distance ϵ (see Algorithm 6, line 3), in the case of an RRT algorithm under motion constraints the tree is expanded by applying an input u for a period of time $Δt$ (line 4).

In some AUV applications, a common assumption is that the vehicle navigates at a constant surge speed v , but can vary its turning rate w

¹ There exists another RRT variant called RRT-Connect that grows two trees, one from the start state and another one from the goal state. However, this is not possible when dealing with motion constraints, since merging both trees at a common configuration may become computationally intractable [72].

up to a maximum value w_{\max} . This means that calculating the input u is limited to firstly establishing a constant v that will be used along the mission, and then computing a valid value for w that meets the vehicle motion constraints. There are different approaches that permit calculating u . Two alternatives, also presented by LaValle and Kuffner [78], are either to randomly sample the control space, or to test a number of possible inputs and then select the one that generates the state q_{new} that is closest to q_{rand} . The implementation used in a first stage of this thesis is based on a combination of both approaches.

In such an implementation, instead of testing a large number of different turning rates w , a set of five possible values is established as follows: $w_{\text{control}_i} \in \{-w_{\max}, -\frac{w_{\max}}{2}, 0, \frac{w_{\max}}{2}, w_{\max}\}$. These values, in turn, define a set of four sub-intervals over the control space: $\{[-w_{\max}, -\frac{w_{\max}}{2}], [-\frac{w_{\max}}{2}, 0], [0, \frac{w_{\max}}{2}], [\frac{w_{\max}}{2}, w_{\max}]\}$. Then, using Eq. (1) with q_{near} as the starting state, each $u_i = [v, w_{\text{control}_i}]$ is applied for a period of time Δt to generate five different states q_{control_i} . Having them, it is possible to estimate which of the four sub-intervals may contain a control input that leads the tree expansion from q_{near} closer to q_{rand} . The final value of w_{rand} is obtained by sampling a random turning rate over the chosen sub-interval. This procedure avoids discretizing the control space, which would imply the loss of the random exploration of the **C-Space**, an important property of the original algorithm. Figure 22 presents the main concept of the control input selection.

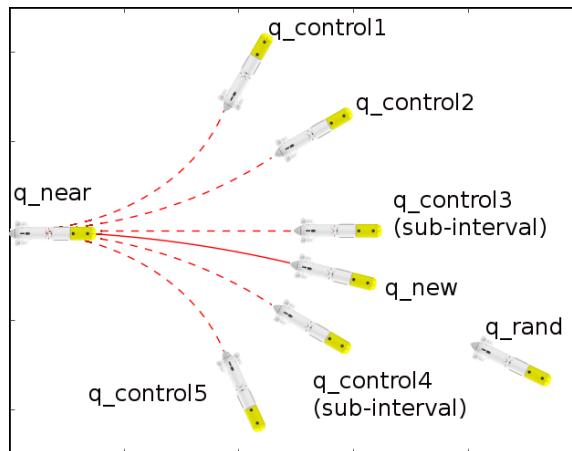


Figure 22: Control input selection. Tree is expanded from q_{near} towards q_{rand} . To do so, an interval of possible turning rates (q_{control_i}) are evaluated. This permits defining a sub-interval from which the control input is sampled randomly, and is applied to generate the new state q_{new} .

The remaining of the *extend* procedure (Algorithm 8) calculates the new state q_{new} (line 4). This is done by integrating Equation (1) while using the final sampled input $u_{\text{near_to_rand}} = [v, w_{\text{rand}}]$. Finally, if q_{new} is proved to be collision-free, it is added and connected to the tree (lines 5-7). Figure 23 depicts an example of an RRT algorithm that solves a start-to-goal query under these motion constraints.

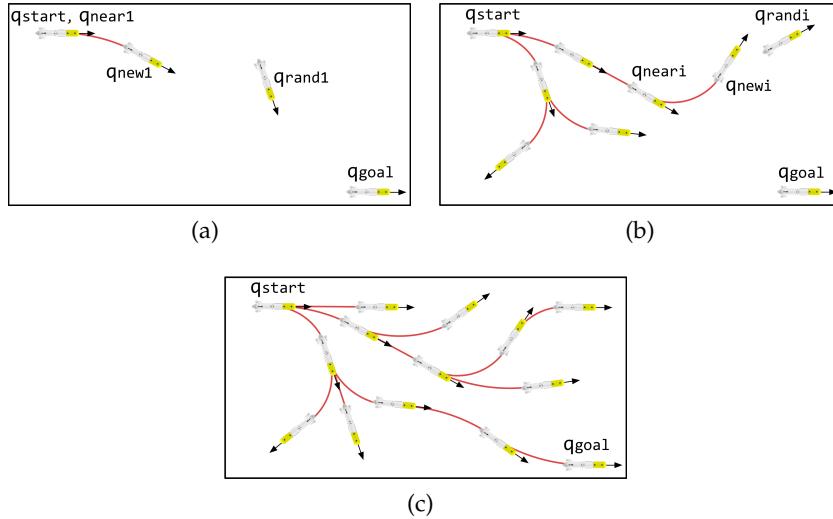


Figure 23: Tree expansion of an [RRT](#) algorithm that considers the differential constraints described by Equation (1). (a) Given the start and goal states, a first random sample is used to generate the first expansion of the tree (q_{new}). (b) The expansion i^{th} of the tree. (c) A feasible path has been found from the start to the goal state.

3.3 MOTION PLANNING BY USING DUBINS CURVES

There are several examples where a purely geometric path is first computed, and then it is transformed into a path that is appropriate for the considered vehicle. For example, Yang et al. used an [RRT](#) algorithm to find a route of collision-free waypoints, which is interpolated with a cubic Bézier spiral [129]. This seeks to convert the initial route into a smooth and feasible path for an unmanned aerial vehicle ([UAV](#)). As another example, Kuwata et al. proposed to expand an [RRT](#) by considering not only the vehicle dynamics, but also its controller behavior [73].

There are other [RRT](#)-based approaches that have been proposed to generate feasible paths for aerial and terrestrial vehicles. In one of those approaches, for instance, a geometric [RRT](#) algorithm finds a route of collision-free waypoints, which are interpolated with a cubic Bézier spiral. This seeks to convert the initial route into a smooth and feasible path for an [UAV](#) [129]. In another approach that is closer to the one presented in the previous section, an [RRT](#) algorithm is expanded by considering not only the vehicle dynamics, but also its controller behavior [73].

Nonetheless, all those approaches, including the one presented in the previous section, have a major drawback; they do not guarantee optimality for any metric. It is a common characteristic of most sampling-based methods, at least in their original form. This issue could be critical in underwater applications, which may require optimizing different criteria such as visibility (for gathering information), vehicle autonomy, or even the safety associated with a path when navigating in close-proximity to nearby obstacles.

As explained in Section 2.6.3, one option to cope with this situation is to use the [RRT*](#) algorithm, which is a variant that incorporates the asymptotic optimality property [62]. Its main difference with respect to other [RRT](#)-

based methods, is a routine that checks if reconnecting new state's nearest nodes improves their associated cost. This implies that the probability of obtaining an optimal path converges to 1 over time. (see Algorithm 7). For doing this, the RRT* algorithm requires a steering function that permits calculating such states (nodes) reconnection. In the case of systems under motion constraints, having such a function implies calculating the required input to dynamically evolve the system from a given state to a desired one. However, defining this function requires solving a two-point boundary value problem, which is in general a very difficult problem.

For the purposes of this thesis, as an alternative to defining a steering function, it is possible to adopt the Dubins vehicle model [27]. Dubins geometrically demonstrated that, for a system that is only capable of traveling forward and with a constraint on the curvature of the path, the shortest path to connect any two configurations (states) $q_i, q_j \in SE(2)$, when no obstacles are present, can be obtained analytically by the combination of circular arcs and straight lines. Using three possible maneuvers as input, left (L), straight (S) or right (R), Dubins curves define six possible combinations: RSR, RSL, LSR, LSL, RLR, LRL. For a Dubins vehicle, at least one of these characterizes the optimal (shortest) trajectory between two states (see Fig. 24).

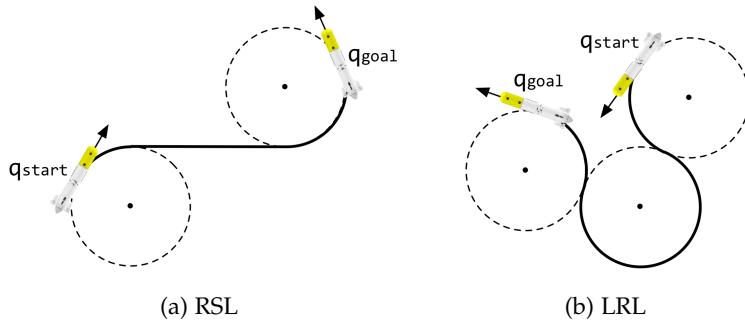


Figure 24: Examples of Dubins curves

These constrained maneuvers are commonly used to describe feasible trajectories for different ground, aerial, and underwater vehicles such as a torpedo-shaped AUV. For this latter case, Equation (1) describes an AUV that navigates at constant depth and with a constant surge speed v , where the maximum turning rate ω_{\max} establishes the minimum turning radius r_{\min} for a Dubins vehicle. For both arcs and straight lines, this means that for any given position (x, y) over the path, its tangent angle corresponds to the AUV heading (ψ), thus completing the state information $q = [x, y, \psi]$.

This alternative formulation for the AUV motion constraints can be used with incremental search methods such as the RRT variants. In such a case, the control input $u_{\text{near_to_rand}}$, required to evolve the system from q_{near} to q_{rand} (Algorithm 8, line 3), can be replaced with Dubins curves (see Fig. 25). Furthermore, it is important to note that Dubins curves also work as a steering function, thus permitting to generate near-optimal paths with an RRT* algorithm. Figure 26 depicts an example of how this approach is used for reconnecting configurations near to q_{new} , thus improving their

associated cost. In this example the cost is assumed to be the path length. Further details about Dubins curves are found in references [27],[108].

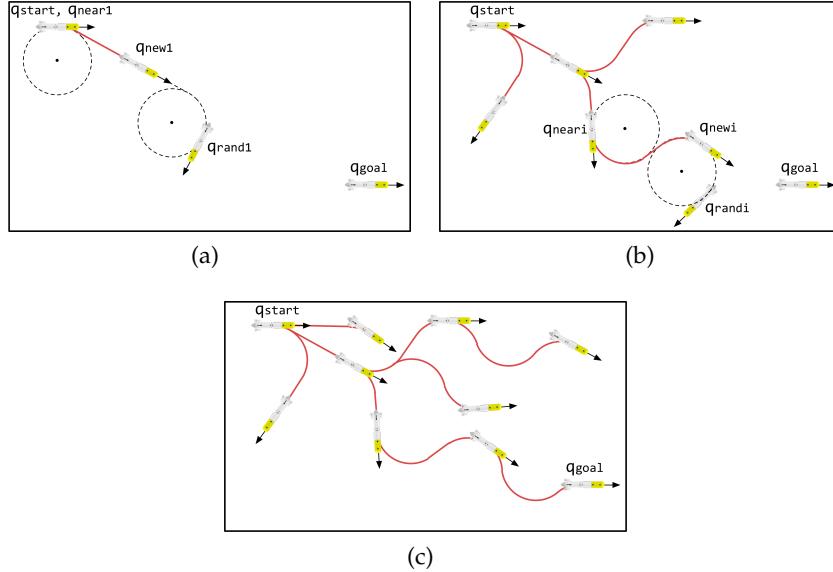


Figure 25: Tree expansion of an RRT algorithm that uses the Dubins curves. (a) Given the start and goal states, a first random sample is used to generate the first expansion of the tree (q_{new}). (b) The expansion i^{th} of the tree. (c) A feasible path has been found from the start to the goal state.

Finally, in order to fully understand the aforementioned approaches, next section presents solutions for different start-to-goal queries that have been obtained using both geometric and motion constraints. Moreover, it also proves the advantages of using Dubins curves instead of using a standard RRT algorithm by integrating Equation (1).

3.4 RESULTS

The main motivation behind the research presented throughout this thesis was the need to plan more accurate motions that permit an AUV to operate in environments that are more complex. As explained in the Introduction, this manuscript is organized in different chapters, each presenting the incremental development of a motion planning framework that endows AUVs with such capabilities. Therefore, the results presented and discussed in this section not only tackled one of the proposed objectives, but largely established the starting point for the rest of this research.

Having said that, the rest of this section seeks to prove the importance of taking motion constraints into consideration when using non-holonomic AUVs. To do so, simulation results present the Sparus II AUV conducting different missions that were planned under both geometric and differential constraints. In all cases, the vehicle navigated at a constant depth in a pre-explored environment, i.e., the obstacles and their locations were known.

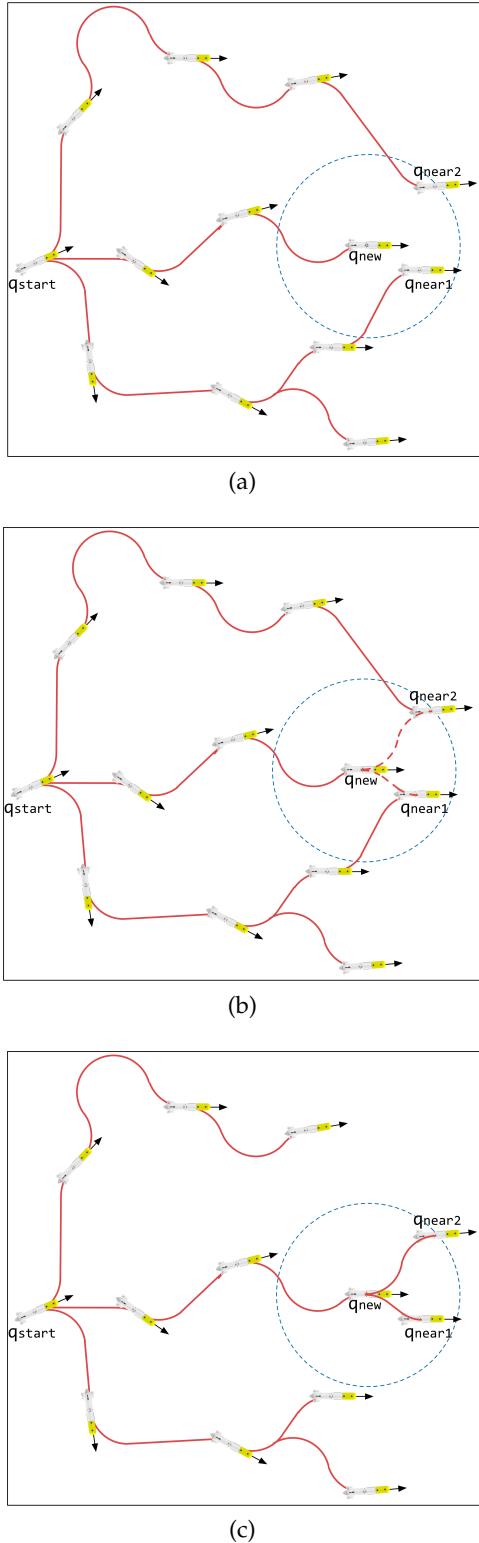


Figure 26: **RRT*** algorithm reconnection using Dubins curves as steering function.

(a) Once a new configuration q_{new} has been obtained, (b) the **RRT*** algorithm checks if the nearest configurations to q_{new} can be reconnected. (c) If such a reconnection decreases the cost associated with the nearest configurations, the new paths are created while discarding the previous ones. This allows to progressively improves the solution path.

3.4.1 Conducting Missions under Geometric Constraints

One simple alternative for planning paths for an AUV that operates at constant depth could be to approximate the vehicle to a point-like system, where each configuration $q = [x, y] \in \mathbb{R}^2$. In such a case, an RRT* algorithm could generate (near) optimal paths if enough computing time is provided. Figure 27 depicts the solution path for a start-to-goal query calculated for a point-like system using the RRT and the RRT* algorithms, both under geometric constraints. However, in both cases there are situations in which the path is so close to the obstacles that it could lead to a collision if any real (non-point-like) vehicle would actually try to follow the path. To prevent this, a simple solution is to increase either the size of the obstacle or the size of the vehicle when checking for collisions [20].

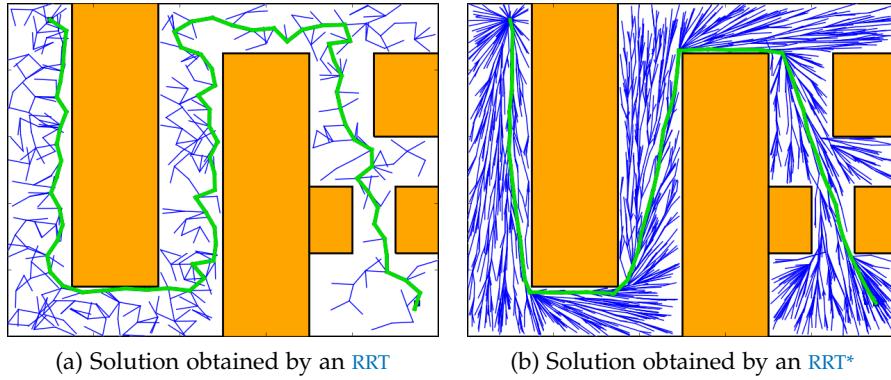


Figure 27: Start-to-goal query solution for a point-like system using both (a) the RRT and (b) the RRT* algorithms. The simulated scenario is composed of a series of polygonal obstacles presented in orange, the safe or collision-free space in red, the tree branches in blue, and the solution path in green.

Nonetheless, even if the shape checked for collision is enlarged, it does not guarantee that a vehicle can accurately follow the path. Figure 28 depicts this situation in a simulated scenario composed of two obstacles that create a narrow passage. In this scenario, two different start-to-goal queries were defined in such a way that required the solution path to be amidst the obstacles. Such paths were calculated by an RRT* algorithm under geometric constraints. In both cases, a simulated Sparus II AUV attempted to follow the path, however new risky situations appeared when conducting turning maneuvers. This occurred because the planner, over $\mathcal{C} = \mathbb{R}^2$, does not consider the vehicle motion limits (constraints) but, instead, it assumes the capability of instantaneously changing the vehicle direction of motion. It is also important to highlight that despite the fact that the executions of the missions were collision-free, the AUV surge speed and turning rate could have been established in a way that would have made the vehicle collide with the obstacles.

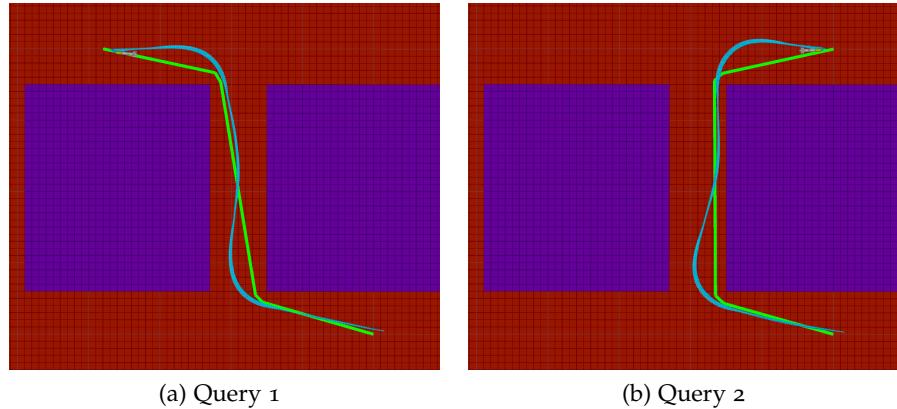


Figure 28: Simulated environment composed of two obstacles that create a narrow passage. Two different start-to-goal query solutions that were calculated by an RRT* algorithm under geometric constraints are shown in green. A simulated Sparus II AUV attempted to follow the paths, however the resulting vehicle trajectory (in light blue) differs from the one calculated, thus leading to risky situations when conducting turning maneuvers.

3.4.2 Conducting Missions under Motion Constraints

As explained along previous sections, a more appropriate formulation to limit unexpected vehicle trajectories when following a path is to use a planner that incorporates motion constraints. In such a case, the planner must not only specify the position and orientation, but also use the vehicle equation of motion. For the particular case of a torpedo-shaped AUV, this chapter explained two alternatives to incorporate Eq. (1) into the motion planner. In order to compare the approaches, both were used to solve the same start-to-goal queries presented in Fig. 28. For the first alternative, Figures 29a and 29b depict a simulated Sparus II AUV following solution paths that were calculated by an RRT algorithm that mathematically integrates Eq. (1). In this case, the RRT algorithm does not provide any guarantee for optimality.

The second alternative uses the RRT* algorithm and the Dubins curves, which work as a steering function equivalent to Eq. (1). Figures 29c and 29d depict how, with this approach, the simulated Sparus II AUV follows (near) optimal paths of minimal length. Although this latter approach generates better and more feasible solution paths, there are also some risky states; these were generated because the planner tried to provide the shortest paths, which were close to nearby obstacles. This specific issue about the safety of the resulting path is addressed in detail in Chapter 5.

A final remark about the content of this chapter is the fact that all cases assume an AUV that navigates at a constant depth. Even though this assumption is valid in several applications, there are others, however, that require to vary the vertical position of the vehicle. For this reason, the next chapter covers the extension of this latter approach (using Dubins curves) for dealing with 3D workspaces.

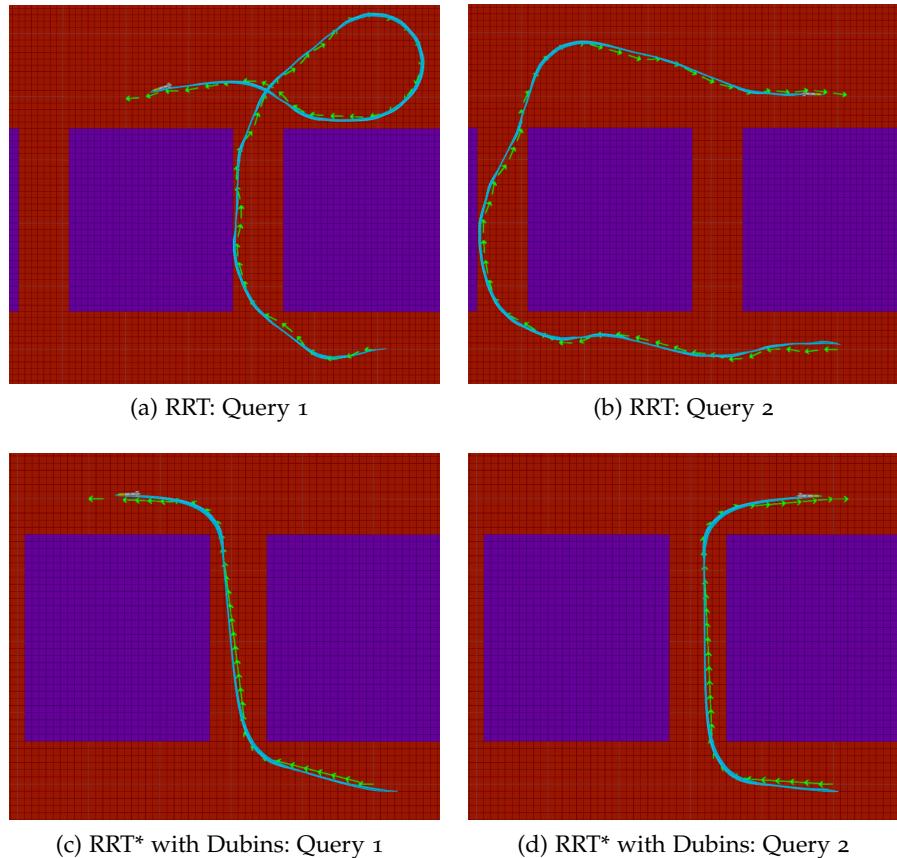


Figure 29: Simulated environment composed of obstacles that create narrow passages. Sparus II AUV follows the calculated paths. (a), (b) Two different start-to-goal query solutions that were calculated by an RRT algorithm under motion constraints are shown in green. In this case, the RRT algorithm does not provide any guarantee for optimality. (c), (d) The same start-to-goal queries were solved by an RRT* algorithm with Dubins curves as steering function. Solutions are shown in green.

PLANNING 3D MOTIONS FOR TORPEDO-SHAPED AND PROPELLER-DRIVEN AUVS

Although there is an important number of AUV applications in which the vehicle navigates at a constant altitude or depth, and therefore a 2D path-/motion planning approach is valid, there are other scenarios in which the vehicle is required to conduct 3D motions (see Chapters 1 and 2). Chapter 3 presented two alternative approaches to plan collision-free paths, which take into account the motion constraints of a torpedo-shaped AUV. While both approaches are limited to plan 2D paths, the one that uses Dubins curves also allows to obtain near-optimal solutions.

In order to define a correct strategy for planning 3D AUV paths, it is necessary to firstly establish the specific vehicle type, and then to understand its main motion characteristics. Underwater gliders, for instance, always require 3D motion planners and controllers, especially if considered their propulsion mechanism and the kind of trajectories they follow (see Chapter 1, Fig. 2). However, given that they operate in open sea areas, these vehicles do not usually have to deal with obstacles, narrow passages, or high-relief environments. Instead, their motion planners are more focused on coping with external perturbations such as ocean currents, which may deviate the glider from the desired trajectory. This makes that, in most cases, motion planning becomes a trajectory planning problem, which must consider the ocean forecast in order to minimize the effect of such perturbations [109, 118].

Propeller-driven AUVs, on the other hand, are equipped with thrusters that increase their maneuverability for 2D and 3D motions. This chapter discusses the 3D first-order model, which approximately describes the kinematic behavior of a torpedo-shaped and propeller-driven AUV. It also analyses the motion constraints involved, and how they can be included into a extended version of the Dubins curves approach presented in the previous chapter.

4.1 3D FIRST-ORDER MOTION MODEL

For 3D motions, six different variables specify the vehicle's position and orientation, which means that the C-Space is $\mathcal{C} = \text{SE}(3) = \mathbb{R}^3 \times \text{SO}(3)$ (see Fig. 30). Dynamic models, which normally consider all the 6 variables, have also been used in some AUV path-planning applications [125, 130]. Nonetheless, and as it has been already mentioned, their high computational cost make them an inappropriate alternative for the scenarios proposed in this thesis.

As occurs with 2D AUV motions, there exist different kinematic models that could be used for describing 3D motions. They mainly vary according to the AUV motion mechanism and its corresponding constraints. Two different cases will be explained in the following sections, one of which has been employed for planning collision-free paths for the Sparus II AUV.

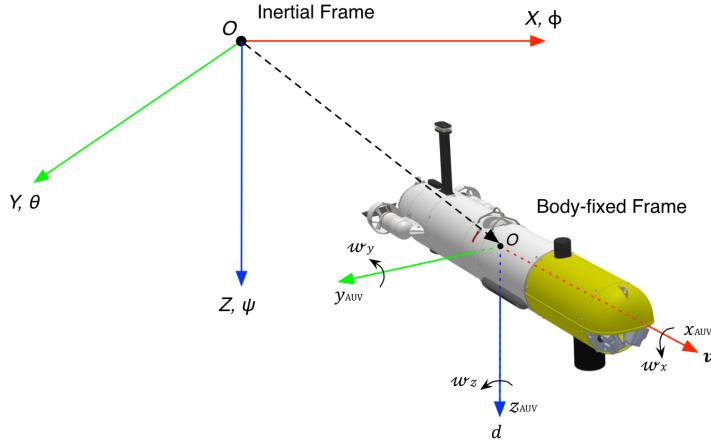


Figure 30: Perspective view of the Sparus II AUV, including the inertial and body-fixed frames, and the 6D vehicle state and control variables.

4.1.1 Vertical Motion Constraints

As explained in Section 3.1, a torpedo-shaped and propeller-driven AUV that conducts 2D motions, is generally subject to differential (motion) constraints. These constraints prevent the vehicle from laterally moving by only allowing it to turn while moving forward or backward, or to spin over when it is equipped with two back thrusters, for example as occurs with the Sparus II. When analysing 3D motions for this kind of vehicles, such limitations must be combined with those imposed by the mechanism that permits the vehicle ascending and descending.

Since the AUV volume remains constant during a mission, it is necessary to increase the downward force in order to make the vehicle vary its navigation depth. For doing so, there are different submerging mechanisms that can be classified into two groups. 1) Those vehicles with only one propulsion force in the back, which must be distributed into forward and downward components. This can be done either with inner ballast systems or with external foreplanes. 2) Those vehicles with two propulsion forces, one for forward motion, and an independent one for submerging, which is commonly generated by one or more vertical thrusters.

From the kinematics perspective, both groups generate different kind of motion constraints. The former group's vehicles descend and ascend with a non-zero pitch angle. In this case, the maximum surge speed and the pitch angles establish the maximum descending and ascending speeds. The second group's, on the other hand, are normally designed and trimmed for diving with a near-zero pitch angle. In this case, the maximum descending and ascending speeds are determined by the vertical thruster(s) specifications. In both cases, the vehicles are normally equipped with additional aft planes to generate turning maneuvers, and to minimize the vehicle roll. These two approaches result into different mathematical equations that describe the vehicle kinematics.

4.1.2 Mathematical Formulation

A torpedo-shaped AUV with non-zero pitch vertical motion, can be kinematically described with a 6-DOF model as follows [123]:

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} \cos(\psi) \cos(\theta) & 0 & 0 & 0 \\ \sin(\psi) \cos(\theta) & 0 & 0 & 0 \\ -\sin(\theta) & 0 & 0 & 0 \\ 0 & 1 & \sin(\phi) \tan(\theta) & \cos(\phi) \tan(\theta) \\ 0 & 0 & \cos(\phi) & -\sin(\phi) \\ 0 & 0 & \sin(\phi) \sec(\theta) & \cos(\phi) \sec(\theta) \end{bmatrix} \begin{bmatrix} v \\ w_x \\ w_y \\ w_z \end{bmatrix} \quad (2) \\ &= \begin{bmatrix} v \cos(\psi) \cos(\theta) \\ v \sin(\psi) \cos(\theta) \\ -v \sin(\theta) \\ w_x + w_y \sin(\phi) \tan(\theta) + w_z \cos(\phi) \tan(\theta) \\ w_y \cos(\phi) - w_z \sin(\phi) \\ w_y \sin(\phi) \sec(\theta) + w_z \cos(\phi) \sec(\theta) \end{bmatrix}, \end{aligned}$$

where $\mathbf{q} = [x, y, z, \phi, \theta, \psi]^T$ corresponds to the system state that includes its 3D position and orientation with respect to an inertial reference frame, and $\dot{\mathbf{q}} = [\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ is the first time derivative that depends on the state itself and the control inputs, i.e., linear/surge speed (v) and turning rates around the different vehicle axis (w_x, w_y, w_z). Here the vehicle is subject to two non-holonomic constraints, which are established on the linear velocities along the vehicle y and z directions. This full kinematic model presented in Eq. (2), has been used in trajectory planning applications [106]. However, it can also be simplified if the vehicle is assumed to maintain a near-zero roll angle ($\phi \approx 0$):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v \cos(\psi) \cos(\theta) \\ v \sin(\psi) \cos(\theta) \\ -v \sin(\theta) \\ w_y \\ w_z \cos(\phi) \sec(\theta) \end{bmatrix} \quad (3)$$

On the other hand, torpedo-shaped and propeller-driven AUVs with near-zero pitch motion ($\theta \approx 0$), i.e., those with an independent propul-

sion force for vertical motion, must be described with a different kinematic model as follows:

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} \cos(\psi) & 0 & 0 & 0 & 0 \\ \sin(\psi) & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} v \\ d \\ w_x \\ w_y \\ w_z \end{bmatrix} \\ &= \begin{bmatrix} v \cos(\psi) \\ v \sin(\psi) \\ d \\ w_x \\ w_y \sin(\phi) + w_z \cos(\phi) \end{bmatrix}, \end{aligned} \quad (4)$$

where, once again, $\mathbf{q} = [x, y, z, \phi, \psi]^T$ corresponds to the system state that includes its position and orientation with respect to an inertial reference frame. Furthermore, $\dot{\mathbf{q}} = [\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\psi}]^T$ is the first time derivative that depends on the state itself and the control inputs, which in this case are the linear speeds (surge and heave (v, d)) and the turning rates (w_x, w_z). Here the vehicle is subject to one non-holonomic constraint, which is established on the linear velocity along the vehicle y direction. Equation (4) can also be simplified if the vehicle is assumed to maintain a near-zero roll angle ($\phi \approx 0$):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v \cos(\psi) \\ v \sin(\psi) \\ d \\ w \end{bmatrix}, \quad (5)$$

which can be written in the form $\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \mathbf{u})$, where $\mathbf{q} = [x, y, z, \psi]^T$, $\dot{\mathbf{q}} = [\dot{x}, \dot{y}, \dot{z}, \dot{\psi}]^T$, and the control input $\mathbf{u} = [v, d, w]^T$ that corresponds to the surge speed (v), the heave speed (d), and the rate of turn (w).

4.2 TREE EXPANSION USING DUBINS CURVES FOR 3D MOTIONS

Different authors have used Eq. (5) to generate 3D motions for torpedo-shaped and propeller-driven AUVs. For instance, Heo et al. presented an approach to build a tree of collision-free and feasible configurations, over which a final path is found with the A* algorithm [45]. The tree expansion is done by directly integrating Eq. (5), similarly as it was explained in Sec. 3.2. However, although A* finds the best solution over the available paths, it requires an estimate of the cost to the goal, and this may be hard to obtain, especially in partially known environments.

An alternative to find better 3D solution paths is to use the RRT* algorithm, however it requires a steering function for improving the configurations connections (see Sec. 2.6.3). Section 3.3 introduced the use of the

Dubins curves as a steering function for 2D paths, which can also be extended to 3D motions. To do so, let us consider the AUV has an independent propulsion force for its vertical motion, and therefore its kinematics can be described with Eq. (5). In this case, it is possible to affirm that the motion in the horizontal plane can be decoupled from the heave (vertical) motion. This allows to directly deal with the former component as stated in Chapter 3, while treating the latter component as an additional constraint over an extended C-Space, $\mathcal{C} = \text{SE}(2) \times \mathbb{R}$. This motion separation is not new for underwater vehicles, since it has also been done for dynamic models formulations [96].

On this basis, the path between two states for the 3D decoupled kinematics of our vehicle is the one that connects their components in the horizontal plane using Dubins curves, and their vertical components with a linear interpolation. The vertical motion must have a gradient no greater than the maximum ascending/descending AUV speed, thus meeting the vehicle motion capabilities. In order to complete the steering function required by the RRT* algorithm, it is also necessary to establish a metric ρ that allows determining the distance between two states. In this case, one alternative is to combine the lengths of the horizontal and the vertical paths as:

$$\rho(q_i, q_j) = \rho_h(q_i, q_j) + \rho_v(q_i, q_j), \quad (6)$$

where the first term corresponds to sum of the n Dubins sectors lengths, which are required to connect q_i and q_j in the horizontal plane, and the second term corresponds to the distance in the heave (vertical) motion:

$$\rho_v(q_i, q_j) = |q_{iz} - q_{jz}| \quad (7)$$

All this together, i.e., the mathematical formulation, the use of the Dubins curves over a 4D space, and the corresponding metrics for calculating distances between different configurations, establish an alternative approach to calculate 3D motions for torpedo-shaped AUVs under motion constraints. Note that in the 3D case, we no longer have an analytical solution for the shortest path between two poses. However, under the assumption that the AUV moves with a constant surge speed, the RRT* algorithm will, with the distance metric defined above, converge on the shortest path that satisfies the descent speed constraint. The next section presents different simulation tests that seeks to prove the validity of this approach.

4.3 RESULTS

This section presents and discusses simulations of the Sparus II AUV conducting 3D start-to-goal missions in different scenarios. Similarly as it was done in the previous chapter, the solution paths have been calculated under both geometric and differential constraints, in order to better understand the importance of considering the AUV motion capabilities when navigating in complex environments. In all cases, the environment was assumed to be pre-explored, i.e., a map of the surroundings was available.

4.3.1 Conducting 3D Missions under Geometric Constraints

As it was explained before, Sparus II is a torpedo-shaped AUV equipped with an independent vertical thruster that allows decoupling the horizontal and vertical motions. This means that the vehicle is capable of descending or ascending without needing to move forward or backward. For example, if a start-to-goal query is defined as $q_{start} = [x_0, y_0, z_0, \psi_0]$ and $q_{goal} = [x_0, y_0, z_0 + depth, \psi_0]$, and the planner only takes into account geometric constraints, the solution path should be one that connects both configurations with a straight descent trajectory (see Fig. 31).

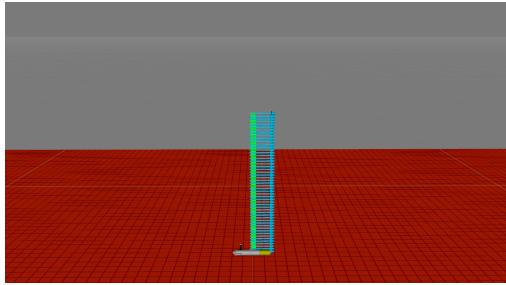


Figure 31: Start-to-goal query that is defined as $q_{start} = [x_0, y_0, z_0, \psi_0]$ and $q_{goal} = [x_0, y_0, z_0 + depth, \psi_0]$. The solution path (in green) is calculated only taking into account geometric constraints, and requires the AUV to follow a straight descent trajectory (in light blue).

Let us define now a start-to-goal query where $q_{start} = [x_0, y_0, z_0, \psi_0]$ and $q_{goal} = [x_0 + distance, y_0, z_0 + depth, \psi_0]$, which means that the vehicle must conduct forward and vertical motions simultaneously. Furthermore, let us assume that the vehicle is required to navigate at a constant surge speed (v), as occurs in several AUV applications. Under such constraints, the time required to travel the horizontal distance can be calculated as $t_h = \frac{distance}{v}$, while the descending speed (d) can be adjusted up to a maximum value (d_{max}), which can be calculated as $d = \frac{depth}{t_h}$. This kind of tasks could also be tackled with geometric constraints, however there may be situations in which the resulting path cannot be followed by the AUV. For example, if t_h allows enough time to descend the desired vertical distance, i.e., $d \leq d_{max}$, the vehicle will successfully complete the task, otherwise it may not reach the desired depth (see Fig. 32).

Previous examples did not require the AUV to change its position along the sway direction, and therefore a constant orientation was assumed during the mission. Nonetheless, real missions, especially those intended for this thesis, present more challenging scenarios. In such cases, the vehicle probably needs not only to vary its vertical position, but also to change its orientation in order to conduct turning maneuvers.

Figure 33 depicts a test scenario that is composed of two blocks (12m \times 12m \times 8m), which are separated by a passage of 4m. Over this simulated environment, a start-to-goal query solution and its execution are presented, where $q_{start} = [x_0, y_0, z_0]$ and $q_{goal} = [x_1, y_1, z_1]$, which means that the orientation was not taken into account. The 3D solution path was calculated using an RRT* algorithm under geometric constraints, and it was followed by the simulated Sparus II. It can be observed that, although the change of depth did not require a descending speed greater than the ve-

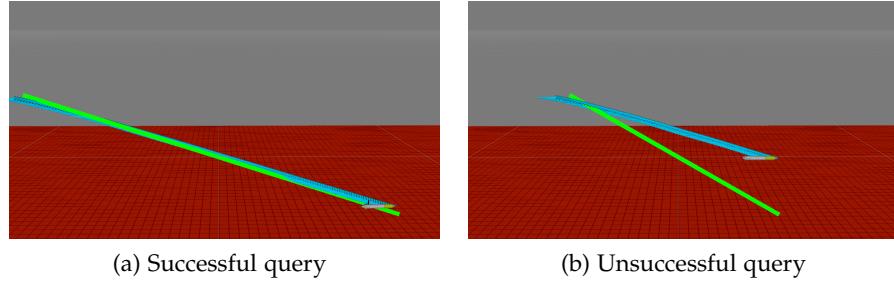


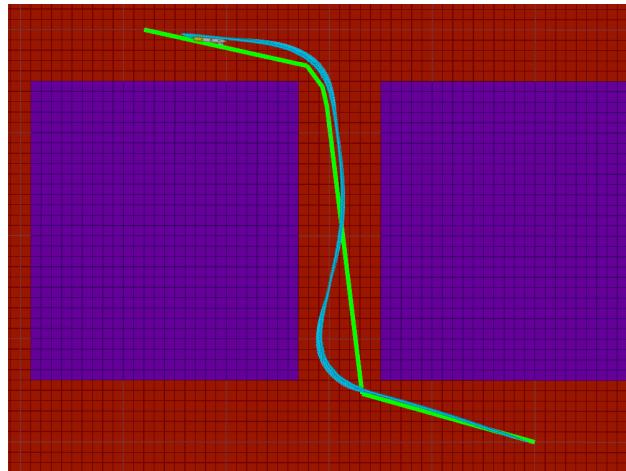
Figure 32: Start-to-goal queries that are defined as $q_{\text{start}} = [x_0, y_0, z_0, \psi_0]$ and $q_{\text{goal}} = [x_0 + \text{distance}, y_0, z_0 + \text{depth}, \psi_0]$. The solution path (in green) is calculated only taking into account geometric constraints, and requires the AUV to conduct forward and vertical motion (in light blue). The Sparus II is assumed to have a constant surge speed $v = 0.6\text{m/s}$, and a maximum descending speed $d_{\text{max}} = 0.2\text{m/s}$. (a) Successful query with distance = 20m and depth = 6m. (b) Unsuccessful query with distance = 10m and depth = 6m.

hicle capabilities (i.e., $d < d_{\text{max}}$), the vehicle followed a trajectory that was not initially planned, especially when conducting turning maneuvers. This was mainly due to the fact that the planner did not consider the vehicle’s motion constraints. Next sections will demonstrate how the use of the extended 3D Dubins curves approach, explained before, can overcome these situations.

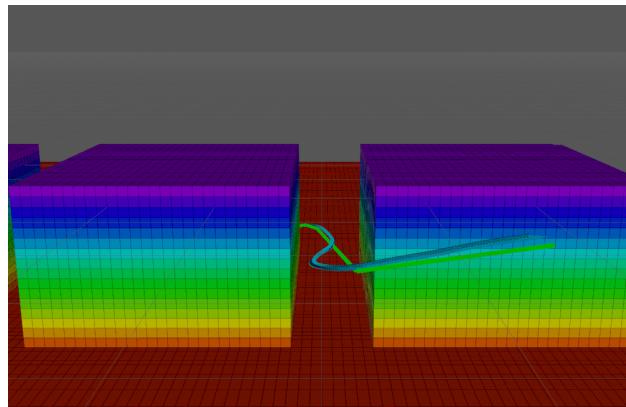
4.3.2 Conducting 3D Missions under Motion Constraints

The previous section presented as a first example, a simulated task in which the Sparus II AUV was required to conduct a vertical motion while attempting to maintain its orientation invariant (see Fig. 31). Let us now assume that the vehicle has to solve the same task, i.e., a start-to-goal query defined as $q_{\text{start}} = [x_0, y_0, z_0, \psi_0]$ and $q_{\text{goal}} = [x_0, y_0, z_0 + \text{depth}, \psi_0]$; but this time the vehicle has to navigate with a constant surge speed v and a descending speed up to a maximum value d_{max} . Under such constraints, the motion planner must find a solution that combines turning and descending maneuvers to reach the final position and orientation. This problem can be solved with the extended Dubins curves approach that has been explained throughout previous sections. One possible solution, where the vehicle circumnavigates while descending to reach the desired depth and orientation, can be observed in Fig. 34. Furthermore, this solution is near-optimal under the assumption of a constant surge speed.

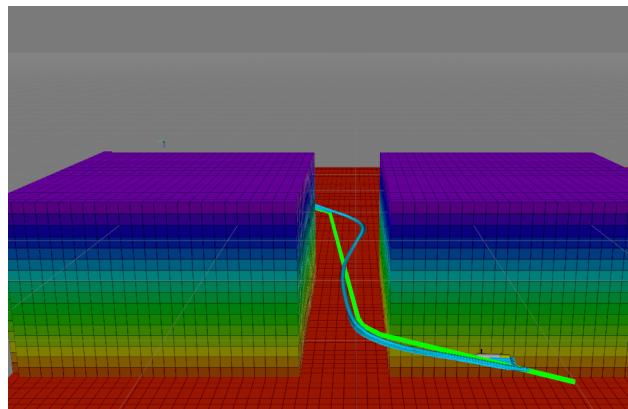
Figure 35 presents the same test scenario composed of two blocks separated by a narrow passage used in the previous section. Similarly as occurred in the simulation presented in Fig. 33, the start-to-goal query requires the vehicle to completely change its 3D position, but this time it also specifies the desired AUV heading. This task, for which the vehicle is also under the surge and descending speeds constraints mentioned before, consists of a $q_{\text{start}} = [x_0, y_0, z_0, \psi_0]$ and a $q_{\text{goal}} = [x_1, y_1, z_1, \psi_1]$. This time the resulting vehicle trajectory coincides with the planned path, even when conducting turning maneuvers.



(a) Query top view



(b) Query front view



(c) Query back view

Figure 33: Simulated environment composed of two blocks ($12\text{m} \times 12\text{m} \times 8\text{m}$), where a start-to-goal query is defined as $q_{\text{start}} = [x_0, y_0, z_0]$ and $q_{\text{goal}} = [x_1, y_1, z_1]$. The solution path (in green) is calculated only taking into account geometric constraints. A simulated Sparus II AUV attempted to follow the path, however the resulting trajectory (in light blue) differs from the one calculated, thus leading to risky situations when conducting turning maneuvers.

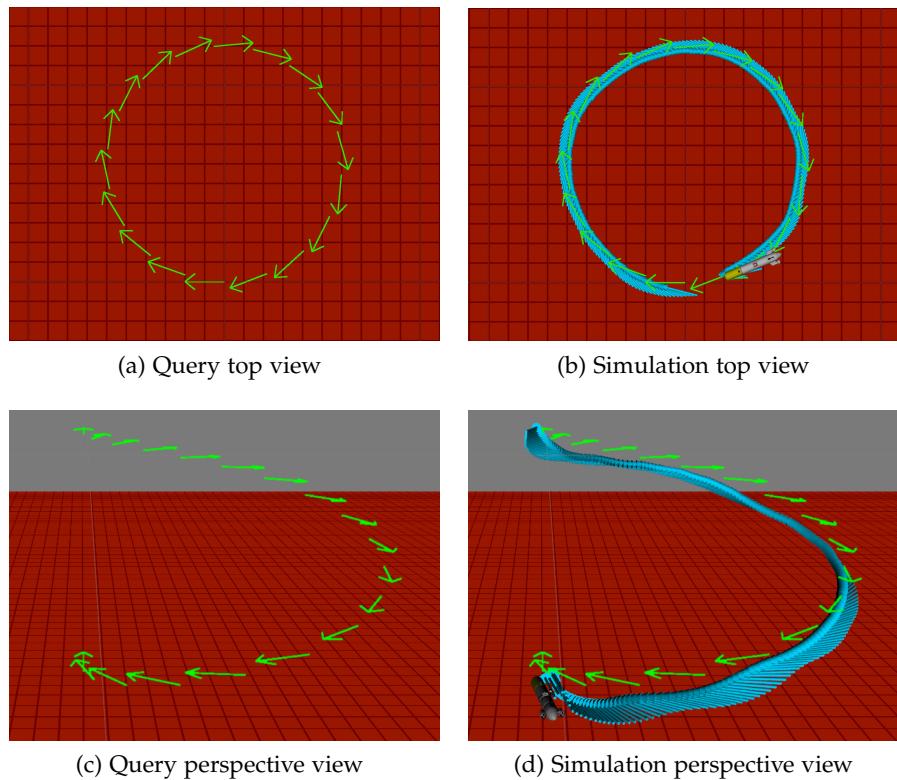
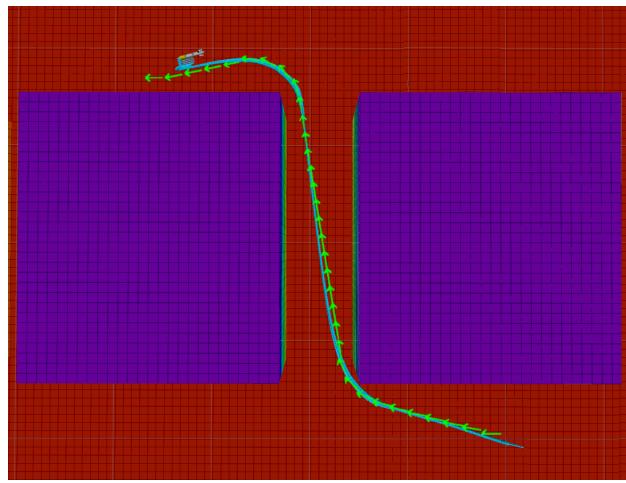
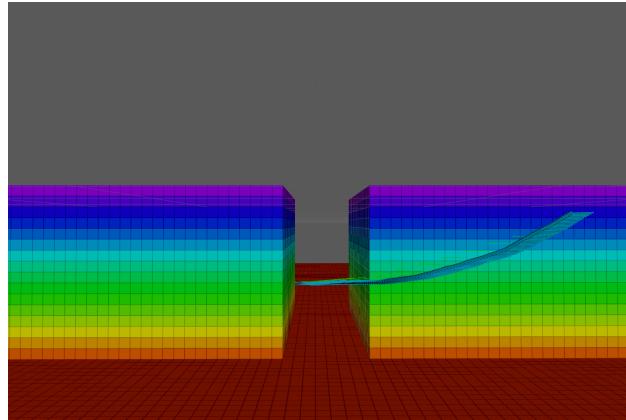


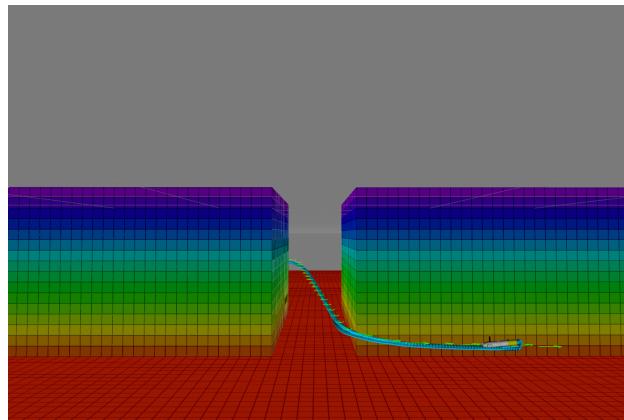
Figure 34: Start-to-goal query that is defined as $q_{\text{start}} = [x_0, y_0, z_0, \psi_0]$ and $q_{\text{goal}} = [x_0, y_0, z_0 + \text{depth}, \psi_0]$. The AUV is assumed to navigate at a constant surge speed v , and a descending speed up to d_{\max} . The solution path (in green) is calculated with an RRT* that uses the extended Dubins curves, which incorporate the vehicle 3D motion capabilities. The simulated Sparus II successfully describes a helical trajectory (in light blue) that follows the calculated path.



(a) Query top view



(b) Query front view



(c) Query back view

Figure 35: Simulated environment composed of two blocks ($12\text{m} \times 12\text{m} \times 8\text{m}$), where a start-to-goal query is defined as $q_{\text{start}} = [x_0, y_0, z_0, \psi_0]$ and $q_{\text{goal}} = [x_1, y_1, z_1, \psi_1]$. The solution path (in green) is calculated by an RRT* that uses the extended Dubins curves approach. A simulated Sparus II AUV describes a trajectory (in light blue), which does not differ significantly from the one calculated, even when conducting turning maneuvers.

Finally, in order to completely assess the extended Dubins approach for 3D motions, Figure 36 depicts a start-to-goal mission in a high-relief environment. The requested query has been defined in such a way, that any solution path would require the AUV to conduct turning, ascending and descending maneuvers. The resulting path and its corresponding execution by the simulated Sparus II AUV was successful, thus completely proving the presented approach capabilities.

There are, however, some aspects that have to be addressed in order to make this approach really useful for the proposed applications. The first of them is that the intended usage scenarios involve generally unexplored spaces, which means that a map is not available. A second aspect is the safety of the vehicle, which can be compromised when navigating in close proximity to the obstacles. These and other aspects, as well as the different strategies proposed to cope with them, will be discussed in detail in the next chapter.

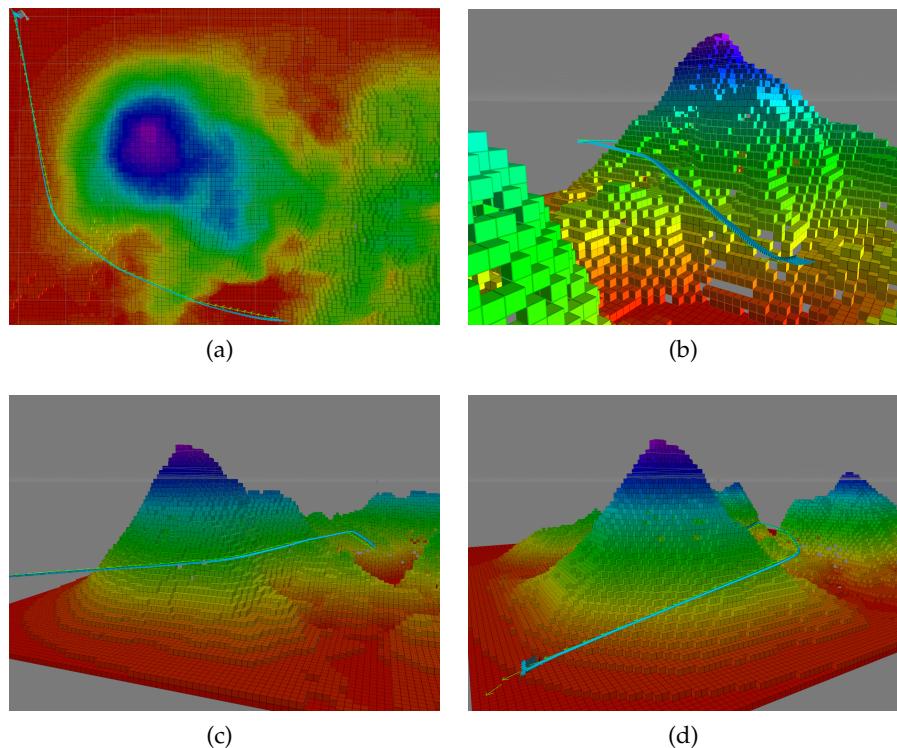


Figure 36: High-relief simulated environment composed of multiple seamounts, where a start-to-goal query is defined as $q_{\text{start}} = [x_0, y_0, z_0, \psi_0]$ and $q_{\text{goal}} = [x_1, y_1, z_1, \psi_1]$. A simulated Sparus II AUV describes a trajectory (in light blue), which attempts following a solution path that. This path is calculated by an RRT*, which uses the extended Dubins curves approach.

FRAMEWORK FOR ONLINE MOTION PLANNING IN UNEXPLORED ENVIRONMENTS

New and potential AUV applications that were presented in Chapter 1, establish most of the requirements and objectives for this thesis. One of them, and probably the most relevant, is the necessity of incrementally building a map of the surroundings, while simultaneously (re)planning the collision-free path to the goal. This characteristic would allow the vehicle to navigate through unexplored environments, as well as to overcome part of the navigation inaccuracy.

In order to endow an AUV with these capabilities, this chapter presents a motion planning framework, which solves start-to-goal queries online for an AUV that operates in unexplored environments. The framework is composed of three functional modules. 1) A *mapping* module that builds an occupancy map of the environment using on-board perception sensors. 2) A *planning* module that generates safe (collision-free) and feasible paths online. 3) A *mission handler* that works as a high-level coordinator that exchanges information with the other two modules and the AUV controllers. Figure 37 depicts the proposed framework, and how its functional modules are connected to one another. The following sections will explain in detail each of these modules.

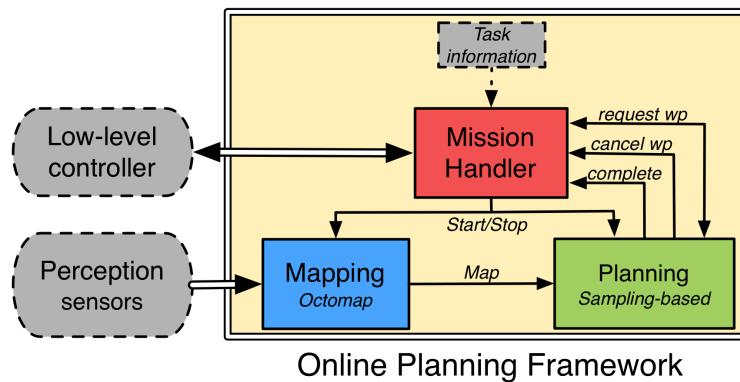


Figure 37: Framework for online AUV motion planning.

5.1 MISSION HANDLER

The first functional module that constitutes the path/motion planning framework is the *mission handler*. This module is in charge of controlling and coordinating the other modules (*mapping* and *planning*). It also verifies whether the AUV is prepared to start solving and conducting a task. To do so, this module communicates with other functional modules in the vehicle to verify both that navigation data is correctly being generated, and that the vehicle low-level controllers are not conducting any safety maneuvers. After completing this initial checking stage, the *mission handler*

starts requesting waypoints from the *planning* module, which, after being received, are adapted and sent to the vehicle low-level controllers. This module is also responsible for cancelling any ongoing waypoint if it is notified by the *planning* module. This latter situation will be explained in the following sections.

5.2 MODULE FOR INCREMENTAL AND ONLINE MAPPING

The *mapping* module incrementally builds a representation of the environment. To do so, it uses data that can be obtained from different kinds of perception sensors such as multibeam sonars, mechanically scanned profiling sonars, echosounders, etc. These sensors provide range information about nearby obstacles that, combined with the vehicle's position and orientation, allows establishing the free and occupied space with respect to an inertial coordinate frame. In order to represent this data, this module uses an octree-based framework called Octomap [50]. This representation has three main characteristics that efficiently model such volumetric information.

The first characteristic is the probabilistic state representation. This allows not only to modify the map when updated environment information is available, but also protects it from noisy measurements. This latter feature is possible because the state of a particular position over the map considers previous information, and calculates its new value according to probabilistic functions. The second characteristic is the capacity of representing unexplored areas, which can be relevant for guiding the planner over the exploration of unknown environments. The last characteristic allows an Octomap to be enlarged or extended as demanded in a computationally efficient way.

Figure 38 shows a breakwater structure, which is located in the harbor of Sant Feliu de Guíxols (Spain), and its representation with an Octomap that has been built using real-world multibeam sonar data obtained by a surface vessel. This is one of the test environments used during the development of this work for both simulated and real-world trials. Results presented in Chapters 3 and 4, for instance, used a virtual scenario and the corresponding Octomap that are based on this breakwater structure (see Fig. 39).

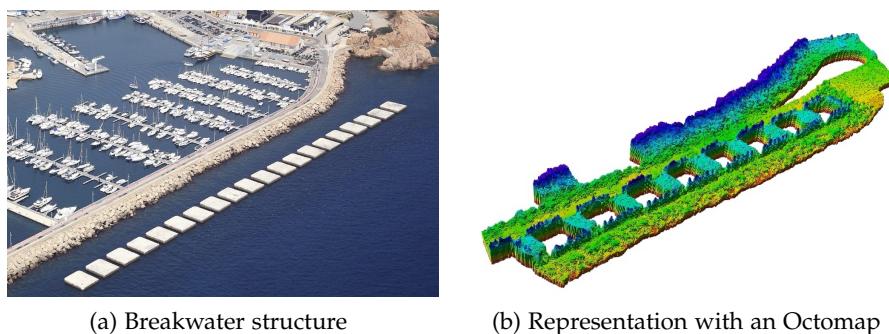


Figure 38: Harbor of Sant Feliu de Guíxols (Spain). (a) Aerial view of the breakwater structure. (b) Octomap representation that has been created with real-world data acquired with a multibeam sonar.

In order to determine if the vehicle is or would be under collision in a certain position, the Octomap has to be verified in an equivalent volume that contains the vehicle. To do so, all voxels within such a volume have to be considered as non-occupied. This validation can be simplified by representing the AUV volume with one solid such as a cylinder or a hexahedron (see Fig. 39). This collision checking routine between an Octomap and a solid can be efficiently performed by the flexible collision library (FCL) [32].

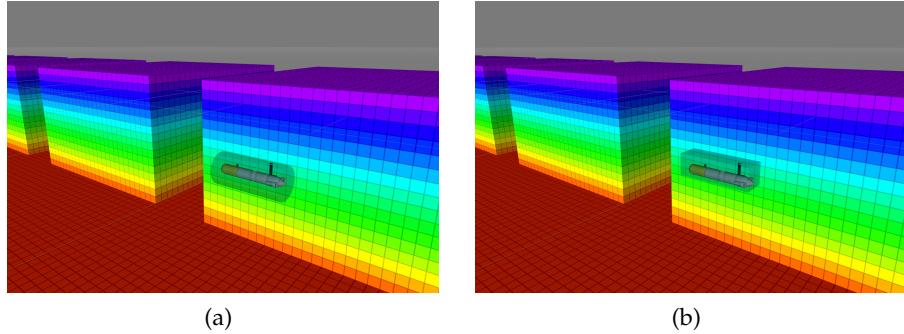


Figure 39: Collision checking using Octomaps is done by evaluating the Octomap voxels that contain the equivalent volume of the AUV. In order to simplify this validation routine, the AUV volume can be represented by a solid such as a (a) cylinder or a (b) hexahedron.

5.3 MODULE FOR (RE)PLANNING PATHS ONLINE

The third functional module of the proposed framework is the *planning* module, which is in charge of calculating a collision-free and feasible path for the AUV. For doing so, this module receives a query that is specified with a start configuration (q_{start}) and a goal configuration (q_{goal}), and other parameters, such as the available computing time and the minimum valid distance to the goal. Furthermore, given that the vehicle navigates in an unexplored (or unknown) environment, this module must continuously verify and repair (if necessary) the path from the current vehicle configuration to q_{goal} . In order to calculate a path under such constraints, this module contains a modified implementation of the RRT* algorithm.

This RRT* variant not only permits to progressively improve the solution, but it has also been extended to safely navigate while efficiently re-planning the path as the environment is incrementally explored. Our extensions include an optimization function that combines collision risk and path length; a strategy to avoid unnecessary and expensive state checking routines; and the reuse of the last best known solution as a starting point for an anytime planning approach. These new characteristics are presented and explained in detail in the following sections.

5.3.1 Planning Safe Paths using Risk Functions

Conducting underwater missions with AUVs is a challenging task, especially when the vehicle is kinematically constrained, as it was explained in Chapters 3 and 4. However, including motion constraints to the path plan-

ner may not suffice to minimize the risk of colliding with nearby obstacles. This is especially true when the vehicle is exposed to external perturbations, which do not permit the AUV to accurately follow the calculated path. Figure 40, for instance, shows a feasible path, i.e., one that considers the vehicle motion constraints. However, as the solutions are only asymptotically optimized with respect to their length, they lead the vehicle close to nearby obstacles, especially in turning maneuvers.

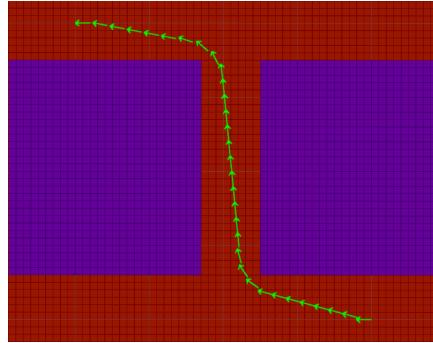


Figure 40: A start-to-goal query solution that was calculated by an RRT* algorithm with Dubins curves as steering function. The solution path requires the vehicle to move close to nearby obstacles when conducting turning maneuvers.

Bearing this in mind, the objective would be now to extend the planner to not only plan feasible paths, but also to attempt to minimize the risk of colliding with the surroundings. To accomplish this, one alternative is to provide an optimization objective to the planner that combines the length and the safety of the path. The following sections present different approaches to achieve this.

5.3.1.1 Path Length + Clearance

A first and straightforward option is to maintain a minimum safe distance, or clearance, to the obstacles. For doing so, it is necessary to establish a weighted metric that combines path length and clearance in order to minimize detrimental effects in the path quality. This allows us to define the associated cost of each configuration when planning with an RRT* algorithm. However, this approach has two main drawbacks, its high computational cost and the need to correctly specify weights to obtain a balanced metric, which is a non-trivial problem [120].

5.3.1.2 Risk Zones

Including clearance calculation is especially expensive for sampling-based path/motion planning methods, since it has to be performed for each sampled configuration and its intermediate steps when connecting to the others (e.g., RRT* algorithm expansion). One alternative is to heuristically establish risk zones around the vehicle, as shown in Fig. 41.

The red and blue zones represent the collision and the non-risk zones, respectively, while the others (orange, yellow and green) have associated decreasing risk values as collisions move away from the vehicle position.

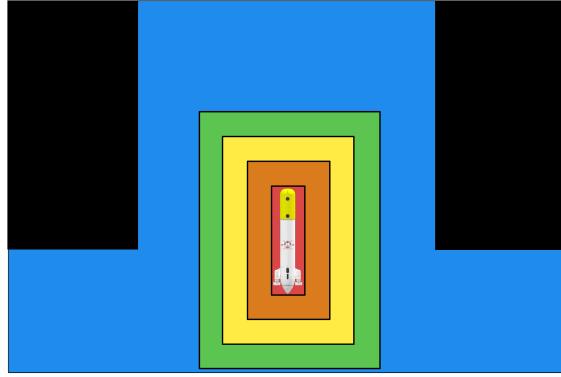


Figure 41: Sparus II AUV navigating between two obstacles (black). An example of the risk zones around the vehicle, and how the risk decreases as moves away from the vehicle can be observed (red means high risk, while blue means low risk).

Therefore, the risk associated with a given configuration q , can be represented by Eq. (8), where n is the number of zones, zone_{i-1} is closer than zone_i to the collision zone, $\text{risk}_{i-1} > \text{risk}_i > \dots > \text{risk}_n > 1$.

$$\text{Risk}(q) = \begin{cases} \text{risk}_1, & \text{if Collision}(\text{zone}_1) \\ \text{risk}_2, & \text{if Collision}(\text{zone}_2) \\ \dots, & \dots \\ \text{risk}_n, & \text{if Collision}(\text{zone}_n) \\ 1, & \text{if not Collision(any zone)} \end{cases} \quad q = [x, y, z, \psi]^T \quad (8)$$

In order to combine this function with the path length, the total accumulated cost associated with each configuration is calculated as the integral of risk with respect to distance, as presented in Eq. (9). Such a cost function not only combines the risk and the length associated with a path, but it also establishes the optimization criterion required in order to plan feasible and safe paths. A visual comparison between paths calculated using only the path length and those with risk zones can be observed in Fig. 42.

$$\text{Cost}(q) = \int_0^q \text{Risk}(q) dq \quad (9)$$

5.3.1.3 Risk Vectors

The previous approach attempts to penalize those configurations close to nearby obstacles by specifying a risk value, which depends on the affected zone. However, another alternative is to limit such evaluation to the directions defined by the possible maneuvers of the vehicle at the considered time. To do this, let us define three vectors in the straight and lateral motion directions. Now, instead of checking for complete zones, only points

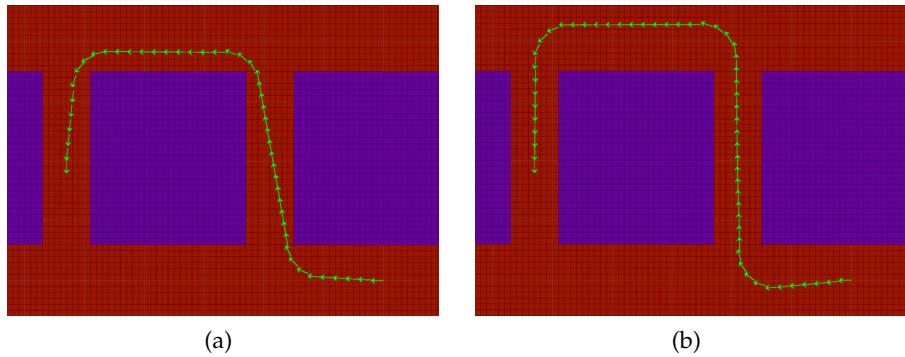


Figure 42: (a) RRT* expansion for a start-to-goal query using Dubins curves with path length as cost criterion. (b) RRT* expansion with risk zones cost criterion. It is worth noting that the resulting path is far from corners in turning maneuvers and attempts to stay in the middle of the corridor when navigating between two obstacles.

along those vectors will be checked for collision, assigning risk values with the same principle as done with risk zones, i.e., moving away from the vehicle decreases the risk (see Fig. 43). This approach is here called *risk vectors*. It is computationally a less expensive alternative, given that checking collision for single points is more efficient than doing the check for zones (multiple points), especially when using of Octomaps to represent the environment.

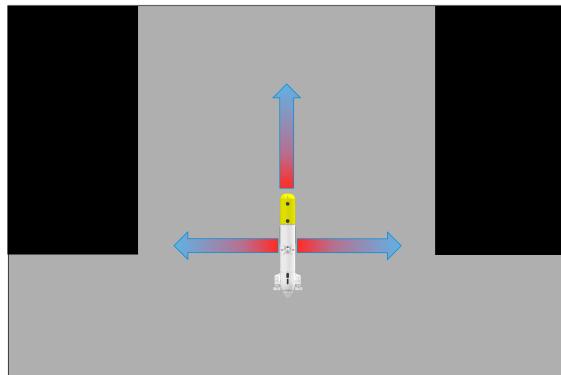


Figure 43: Sparus II AUV navigating between two obstacles (black). An example of the risk vectors, and how the risk decreases as moves away from the vehicle can be observed (red means high risk, while blue low risk).

Both the risk zones and the risk vectors have also been extended for 3D motions. In the case of the risk zones, the areas around the vehicle become volumetric shapes of increasing size (e.g., rectangular prisms, or cylinders), whereas with the risk vectors it is enough to add the corresponding vectors for the vertical motion (see Fig. 44). These alternatives to quantify the risk associated with a path have been evaluated and compared with each other. The results of this analysis will be discussed in Sec. 5.5.

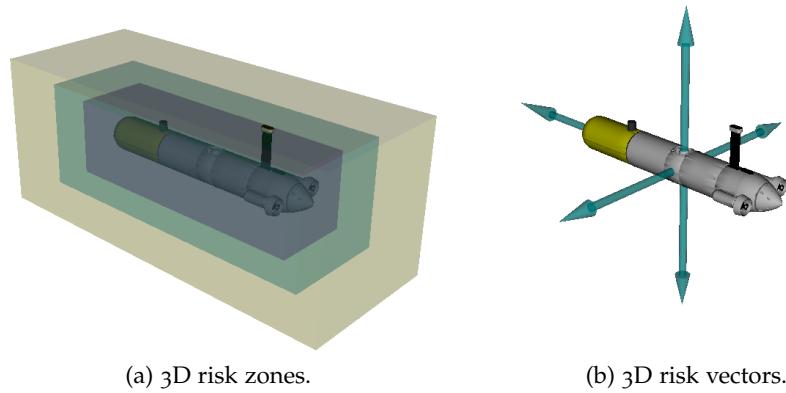


Figure 44: Extension of risk zones and risk vectors for 3D motions.

5.3.2 Opportunistic State Checking

Previous sections presented the estimation of the risk associated with a path, which is the first main characteristic of the proposed framework and its *planning* module. This section, on the other hand, introduces the second characteristic that seeks to optimize the computation time in missions where the environment is incrementally explored.

When conducting an autonomous mission without initial information of the surroundings, the *AUV* is required to incrementally map and continuously (re)plan collision-free paths according to its partial knowledge of the environment. Because of this, and assuming the *planning* module uses a sampling-based method such as the *RRT** algorithm, an important number of configurations (sampled or obtained after expanding the tree) are located in unexplored regions of the environment. In these situations, it is not only impossible but also unnecessary to attempt to determine if a configuration is at risk of collision. In order to compensate this, one alternative is to assume as safe (collision-free and minimum risk) any configuration that is out of the explored area. This can be efficiently determined when using Octomaps (as explained in Section 5.2). This strategy, proposed in this thesis, is called *opportunistic state checking*, and is inspired by the *lazy collision checking* (which was explained in Sec. 2.7.2.2).

Figure 45 depicts a simulation where the explored and free regions of the map are presented in light blue, the occupied ones are presented in purple, and the rest correspond to unexplored areas. With this visualization, it is possible to observe the importance of avoiding checking for collision of those configurations that are located in undiscovered regions. Section 5.5 will show different tests that demonstrate the advantages of using this strategy.

In this incremental mapping and planning approach, the tree expansion is periodically interleaved with updating the map, and such parts initially assumed as safe must be verified and discarded if found under collision as the vehicle explores the environment (a situation that can also be observed in Fig. 45). The next section describes and analyses two alternative mechanisms to check and reshape the tree.

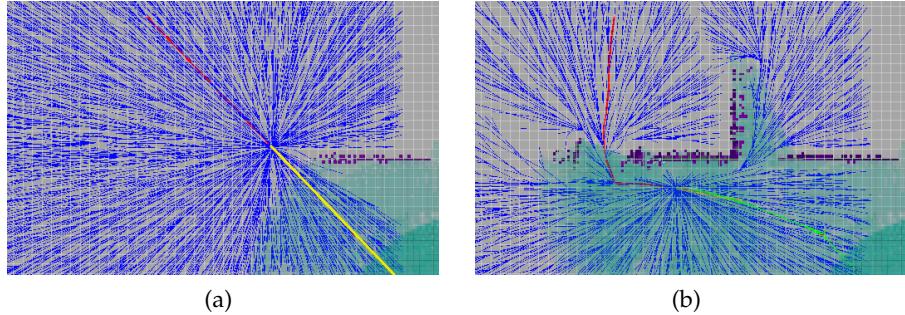


Figure 45: Sparus II AUV conducting an autonomous mission in the simulated breakwater structure scenario without an initial map. (a) The explored region, presented in light blue, expands as the vehicle moves towards the goal. It is important to notice that a significant part of the tree (dark blue) is located in unexplored areas of the workspace. (b) Those branches are initially assumed as safe (collision-free) until the corresponding region is explored, thus avoiding unnecessary collision-checking routines computation.

5.3.3 Reuse of the Last Best Known Solution

This section presents the *reuse of the last best known solution*, which is the third main characteristic of the proposed framework and its *planning* module. Such a strategy allows an incremental and tree-based path planner, such as the [RRT*](#) algorithm, to replan or reshape the solution path according to the new environment information perceived during the mission. However, in order to fully understand the benefits associated with this strategy, this section firstly explains another alternative that was initially used, but later discarded due to its high computational cost.

5.3.3.1 Pruning the Tree for Anytime Computation

One alternative for dealing with partially known environments is to grow a single tree of collision-free configurations (as any [RRT](#) variant does), while periodically pruning the tree. This process allows discarding those branches that result under collision after updating the map, thus obtaining a valid solution at *anytime* that is requested. This strategy has been used with approaches based on the [RRT](#) [7] and the [RRT*](#) [63]. At a first stage of this thesis, the *planning* module used a modified [RRT*](#) algorithm that was based on this pruning approach.

Like other [RRT](#)-based algorithms, the initially used variant consisted of two procedures, `sample` and `extend` (see Sec. 2.6.2.4). However, the former one had two main modifications (see Algorithm 9). The first one was intended to correct the tree according to the new elements discovered in the environment. To do so, and before sampling new configurations, the `updateTree` procedure traversed the tree using a [DFS](#) algorithm to check if any node or edge was under collision (line 2). If a new collision was detected, the corresponding subtree (i.e., those branches with nodes or edges under collision) was discarded. However, if the tree root was one of the nodes under collision, or if the path from the current vehicle configuration to the root was not feasible, the *planning module* informed the *mission*

handler to cancel the current path followed by the controller, and then it started again planning a new path from the current vehicle position. This latter situation occurred because the tree root always corresponded to the configuration (or position) that the vehicle was moving towards.

The purpose of the second modification in the sample procedure was to make the RRT* behave as an *anytime* algorithm. To do this, if the new configuration q_{new} resulted from the tree expansion met the specified minimum distance to the goal (line 7), it was added to a list of possible solutions (line 8). Furthermore, if the *mission handler* had requested a new waypoint after concluding the tree expansion, and there was at least one available solution stored in the list (line 10), the planner selected the solution with the minimum associated cost, sent the *mission handler* the configuration connected to the root of that solution (line 13), and pruned the tree in such a way that the configuration sent became the new tree root (line 14). During this *pruning* process, subtrees connected to the former root (excepting the one that corresponds to the new root) were discarded. The extend procedure, on the other hand, remained as originally proposed in [62].

Algorithm 9 : SampleAnytimeRRT*

Input :

q_{start} : Start configuration.

q_{goal} : Goal configuration.

\mathcal{C} : C-Space.

```

1 begin
2   updateTree()
3   while not stop_condition do
4      $q_{rand} \leftarrow \text{sampleConf}()$ 
5     result,  $q_{new} \leftarrow \text{extendRRT*}(T, q_{rand})$ 
6     if result  $\neq \text{TRAPPED}$  then
7       if dist( $q_{new}, q_{goal}$ )  $< \epsilon_{goal}$  then
8         updateBestSolution( $q_{new}$ )
9         solution_found  $\leftarrow$  true
10    if solution_found and wp_req then
11      result_path  $\leftarrow \text{getBestSolution}()$ 
12      new_root  $\leftarrow$  result_path[1]
13      sendWaypoint(new_root)
14      pruneTree(new_root)

```

To illustrate the behavior of this tree-pruning strategy when simultaneously mapping and planning collision-free paths, Figure 46 depicts a simulation of the Sparus II AUV conducting a mission in an environment that resembles the breakwater structure presented in Fig. 38. The mission consisted in a start-to-goal query that required navigating from one side of the series of blocks (obstacles) to the other (see Fig. 46d). In this case, the vehicle was assumed to have a mechanically scanning profiler with a perception distance of 20m. The tree, that was generated by the modified RRT* algorithm, is presented in dark blue, while the path to the goal is drawn in red, and the path to the current waypoint appears in yellow.

The whole environment was initially unexplored and was incrementally mapped as the AUV navigated towards the goal. When the vehicle started the mission, and no obstacle had been detected, the waypoint sent to the AUV controllers (the tree root) coincided with the goal, since a straight path to it was feasible (see Fig. 46a). This situation persisted even when obstacles had been detected, as long as the path from the vehicle position to the goal was collision-free (see Fig. 46b). However, when such a straight path was not possible, the planner replanned (reshaped) the path from the vehicle’s current position (see Figs. 46c, 46d).

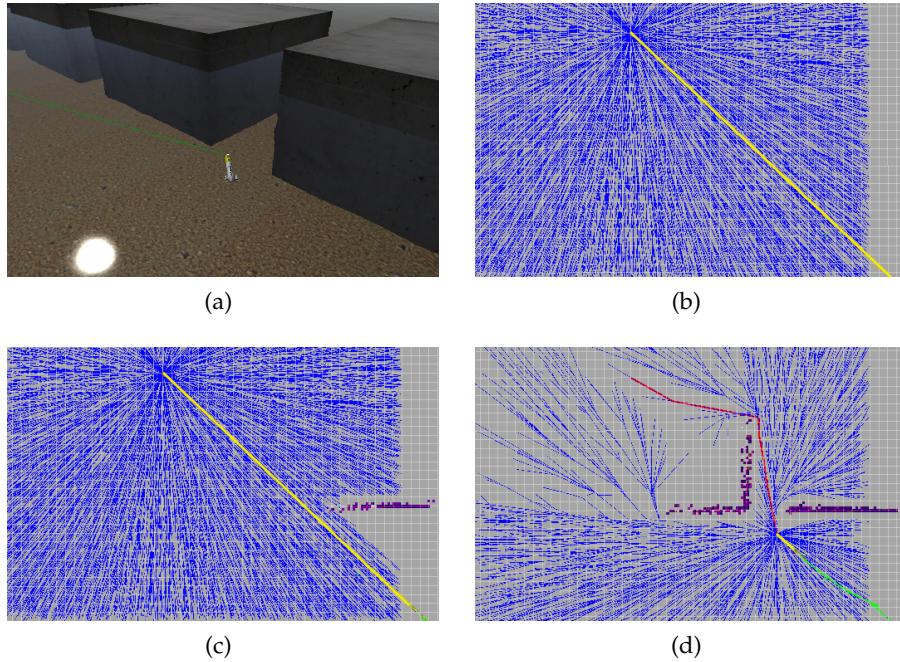


Figure 46: Sparus II AUV conducting a mission in the simulated breakwater structure (a), where it incrementally maps the environment (b), (c) and (re)plans a collision-free path to the goal (d). The tree of configurations is presented in dark blue, the path to the goal in red, the path to the current waypoint in yellow, and the vehicle’s trajectory in green.

5.3.3.2 Reuse of the Last Best Known Solution for Anytime Computation

The previously explained approach used a tree of configurations that is periodically traversed, checked and pruned as the vehicle moves and explores the environment. The main objective was to preserve the information about collision-free areas and known paths, while discarding those that become invalid. One alternative for not conserving the whole tree, is to use the *last best known solution* as the remainder of the path calculated in the previous planning cycle, which starts at the point that the vehicle will reach at the next execution cycle (see Fig. 47).

In this iterative and *anytime* computation scheme, using the last solution to start a new planning cycle implies not only a new valid solution according to an updated map, but also one that is at least as optimal as the previous one. This also permits to eliminate time-consuming pruning routines by avoiding checking subtrees, in which many of their config-

urations have probably become invalid because of the *opportunistic state checking* explained before.

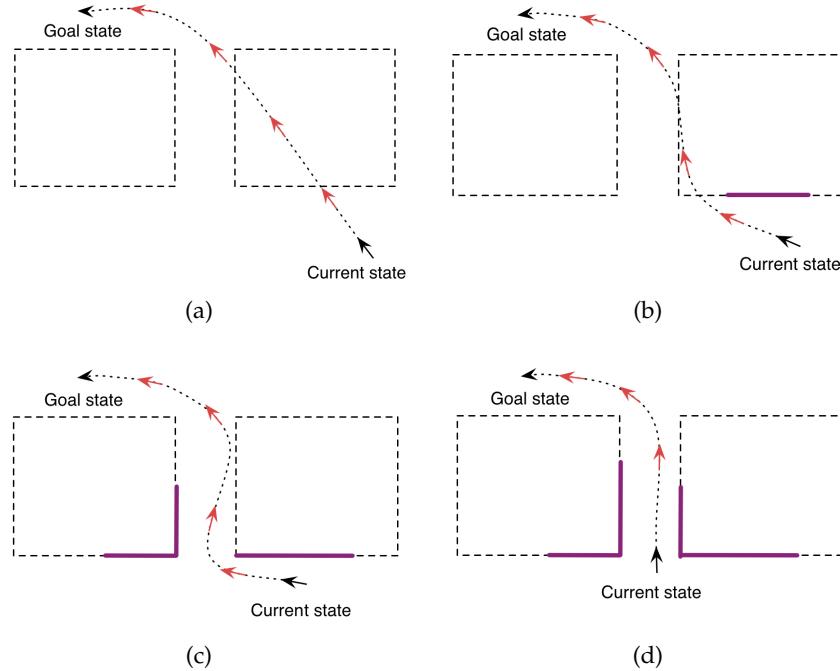


Figure 47: Reuse of the last best known solution. This strategy allows reusing the previous solution path for every new planning cycle, instead of checking and pruning the whole tree. This guarantees having valid solutions that are at least as good as the previous one.

5.4 PIPELINE FOR ONLINE PLANNING FEASIBLE AND SAFE PATHS

Previous sections have presented three new features: an optimization function that combines collision risk and path length; a strategy to avoid unnecessary and expensive state checking routines; and the reuse of the last best known solution as a starting point for an anytime planning approach. These features endow an AUV with the capability to simultaneously map and plan feasible and safe paths online through unexplored environments. In order to fully understand how such characteristics work together, Algorithm 10 presents the execution pipeline to incrementally solve a start-to-goal query. This pipeline has allowed conducting autonomous missions for the intended AUV applications. For this reason, the newly proposed features represent the main contributions to the *planning* module presented in this chapter.

In Algorithm 10, the main input parameters to solve a query are the start and goal configurations (position and orientation). To initialize the incremental solving routine, the RRT* algorithm is selected as the planner that computes paths for a Dubins (or extended 3D Dubins) vehicle, set an empty list as the *last best known solution*, and define $q_{\text{new_start}}$ as the starting configuration that will change as the vehicle conduct the mission (lines 2-4). To incrementally find a path to the goal (line 5), a main loop requests an updated version of the map (Octomap, lines 6-7), informs the

planner to start from *last best known solution* (line 8), uses the planner to find or improve the path (line 9), and gets the solution (line 10). At this point, the planner has provided a valid path that must be as optimal as the previous one, except that it has produced a longer but safer path. Before concluding a planning cycle, the incremental solving routine checks if a re-planning maneuver has been requested. This would imply that the *mission handler* has detected that the path from the current configuration to the last waypoint (WP) sent to the AUV controllers is not feasible, and might lead the vehicle to a collision, thus requiring that a new path should be found from the current configuration (see lines 11-13). Finally, if a new WP is required, the last q_{new_start} will be sent to the *mission handler* (lines 14-15).

It is important to note that even if the previous (*last best known*) solution is not in collision with an updated version of the map, the *planning* module will reuse it as a starting point (line 8), in order to attempt to improve such existing solution during a new planning cycle (lines 9 and 10).

Algorithm 10 : incSolveStart2Goal(q_{start}, q_{goal})

Input :

q_{start} : Start configuration.
 q_{goal} : Goal configuration.

```

1 begin
2   planner  $\leftarrow$  RRT*()
3   last_best_known_solution  $\leftarrow$  {}
4    $q_{new\_start} \leftarrow q_{start}$ 
5   while not stop_condition do
6     map  $\leftarrow$  reqUpdatedMap()
7     planner.updateMap(map)
8     planner.startFrom(last_best_known_solution)
9     planner.solve( $q_{new\_start}, q_{goal}$ )
10    last_best_known_solution  $\leftarrow$  planner.getSolution()
11    if replanning_requested then
12       $q_{new\_start} \leftarrow$  getCurrentConf()
13      last_best_known_solution  $\leftarrow$  {}
14    else if new_waypoint_requested then
15      sendWP(last_best_known_solution.pop())

```

5.5 RESULTS

In order to validate the proposed path/motion planning framework, this section presents simulation and real-world trials of the Sparus II AUV conducting autonomous missions in the breakwater structure scenario. This structure is located in the external and open area of a harbor, and it is composed of a series of concrete blocks of 14.5m long and 12m width, separated by a four-meter gap with an average depth of 7m (see Fig. 48). In this scenario, start and goal configurations were located in opposite sides, so that the Sparus II AUV had to move amidst the concrete blocks. All

queries were defined to conduct missions at a constant depth, thus the motion was restricted to a 2D task.

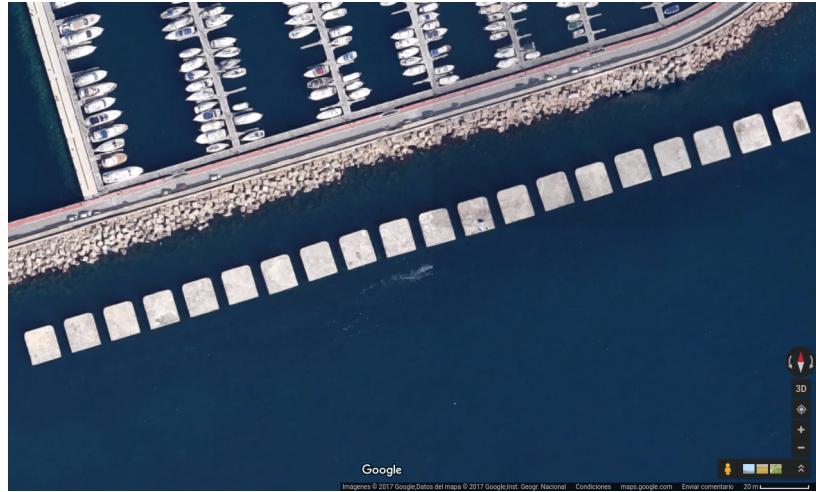


Figure 48: Breakwater structure composed of a series of concrete blocks (14.5m long and 12m width, separated by a four-meter gap) in Sant Feliu de Guíxols in (Spain). Image credit: Map data ©2017 Google.

For these tests, the vehicle was equipped with a mechanically-scanning profiling sonar to perceive and detect the surroundings. The sonar was located to cover a scan sector in the horizontal plane along the vehicle's direction of motion. From the software perspective, the Sparus II used the component oriented layer-based architecture for autonomy (COLA₂) [101], a control architecture fully integrated with the robot operating system (ROS). This software architecture also makes use of the open motion planning library (OMPL) that offers a convenient framework that can be adapted to specific planning problems [115]. Further details about the vehicle hardware and software characteristics can be found in Appendix A.

5.5.1 Comparison of Risk Functions

Before evaluating the framework's capabilities over unexplored environments, this section firstly assesses and establishes the best alternative to estimate the risk associated with a path. This analysis is done over a fully known and mapped scenario. Once this has been determined, tests over an unexplored environment prove the advantages of using the *opportunistic state check* strategy. Then, with this strategy, additional tests compares both the *pruning tree* and the *reuse of the last best known solution* approaches, thus allowing to demonstrate the latter one is the most efficient alternative for reaching an anytime computation approach.

5.5.1.1 Evaluation of the Risk Functions over Fully Mapped Environments

For these initial tests, an Octomap of the surroundings was assumed to be available *a priori*. Over this map, different start-to-goal queries were solved by using the alternatives to estimate the risk presented in Sec. 5.3.1. Figure 49 depicts some examples of the solutions paths obtained. It can be observed how the path maintains a safe distance from nearby obstacles (ar-

eas in purple), in contrast to what occurs when only path length criterion was considered (see Fig. 40).

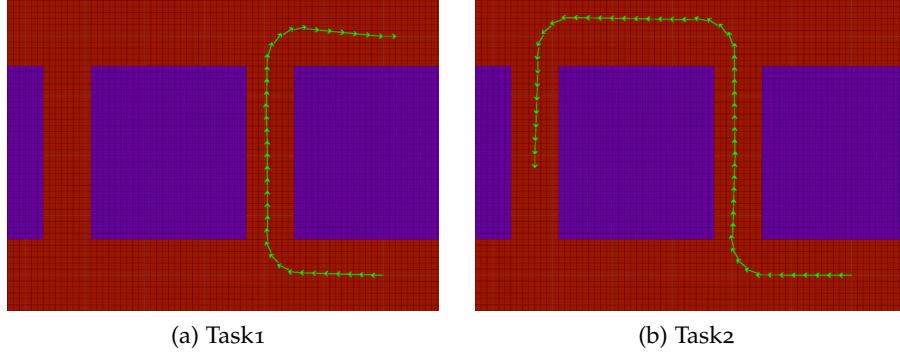


Figure 49: Start-to-goal queries' solutions using an [RRT*](#) algorithm with Dubins curves and risk functions. The solution paths maintain a safe distance from nearby obstacles (areas in purple).

Although all of the proposed approaches to estimate risk of the path can generate similar results, their associated computation times differ considerably. Figure 50 presents the average computation time required to solve Task1 and Task2 (Fig. 49) by each approach. It can be observed that using only the *path length* is clearly the least expensive method, while *path length + clearance* and *risk vectors* are the most and least expensive, respectively, when including the risk of the path. For this reason, the best alternative when dealing with explored (mapped) environments is the *risk vectors* approach.

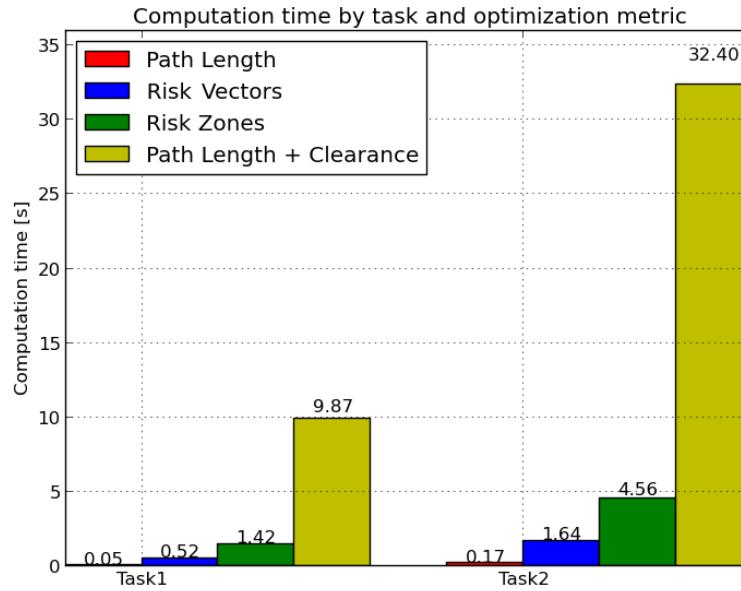


Figure 50: Average computation time, over 20 runs, required to solve Task1 and Task2 (Fig. 49) using different approaches to include risk of the path as the optimization criterion.

5.5.1.2 Evaluation of the Risk Functions over Unknown Environments

From Fig. 50, it can be concluded that the best computational alternative to include the risk associated with a path is the *risk vectors* approach. However, when dealing with unexplored environments, i.e., exploring while mapping incrementally, this approach may cope with situations in which partial information of the environment does not permit to estimate correctly the risk. In turning maneuvers, for instance, if an obstacle is located in the lateral motion direction, and it is not completely represented in the map so that the *risk vectors* does not coincide with the available partial information, this approach will indicate the configuration as safe, while the *risk zones* will estimate correctly the risk. For this reason, the best alternative when dealing with unexplored (unknown) environments is the *risk zones* approach.

5.5.2 Opportunistic State Checking and its Benefits

Although planning feasible and safe paths using *risk zones* is considerably faster than calculating clearance explicitly (see Fig. 50), having calculation times in the order of seconds may become into a limitation when coping with online computation limitations. Section 5.3.2 introduced the *opportunistic state checking* strategy as one of the mechanisms that can contribute to overcome this situation. To validate its efficiency, a start-to-goal query was solved and simulated multiple times, but now without assuming any previous map of the breakwater structure scenario (see Fig. 51).

The solutions for the query presented in Fig. 51 were obtained with and without using the *opportunistic state checking* strategy. Albeit in both cases the framework succeeded in conducting the task, Fig. 52a demonstrates that without it almost 80% of the total computation time is dedicated to risk checking routines over the whole mission. In the opposite scenario, i.e., using *opportunistic state checking*, its associated computation time increases as the environment is progressively explored, but even so, it does not consume such a percentage of computation time, not even at the end of the mission. This is especially noticeable when a mission does not require exploring and mapping completely the environment. Therefore, the use of the proposed mechanism affects not only the time required to find a solution, since it permits a better tree expansion (i.e., more states, see Fig. 52b), but also improves the workspace exploration and the path quality.

5.5.3 Anytime Computation for Solving Start-to-goal Queries in Unexplored Environments

Section 5.3.3 presented a *pruning* tree scheme and the *reuse of the last best known solution*. Both are alternative replanning approaches to make the framework capable of providing valid solutions at *anytime*. In order to prove the latter one is the best option, the execution pipeline presented in Sec. 5.4 has been used with both approaches to run different simulations and real-world in-water trials.

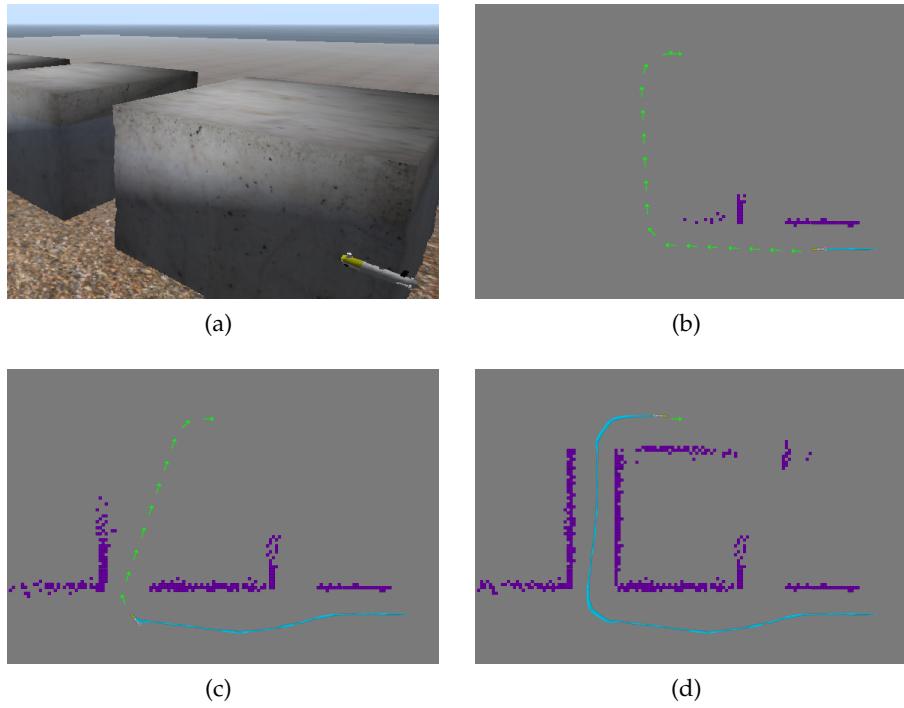


Figure 51: (a) Sparus II initial position for different start-to-goal queries in an equivalent virtual scenario of the breakwater structure. (b) Sparus II starts a mission by submerging to a specified depth. It then maps and solves a start-to-goal query simultaneously. (c) Equipped with a scanning profiler, it incrementally builds a representation of the environment, while continuously reshaping the solution path, (d) to finally approach to the specified goal configuration.

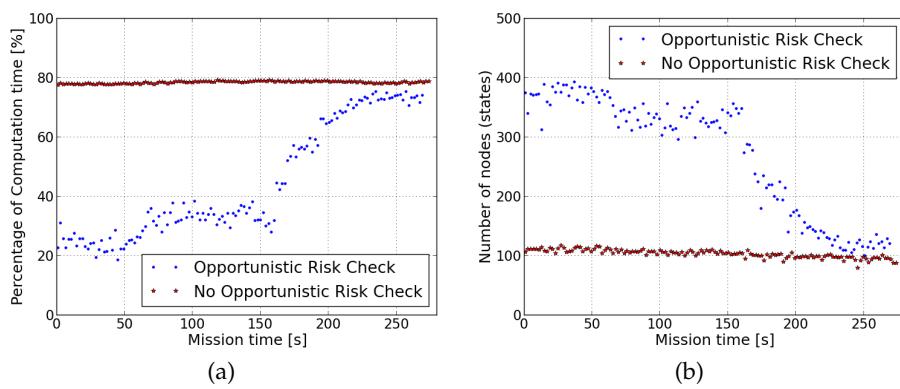


Figure 52: Incidence of *Opportunistic State Checking* approach when solving query presented in Fig. 51. (a) if configurations located in undiscovered areas are not assumed as safe, risk checking routines require almost 80% of computation time during the whole mission. Otherwise, it will increase progressively as the environment is explored. (b) consequently, if a high percentage of computing power is dedicated to risk checking, the number of tree nodes (states) will remain low during the whole mission, thus limiting the tree expansion and path quality.

5.5.3.1 Simulation Results

Two different simulated scenarios were used to compare the *anytime* computation approaches. The first of them is the breakwater structure mentioned in previous sections (see Figs. 38, 48). Over this scenario, two different start-to-goal queries were tested (see Fig. 53). After executing 10 times these missions, it was observed that the number of successful attempts is greater and the mean of replanning maneuvers (over the total successful missions) is smaller when *reusing the last best known solution with opportunistic state checking*, than those obtained with the *pruning tree* scheme (see Tables 2 and 3). Such replanning maneuvers correspond to situations where the calculated path was unfeasible, thus requiring a new valid path.

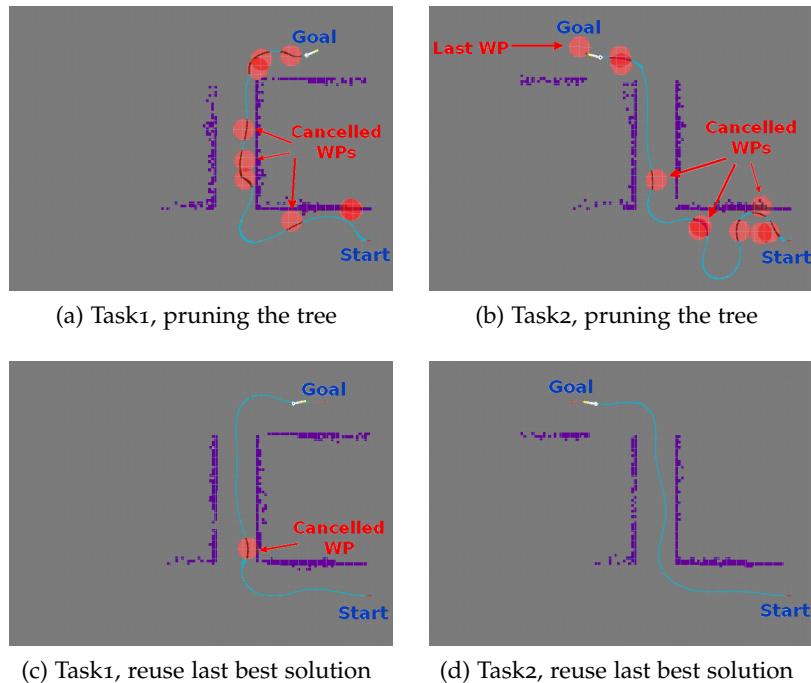


Figure 53: Comparison of the *anytime* computation approaches when solving start-to-goal queries with the *pruning-tree* scheme and the *reuse of the last best known solution* approaches. The number of replanning maneuvers are equivalent to the number of cancelled WPs (circles in red).

The second test scenario resembles a natural environment composed of rocky formations, which create an underwater canyon (see Figs. 54a, 54b). Over it, one start-to-goal query was defined in a way that the vehicle was required to travel through the canyon (see Figs. 54c, 54d). Once again, it was observed that the number of successful attempts is greater and the mean of replanning maneuvers is smaller in the case of *reusing the last best known solution with opportunistic state checking* (see Tables 2 and 3).

5.5.3.2 Real-world Results

After testing the planning framework in simulation, in-water trials were conducted in the real-world breakwater structure scenario (see Figs. 38, 48). The Sparus II AUV performed the autonomous missions with a constant surge speed $u = 0.5\text{m/s}$ and a maximum turning rate $r_{\max} = 0.3\text{rad/s}$.

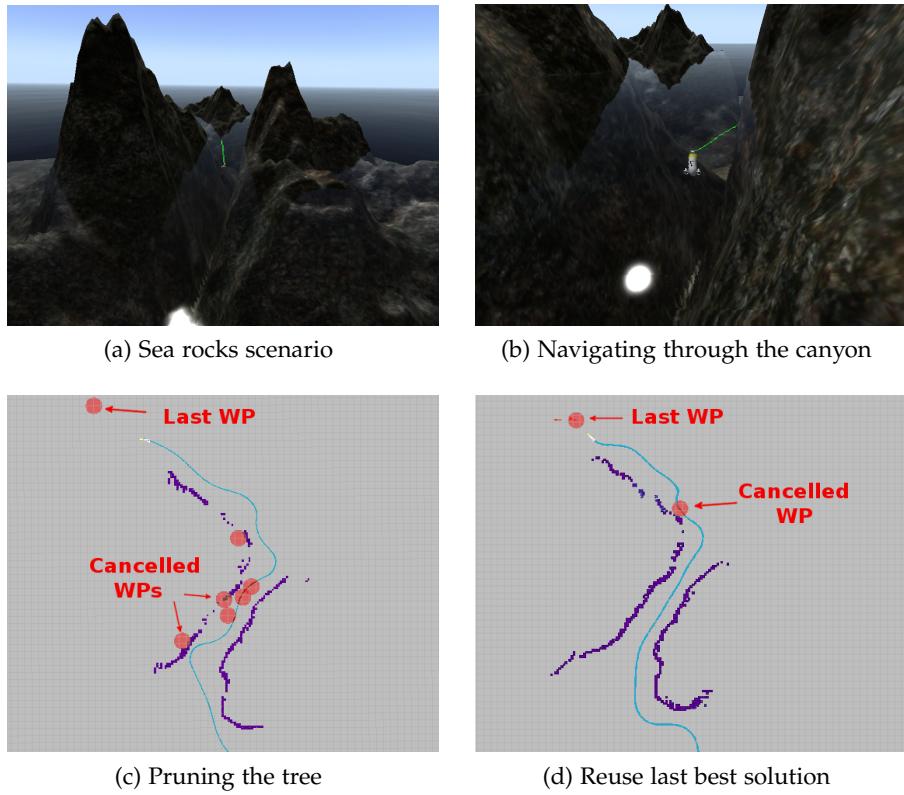


Figure 54: Comparison of the *anytime* computation approaches when solving start-to-goal queries with the *pruning-tree* scheme and the *reuse of the last best known solution* approaches. The simulated scenario resembles a natural-like environment. The number of replanning maneuvers are equivalent to the number of cancelled WPs (circles in red).

Virtual Scenario1: Breakwater Structure (Fig. 53)				
	Task1 (10 attempts)		Task 2 (10 attempts)	
	# Successful attempts	Mean replan. maneuvers	# Successful attempts	Mean replan. maneuvers
Pruning Tree	7	8.85	7	6.42
Reuse Last Sol.	10	0.3	9	0.1

Table 2: Comparison of solving the task shown in Fig. 53 using *pruning-tree* scheme and the *reuse of the last best known solution*. The second one proves to be the best alternative with: 1) the increase in the number of successful attempts, 2) the decrease of the mean of replanning maneuvers over the total successful missions. This latter implies that the vehicle had to deal with less risky situations, since a replanning maneuver supposes that the vehicle was being led to a possible collision.

Virtual Scenario2: Sea Rocks (Fig. 54)		
Task 3 (10 attempts)		
	# Successful attempts	Mean of replan. maneuvers
Pruning Tree	6	6.6
Reuse Last Sol.	7	1.14

Table 3: Comparison of solving the task shown in Fig.54 using *pruning-tree* scheme and the *reuse of the last best known solution*. As shown also in Table 2, reusing the last best solution proves to be the best alternative.

Figure 55 presents the AUV trajectory after conducting one of such missions, which consisted in solving a start-to-goal query that required the vehicle to navigate between two concrete blocks of the breakwater structure. Here again both approaches, i.e., the one using a tree *pruning* scheme and the one *reusing the last best known solution with opportunistic state checking*, were used to solve the task.

Figures 55a and 55b prove how replanning maneuvers decreased when *reusing the last best known solution*, just as it was expected from simulation results. Apart from visual results, it is also worth to mention that number of successful attempts were considerably higher when reusing previous solutions, 7 over 13, while experiments with a tree pruning strategy only succeeded 2 times of 5.

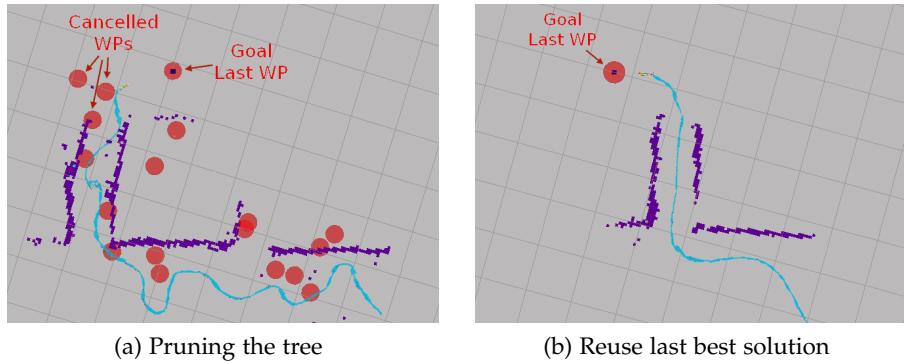


Figure 55: Real-world trial to compare the *anytime* computation approaches when solving start-to-goal queries without an initial map. The test scenario was the breakwater structure where the Sparus II had to navigate amongst two concrete blocks to move from one side to the other of the breakwater structure.

6

RESULTS IN REAL-WORLD SCENARIOS

Previous chapters presented different strategies and extensions that, all together, seek to endow an AUV with the capabilities required to succeed in the applications presented in Chapter 1. The final result is a framework that incrementally builds an Octomap of the surroundings and, simultaneously, plans a collision-free path to move through an initially undiscovered environment. To accomplish this, the framework not only considers the motion constraints involved (thoroughly explained in Chapters 3, 4), but also uses the *risk zones* to establish an optimization function for combining the length and the risk associated with the solution path (see Sec. 5.3.1). Furthermore, while the framework is expanding the tree to find a solution path, it *opportunistically checks* the sampled states and iteratively *reuses the last best known solution*. This allows to efficiently replan (reshape) the path in order to deal with the partial knowledge of the environment (see Secs. 5.3.2, 5.3.3).

This chapter presents an extensive evaluation of the proposed framework. This includes simulation and in-water trials that were carried out in different real-world scenarios. Such experiments were mainly conducted with the Sparus II AUV, but some of them also involved the AsterX AUV¹. The experiments are separated into four different scenarios: 1) planning constant-depth paths to move through artificial marine structures; 2) planning constant-depth paths to move through natural marine formations; 3) planning variable-depth paths to move through confined marine environments; and 4) the autonomous survey replanning for gap filling and target inspection. The following sections will discuss in detail the results obtained for each of these scenarios.

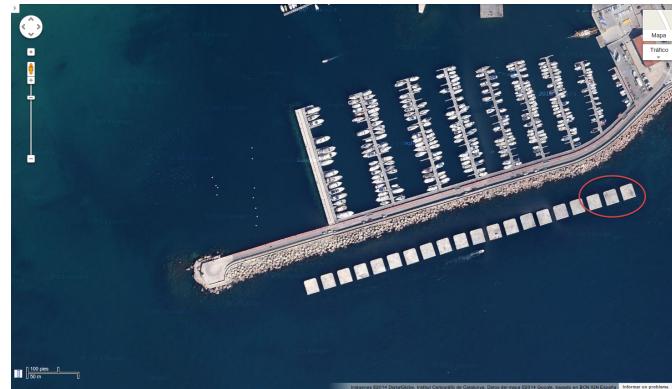
6.1 PLANNING AUV PATHS IN ARTIFICIAL MARINE STRUCTURES

Nowadays, there are a wide range of man-made marine constructions, going from large commercial harbors to offshore platforms. These artificial structures often have a regular and known shape, which could be used to preplan trajectories to travel through them. However, as it was explained in Chapter 1, there are different factors such as low visibility and limited navigation accuracy, which make it difficult to correctly follow such pre-calculated paths. The online path/motion planning framework proposed in this thesis seeks to contribute overcoming these limitations.

An example of this kind of structures is the breakwater mentioned in previous sections (see Fig. 56c). In this scenario, a test mission could be one that makes an AUV to move amidst the concrete blocks. In order to do so, the origin for the north-east-depth (NED) inertial system is defined by a latitude-longitude coordinate, which can be obtained from Google Maps ©. Likewise, a start-to-goal query can be specified, in such a way

¹ Hardware and software details for both vehicles can be found in Appendix A.

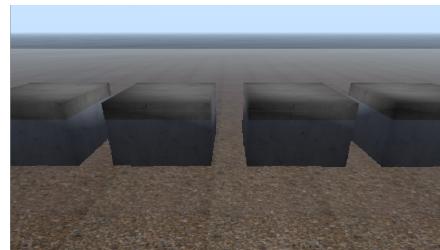
that the **AUV** is required to traverse the blocks from the outer to the inner area, and vice versa. Once the origin, the start configuration and the goal configuration have been selected, it is necessary to define a workspace representation or map, where the collision checking routines can be done. This latter is needed in the case of solving the query with a sampling-based approach, otherwise a discrete or full description of the **C-Space** may be required instead.



(a) Satellite view. Image credit: Map data ©2017 Google.



(b)



(c)

Figure 56: Test scenario of an artificial marine structure. (a) Harbor of Sant Feliu de Guíxols in Catalonia (Spain), where a breakwater structure composed of concrete blocks is demarcated. (b) Close proximity view of the concrete blocks. (c) Virtual environment for simulations over **UWSim**.

As mentioned in previous chapters, an Octomap of the area can be used for collision-checking purposes. Assuming the environment as explored, the Octomap could be either one built from real data (see Fig. 38b) or one synthetically created (see Fig. 33). Then, a collision-free, feasible and safe path could be calculated over such a known map. Simulations of this kind, i.e., using an existing map, were presented in Chapters 3 and 4. However, before conducting any real-world autonomous mission, the Sparus II **AUV** was teleoperated at surface while traversing the concrete blocks. This allowed to verify the map consistency with respect to the GPS data (see Fig. 57).

Figure 57 highlights one of the main limitations of assuming the environment as previously known and mapped. Even if the **AUV** is teleoperated at surface, which allows to continuously obtain GPS fixes, there is an error associated with the GPS that can range from one to two meters. This magnitude of error is especially critical when navigating in narrow passages, such as the four-meter gaps between the concrete blocks. Furthermore, if



Figure 57: Sparus II AUV was teleoperated at surface in the breakwater structure scenario. Although the Sparus II did not collide during the tele-operated mission, GPS data (in red) shows a vehicle trajectory under collision with and even over the concrete blocks. Satellite view. Image credit: Map data ©2017 Google.

the vehicle submerges, the position estimation error will also accumulate the navigation drift associated with the dead-reckoning ([DR](#)) system.

An alternative to conduct the intended mission could be the use of an [USBL](#) system, which would contribute to minimize the navigation error. Nonetheless, this option requires a surface vessel that follows the [AUV](#) along the mission, something that is not feasible in this scenario. Considering all this, an alternative approach is to use the framework proposed in this thesis.

As explained before, the framework requires that the vehicle is equipped with exteroceptive sensors. These sensors allow detecting the surroundings to create a map online. For this scenario, a set of four echosounders and one mechanical-scanning imaging sonar were located within the vehicle payload (front) area, all of them pointing in the horizontal plane (see Fig. 58). Three of the echosounders were separated by 45° , with the central one looking forward and parallel to the vehicle's direction of motion, while the fourth one was perpendicular to the central one (see Fig. 58a). As for the imagining sonar, it was setup to cover a scan sector in the vehicle's direction of motion (see Fig. 58b). Since both sensors cover the horizontal plane, missions with this configuration were executed at a constant depth. For more details about the complete Sparus II's hardware configuration, the interested reader is referred to Appendix A.

During the first simulation and in-water trials conducted over this test scenario, the Sparus II [AUV](#) only used the echosounders to perceive the environment. Nevertheless, the four echosounders beams were insufficient to rapidly establish the validity of the path along the direction of motion, which triggered multiple replanning maneuvers. Although the vehicle was capable of finding a solution path, it did required multiple attempts to complete one full successful mission. Furthermore, this limitation generated unexpected and non-desired trajectories. An example of this was already presented in Chapter 5, Fig. 55a.

In order to avoid the aforementioned situations, the Sparus II used both the echosounders and the mechanical-scanning imaging sonar to incre-

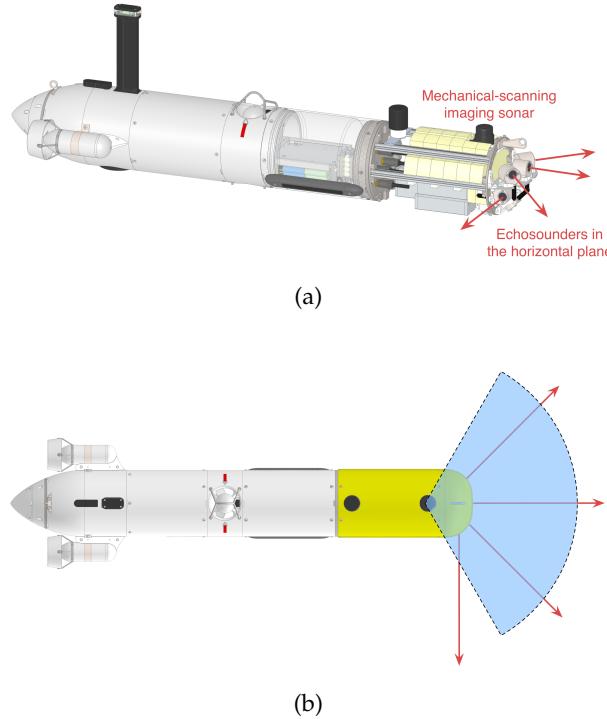


Figure 58: Exteroceptive sensors configuration for the Sparus II AUV to conduct autonomous missions in the breakwater structure. (a) The vehicle was equipped with four echosounders and one mechanical-scanning imaging sonar, all of them pointing in the horizontal plane. (b) Top view, echosounders beams direction and imagining sonar scan sector.

mentally build the map. For each beam position within the scan sector, the imaging sonar provided an array of intensities. From these intensity values, those that were over a specified threshold represented the obstacles detected. Once those values had been identified, they were transformed into ranges or distances to the obstacles. However, since the imaging sonar has a vertical beamwidth of 35° , the maximum range was limited to 10m to avoid false-positive detections from the sea bottom. The echosounders, on the other hand, were setup with a maximum range of 20m, since they have a smaller beamwidth of 10° (see Appendix A).

This payload configuration was used in different sea trials, and allowed the vehicle to traverse the breakwater structure with great repeatability. To prove this latter, the Sparus II conducted missions that included multiple and successive start-to-goal queries. Figure 59 depicts one of these missions. For safety reasons, the vehicle was connected to surface with a wireless access point buoy that allowed us to monitor the mission and abort it in case of detecting an unexpected behavior (see Fig. 59a). For this mission, the Sparus II navigated with a constant surge speed $u = 0.5\text{m/s}$, a maximum turning rate $r_{\max} = 0.3\text{rad/s}$, and at a constant depth of 2m.

The mission was composed of three different and consecutive start-to-goal queries, and assumed the environment as unexplored. Both the NED reference frame origin and the different goals coordinates when obtained from Google Maps © 2017. For the first query, the vehicle was initially in the inner area of the breakwater structure, and the goal was a coordinate in the outer area. Hence, the only possible solution path required the AUV

to navigate through one of the four-meter gaps (see Fig. 59b). The second query was set back in the inner area but in a different coordinate, thus requiring to move through to one of the gaps once again (see Fig. 59c). Likewise, the third and last query was defined to conclude the mission in the outer area (see Figs. 59d, 59e).

Along the mission, the Sparus II did not surface to obtain GPS fixes. Figure 59f depicts the vehicle trajectory and the Octomap built overlapped with a satellite imagine of the breakwater structure. Although the Octomap is coherent with respect to the real-world structure, there are some differences due to the accumulation of navigation error. Yet this did not prevent the AUV from completing successfully the mission.

To further discuss potential applications, Figure 60 depicts a photo-realistic 3D model of the area traveled by the Sparus II during the mission presented above. In this particular mission, the AUV used a stereo optical camera, which gathered additional data from the surroundings. Such information was used to create a more detailed survey of the area after concluding the mission. Other missions may include sensors to measure different variables of interest, such as biological, chemical, as well as bathymetric information. For more details about these 3D reconstructions, the interested reader is referred to [46, 47].

6.2 PLANNING AUV PATHS IN NATURAL MARINE FORMATIONS

Although the mission in the breakwater structure represents a valid application example, it is also true that the obstacles in it, i.e., the concrete blocks, have regular shapes. Their vertical walls, for instance, were correctly detected by a mechanical-scanning imaging sonar, despite its great beamwidth. Natural environments, on the other hand, present less regular and more challenging scenarios. Therefore and in order to prove the proposed framework under different conditions, the Sparus II also conducted autonomous missions over a real-world natural environment. The testing area is also located in Sant Feliu de Guíxols (Spain), and contains rocky formations that create an underwater canyon (see Fig. 61).

For this scenario, the Sparus II AUV used a mechanic-scanning profiling sonar, which has a smaller beamwidth of $1 - 2^\circ$ (see Appendix A). This sensor, which covered the horizontal plane, permitted not only to perceive the obstacles shape with more accuracy, but also to set a higher maximum range without being affected by false-positive detections from the bottom. The AUV was also equipped with a set of three GoPro™ Hero 4 Black edition cameras. They were used to gather the optical images required to create a 3D reconstruction of the surroundings. The cameras were positioned systematically to ensure the highest possible coverage, where two of them were placed in a downward configuration at an angle of 20° , while the third camera was positioned forward looking at 40° (see Fig. 62).

In order to inspect the underwater canyon and the surroundings, two different start-to-goal queries were established by extracting GPS coordinates from Google Maps ©. The first query required the Sparus II AUV to traverse the canyon towards the shore. The second query goal was chosen on the outside of the rocky formation in such a way that the vehicle had to

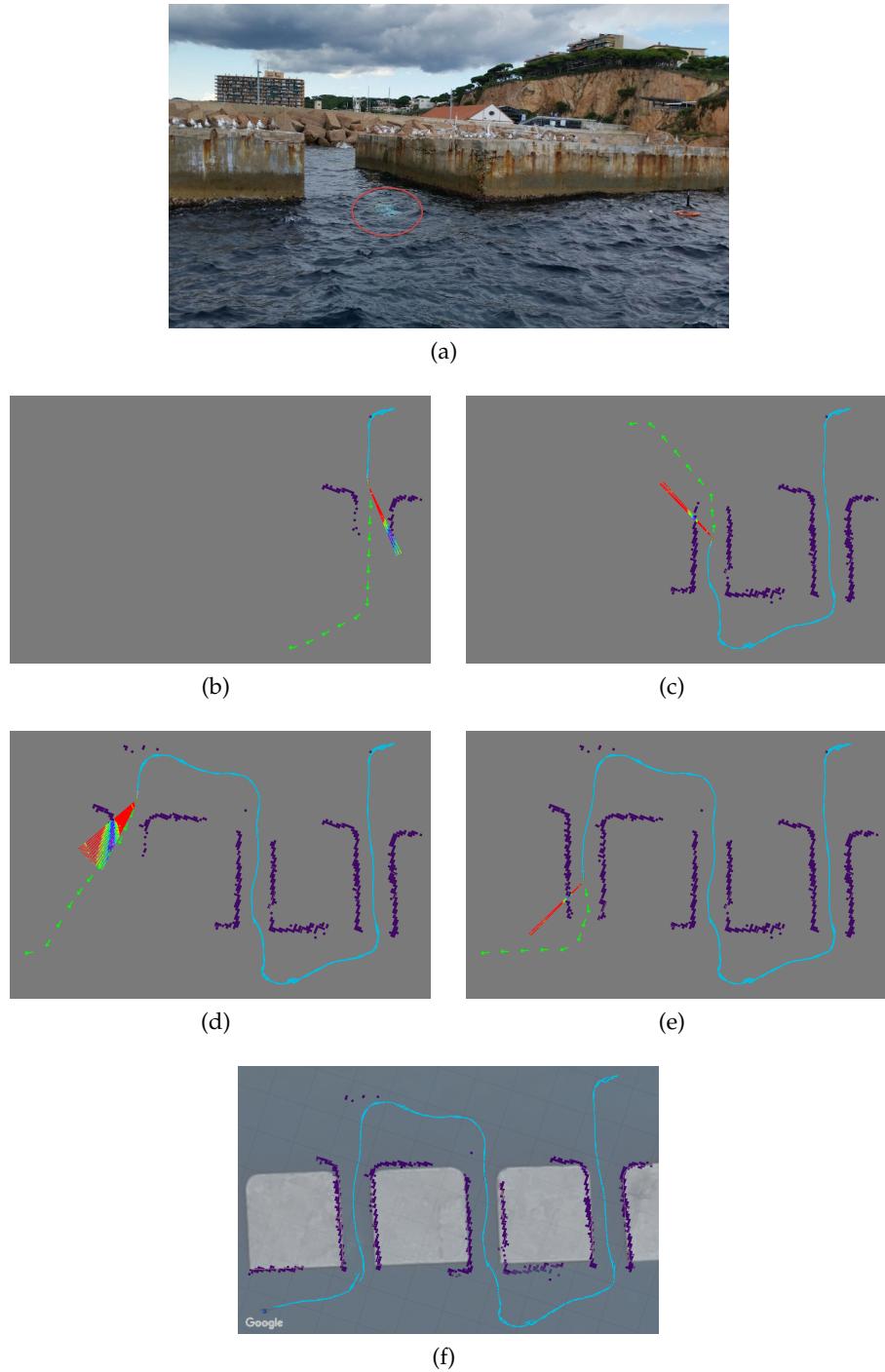


Figure 59: The Sparus II AUV used the path/motion planning framework to navigate through a breakwater structure without a preliminary map of the surroundings. (a) The vehicle is submerged and is moving autonomously amidst two concrete blocks. (b), (c), (d), (e) The mission required traversing the breakwater structure multiple times by solving successive start-to-goal queries. (f) The vehicle trajectory is drawn in light blue, while the map built with the imaging sonar data is presented in purple. This information is overlapped with a satellite image (Google Maps © 2017).

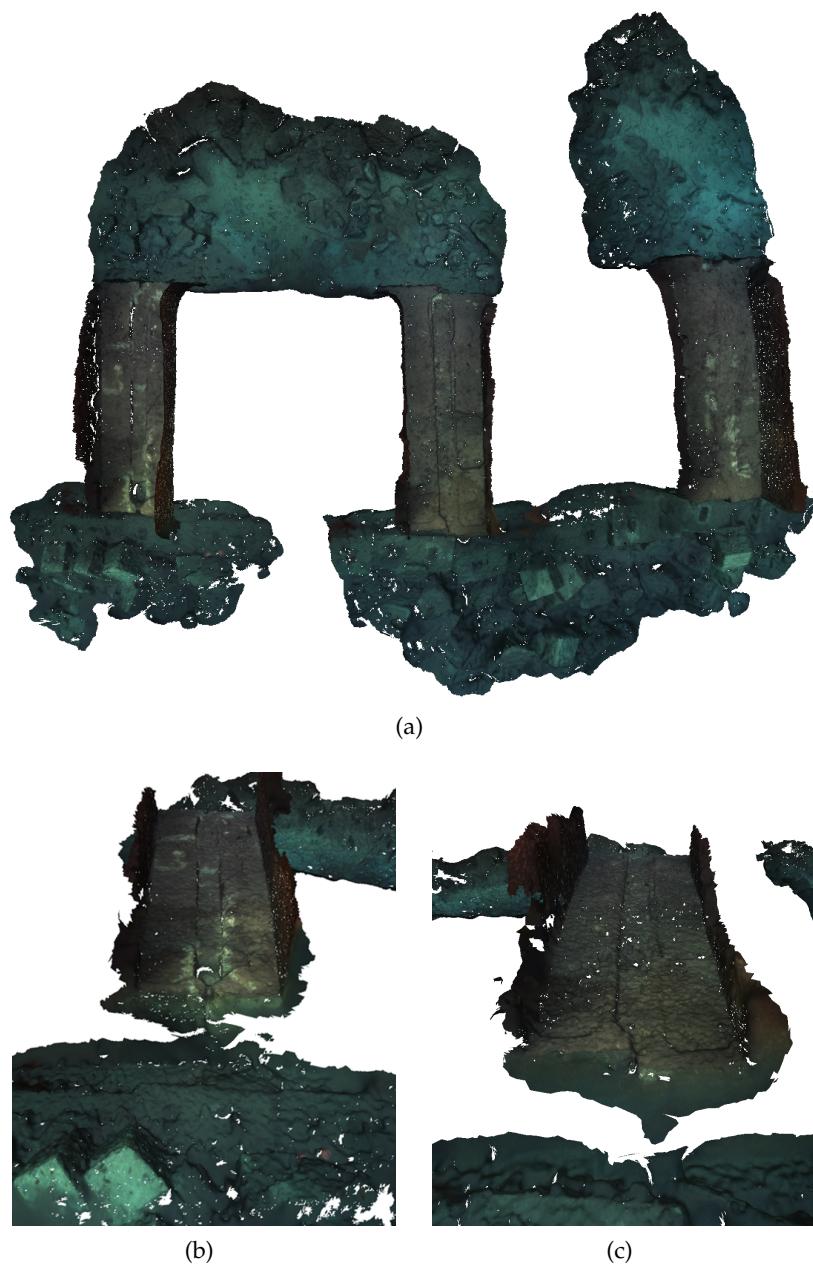


Figure 6o: (a) Top-down view of the textured 3D model built with the images gathered along the mission shown in Fig. 59. (b),(c) 3D views of the corridors or gaps between the concrete blocks.

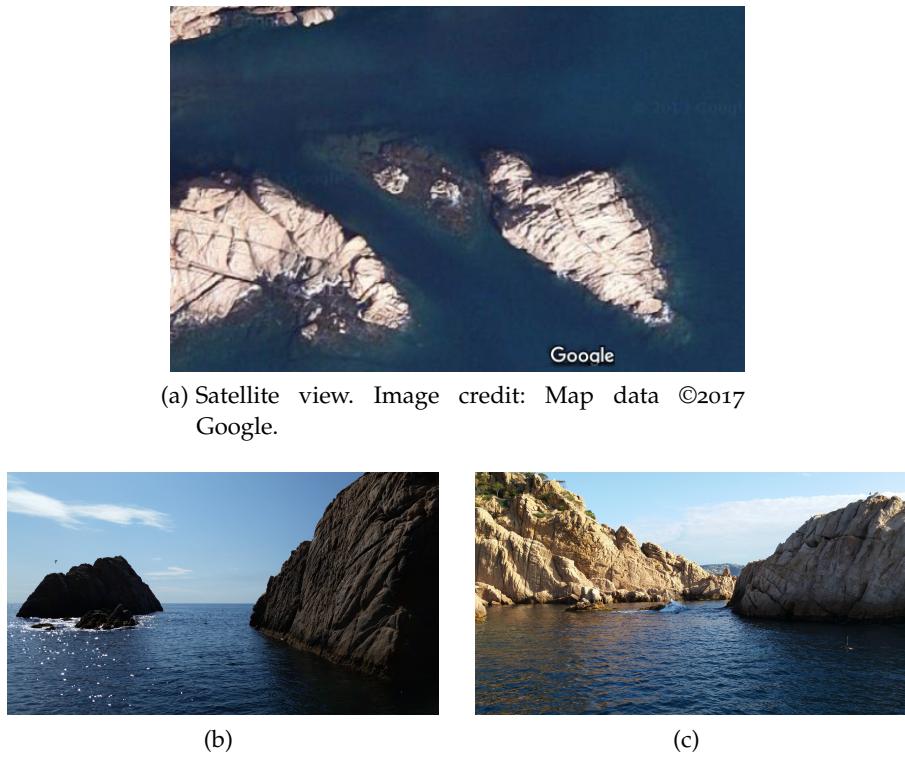


Figure 61: The test scenario consists of rocky formations. (a) They create an underwater canyon that can be observed in the satellite image. (b) From the inner area (i.e., from the coastline towards the open sea) the canyon can be clearly observed. (c) From the outer area the canyon can be observed in the left.

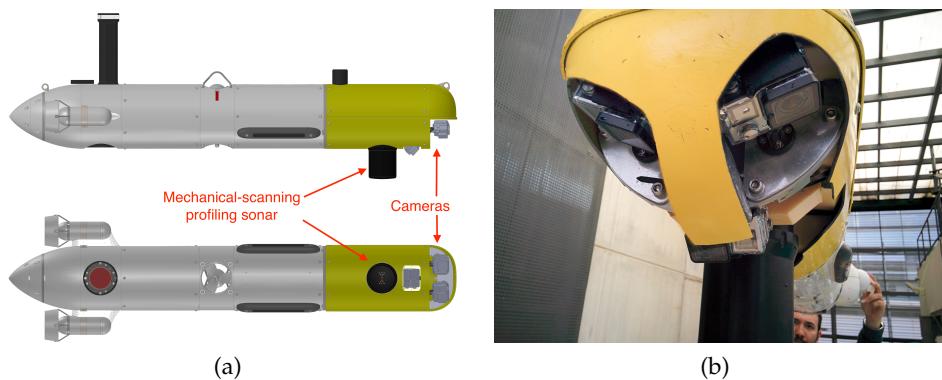


Figure 62: Exteroceptive sensors configuration for the Sparus II AUV to conduct autonomous missions in the underwater canyon. (a) The vehicle was equipped with three GoPro™ Hero 4 Black edition cameras and one mechanical-scanning profiling sonar. (b) Real-world vehicle's payload.

circumnavigate the outer rock. Furthermore, after completing the second query, the first query was executed again until the vehicle overlapped its initial trajectory in the canyon. This allowed us to verify the navigation consistency and to close the imaging acquisition loop, thus improving the reconstruction results. As the profiling sonar covered the horizontal plane, the mission had to be conducted at a constant depth.

Figure 63 depicts one of the inspection missions conducted at 3m of depth in the underwater canyon. The Sparus II not only created a map of a complex and unexplored environment, but also planned a collision-free path simultaneously and incrementally. The map and the vehicle trajectory are shown overlapped with a satellite image. In the initial part of the mission, i.e., when the vehicle traverses the canyon for the first time, the map coincides with the satellite image (see Fig. 63b); however, disparities can be clearly observed after some time (see Figs. 63c, d). Such differences are due to the accumulation of error in the navigation system that depends on the doppler velocity log (DVL), which may provide incorrect data when navigating over rocks, as occurred in this test scenario. Despite this situation, the vehicle succeeded in conducting the mission because both the map and the path are created online, which permits correcting or adjusting them even when moving in previously visited areas (see Fig. 63d when accessing the canyon for a second time).

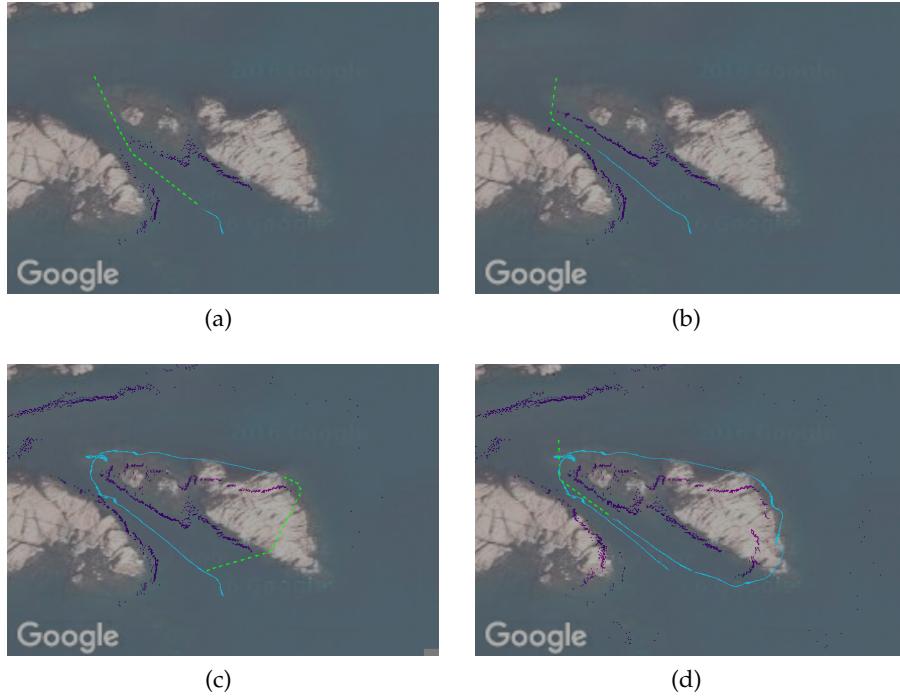


Figure 63: The Sparus II AUV used the path/motion planning framework to navigate through an underwater canyon without a preliminary map of the surroundings. (a), (b) The first start-to-goal query required the Sparus II traversing an underwater canyon. (c) For the second query, the vehicle circumnavigated the outer rock. (d) The AUV partially repeated the first start-to-goal query in order to close the loop and to obtain overlapped images. Vehicle trajectory and Octomap overlap a satellite image (Map data ©2017 Google).

In this kind of explorations, the data that is gathered along the mission can be used to create a more detailed survey of the area. In this particular mission, for instance, the images were extracted and were used to build a photo-realistic 3D model, which is depicted in Figure 64a. This 3D reconstruction allows us to better understand complex environments by generating arbitrary user-defined views (see Figs 64b–d). For more details about these 3D reconstructions, the interested reader is referred to [46, 47].

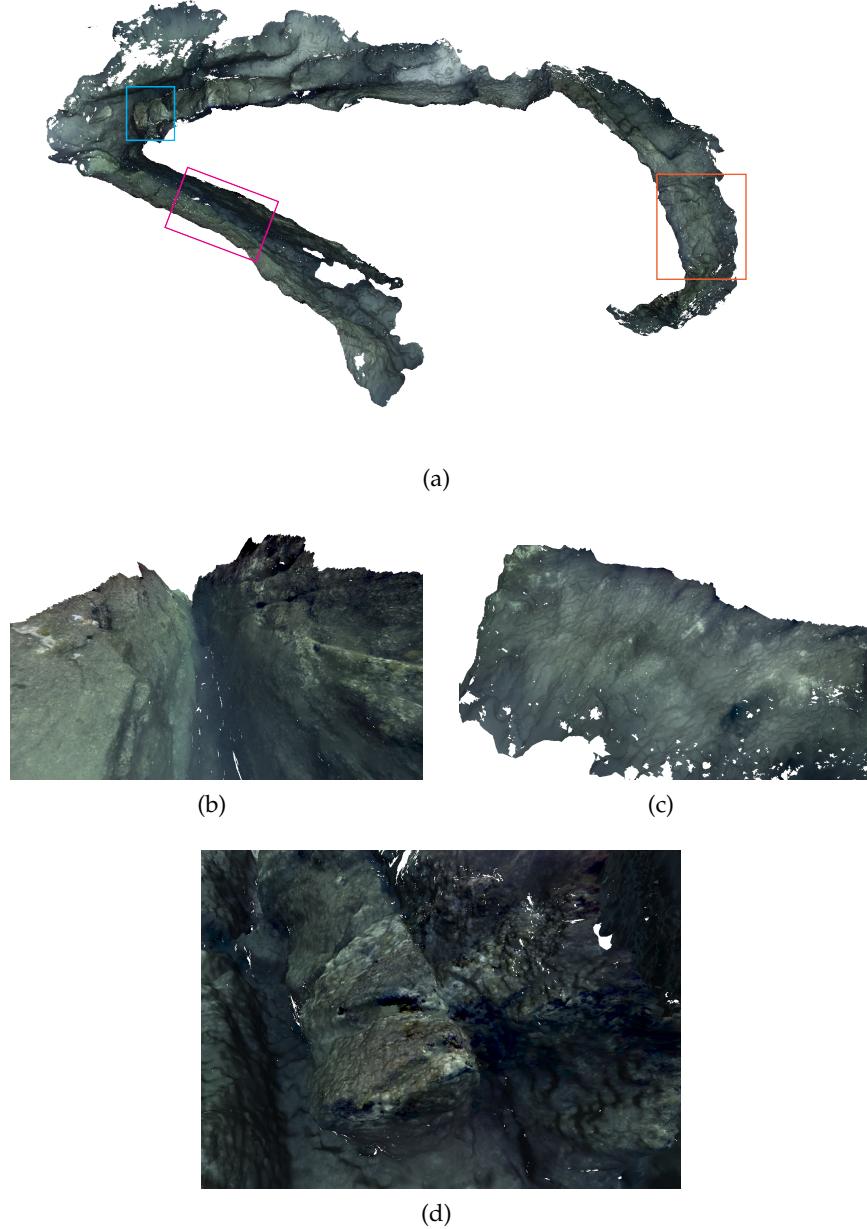


Figure 64: (a) Top-down view of the textured 3D model with marked areas additionally depicted in magenta, orange and blue, which correspond to generated views of: (b) the underwater canyon; (c) the external side of the underwater rocky formation; (d) underwater rocks.

Finally, in order to understand the complexity associated with this mission, Figure 65 presents a visual comparison between the vehicle's trajectory, estimated by its DR system, and the cameras' trajectory, estimated during the reconstruction (green and red, respectively). While both trajec-

tories have a similar shape, it can be clearly observed how the one derived from the cameras is more realistic according to the rock observed in the surface (the rocky formation does not create a vertical wall, which means that the vehicle may have moved further from the visible part of the rock when navigating at 3m deep), while the one estimated by the AUV's dead-reckoning system seems to be colliding with the rock. This latter situation is mainly due to the accumulation of errors in the navigation system, as it was already mentioned before.



Figure 65: Vehicle trajectory (green) calculated by its dead-reckoning system and the cameras trajectory (red) estimated by the image-based reconstruction. Both trajectories are shown overlapped with a satellite image of the test scenario (Map data ©2017 Google).

6.3 PLANNING AUV PATHS IN CONFINED NATURAL ENVIRONMENTS

The experiments that have been presented until now involved the Sparus II navigating at a constant depth. However, there are other scenarios in which the AUV must modify its vertical position in order to complete the intended mission. In such cases, one of the main limitations from the mapping and path-planning perspective is the capability to build online a 3D map of the surroundings. As explained in Sec. 5.2, Octomaps offer an efficient alternative to represent volumetric environment information. Such data can be gathered from a wide range of sensors such as echosounders, profiling/imaging sonars, and multibeam sonars.

One example of a mission that require a 3D path/motion planner is to move through confined natural environments (e.g., underwater caves and tunnels). This kind of scenarios present all sort of limitations, including the impossibility of having either a support surface ship to correct the AUV navigation through an USBL system, or even a wireless access point buoy for monitoring the correct mission execution. Other technical aspects that make more difficult to conduct autonomous missions in these environments, include the navigation error associated with incorrect data provided by the DVL when traveling over rocky surfaces.

In dealing with this latter aspect, Mallios et al. presented the exploration of an underwater caves complex (see Fig. 66), in which a diver guided the Sparus AUV to gather environment acoustic information [91]. To do so, the vehicle was equipped with two mechanically scanning imaging

sonars to cover the horizontal and vertical planes. Such data was used to prove a scan-matching algorithm over a simultaneous localization and mapping ([SLAM](#)) framework, which allows reducing and bounding the [AUV](#) navigation error. This survey provides an extensive dataset that includes not only the sonars' raw data, but also a detailed meshed map (see Fig. 67) [91, 92]. This latter one was obtained by using the splats distance normalized cut ([SDNC](#)) algorithm [15]. For more details about this survey, the interested reader is referred to the cited references.



Figure 66: Satellite image of the dataset area. The underwater caves complex “Coves de Cala Viuda” is located in the L'Escala, Spain (Lat: 42.10388, Lon: 3.18255). It consists of three single-branch caves and several tunnels. Their approximate positions are marked with dotted white lines. Image credit: Mallios et al. 2015 [91, 92]

All this together is nowadays considered a significant step towards the exploration of these kinds of environments. Nonetheless, there are still other aspects that must be tackled in order to conduct this kind of missions fully autonomous. This thesis contributes to coping with some of those aspects. The proposed framework, for instance, seeks to endow [AUVs](#) with additional capabilities that will allow to replace the diver's guidance in a near future. To prove these capabilities, this section presents a simulated mission over the meshed map of the cave complex. The mission consisted in solving six consecutive start-to-goal queries, where the Sparus II was not provided with any preliminary environment information, thus requiring to incrementally map and (re)plan the path to the goals. Figure 67 depicts the meshed map and the approximate locations of six different query goals.

In order to conduct this test mission, the meshed map was added into [UWSim](#) as a [3D](#) simulation environment (see Fig. 68). To perceive the surroundings, the [AUV](#)'s payload was assumed to be equipped with an additional link capable of rotating 120° around the vehicle's z axis. Over this link, a forward-looking multibeam sonar was mounted. The sonar had 240 beams distributed over a total aperture of 120° around the vehicle's x axis. With this payload configuration, the simulated Sparus II was capable of gathering [3D](#) range data of the environment along its direction of motion.

At the beginning of the mission, the simulated Sparus II was located at the origin of the [NED](#) reference system and oriented towards the north,

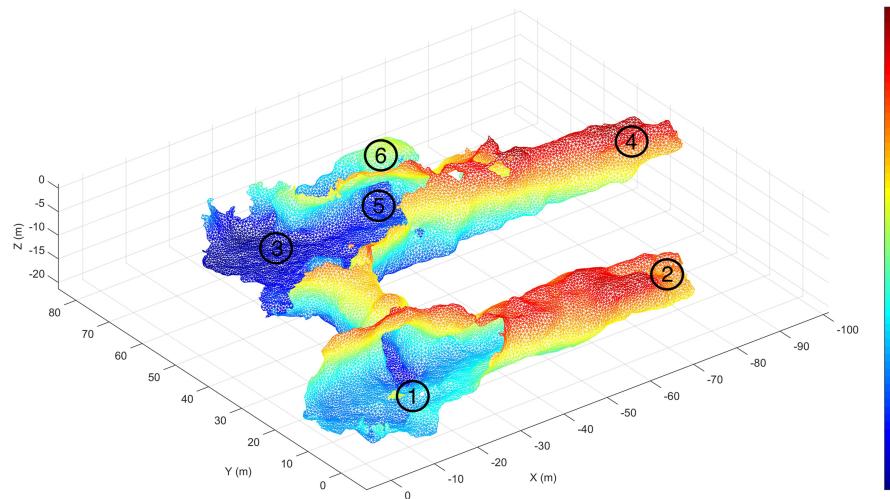


Figure 67: The underwater caves complex “Coves de Cala Viuda”. Meshed map using the SNDc algorithm [15]. Image credit: Mallios et al. 2015 [91]. Over the map, the approximate locations of six different goals have been marked in black.

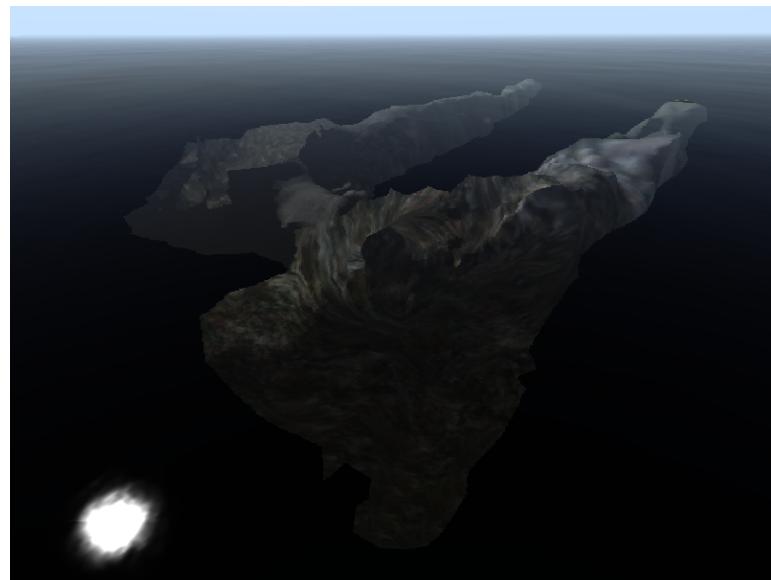
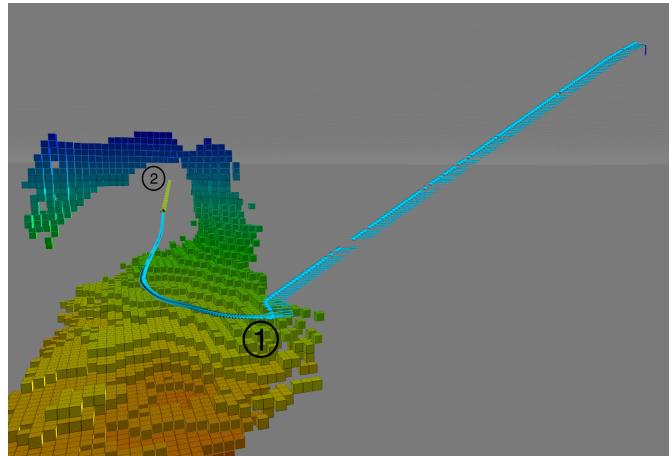


Figure 68: The underwater caves complex “Coves de Cala Viuda” added into UWSim as a 3D simulation environment.

i.e., $q_{\text{start}} = [0.0, 0.0, 0.0, 0.0]$. The first start-to-goal query was defined to guide the AUV closer to the caves complex location. This required setting $q_{\text{goal}_1} = [20.0, 0.0, 12.0, 0.0]$. From this latter position, the second query sought to explore the first and biggest single-branch cave. This meant navigating through and until the end of the cave, which required setting $q_{\text{goal}_2} = [18.0, 55.0, 8.0, 1.57]$. For these queries, the planner limited the vehicle to navigate at a constant surge speed $v = 0.5\text{m/s}$, a maximum ascending speed $d_{\text{ascend}} = 0.2\text{m/s}$, and a maximum descending speed $d_{\text{descend}} = 0.18\text{m/s}$. Figure 69 depicts part of the execution of these two queries, including the AUV trajectory, the calculated path, and the goals.



(a) Vehicle trajectory, calculated path, goals, and Octomap

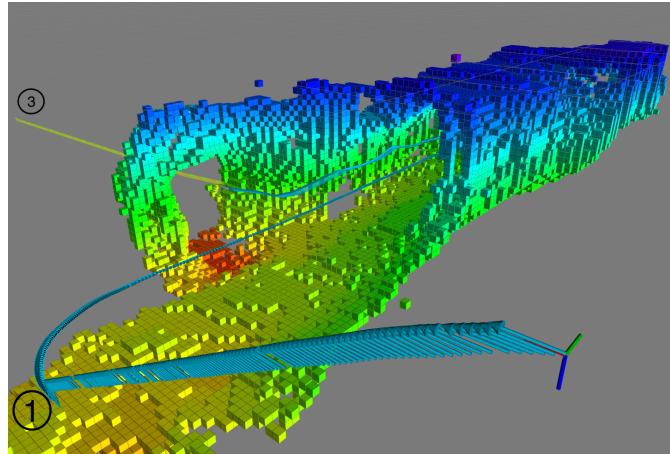


(b) Sparus II approaching the entrance of the first cave

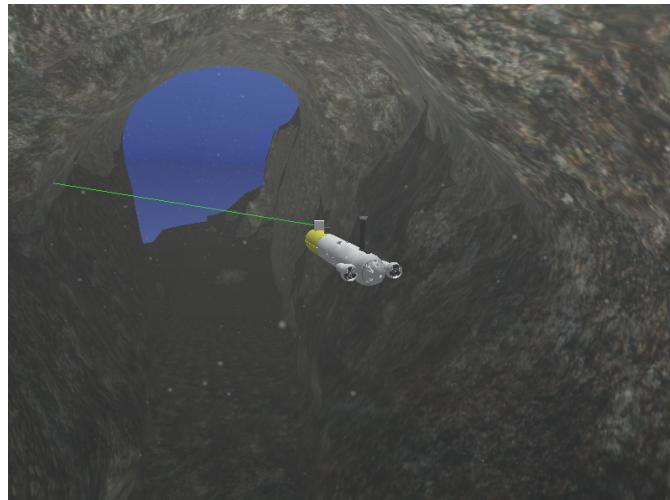
Figure 69: Simulated mission in the caves complex. The first start-to-goal query guided the AUV closer to the complex location. The second query required the vehicle to navigate through the first single-branch cave. In (a) The vehicle trajectory appears in light blue, and the remaining of the calculated path in yellow.

From the second query's goal configuration, a third query was defined to take the Sparus II closer to the second single-branch cave's entrance, i.e., $q_{\text{goal}_3} = [80.0, 25.0, 16.0, 0.23]$. To accomplish this part of the mission, the AUV not only had to find a way out of the first cave, but also had to traverse a tunnel that connects the entrances of both single-branch caves. For

this query, the planner limited the vehicle to navigate at a constant surge speed $v = 0.3\text{m/s}$, while keeping the previous maximum ascending/descending speeds ($d_{\text{ascend}} = 0.2\text{m/s}$ and $d_{\text{descend}} = 0.18\text{m/s}$). This decrease of v permitted the vehicle to conduct maneuvers with a smaller turning radius, especially required to leave the first cave. Figure 70 depicts part of the execution of this query.



(a) Vehicle trajectory, calculated path, goals, and Octomap

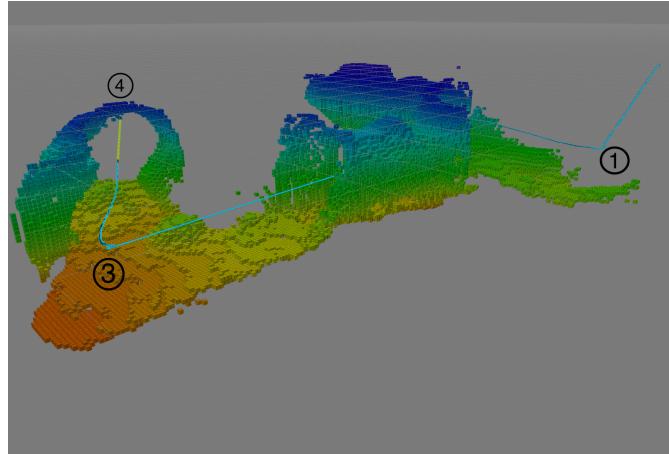


(b) Sparus II traversing the underwater tunnel

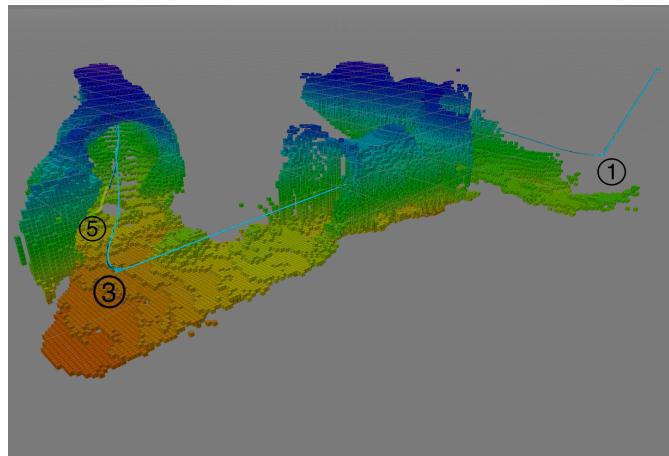
Figure 70: Simulated mission in the caves complex. The third start-to-goal query guided the AUV to the second single-branch cave's entrance. In (a) the vehicle trajectory appears in light blue, and the remaining of the calculated path in yellow. This query required the vehicle not only to find a way out of the first cave, but also to traverse a tunnel that connects the entrances of both single-branch caves.

Once the Sparus II AUV had crossed the tunnel, the fourth query was defined to navigate through and until the end of the second single-branch cave, i.e., $q_{\text{goal}_4} = [60.0, 87.0, 9.0, 1.82]$. To do so, the speed constraints were kept as established for the previous query. Once this was accomplished, the fifth query was set to guide the AUV close to a third cave's entrance; this meant $q_{\text{goal}_5} = [82.0, 47.0, 17.0, 0.0]$. This latter query required the planner to find a way out of the second cave, which is considerably narrower than the first one. To cope with this latter situation, the planner used a lower

surge speed, $v = 0.1\text{m/s}$, which allowed the vehicle to turn back with a smaller turning radius. Figure 71 depicts part of the execution of these two queries.



(a) Sparus II traversing the second cave



(b) Sparus II on its way back to the third cave's entrance

Figure 71: Simulated mission in the caves complex. (a) The fourth start-to-goal query required the vehicle to navigate through the second single-branch cave. (b) The fifth query was defined to guide the vehicle close to a third cave's entrance. This required the planner to find a way out of the cave. In both images, the vehicle trajectory appears in light blue, and the remaining of the calculated path in yellow.

Finally, Figure 72 presents different views of the whole mission execution, where both the vehicle trajectory and the Octomap built along the mission can be observed.

Figure 72 presented a mission in an environment that has been created from a real-world dataset. This test not only evaluates all the aspects that have been covered throughout this thesis, but also represents a clear advance towards fully autonomous inspections. However and as it was mentioned before, there are still some technical aspects that have to be further developed in order to successfully conduct in-water trials. Some of these aspects will be analysed and discussed in the next chapter.

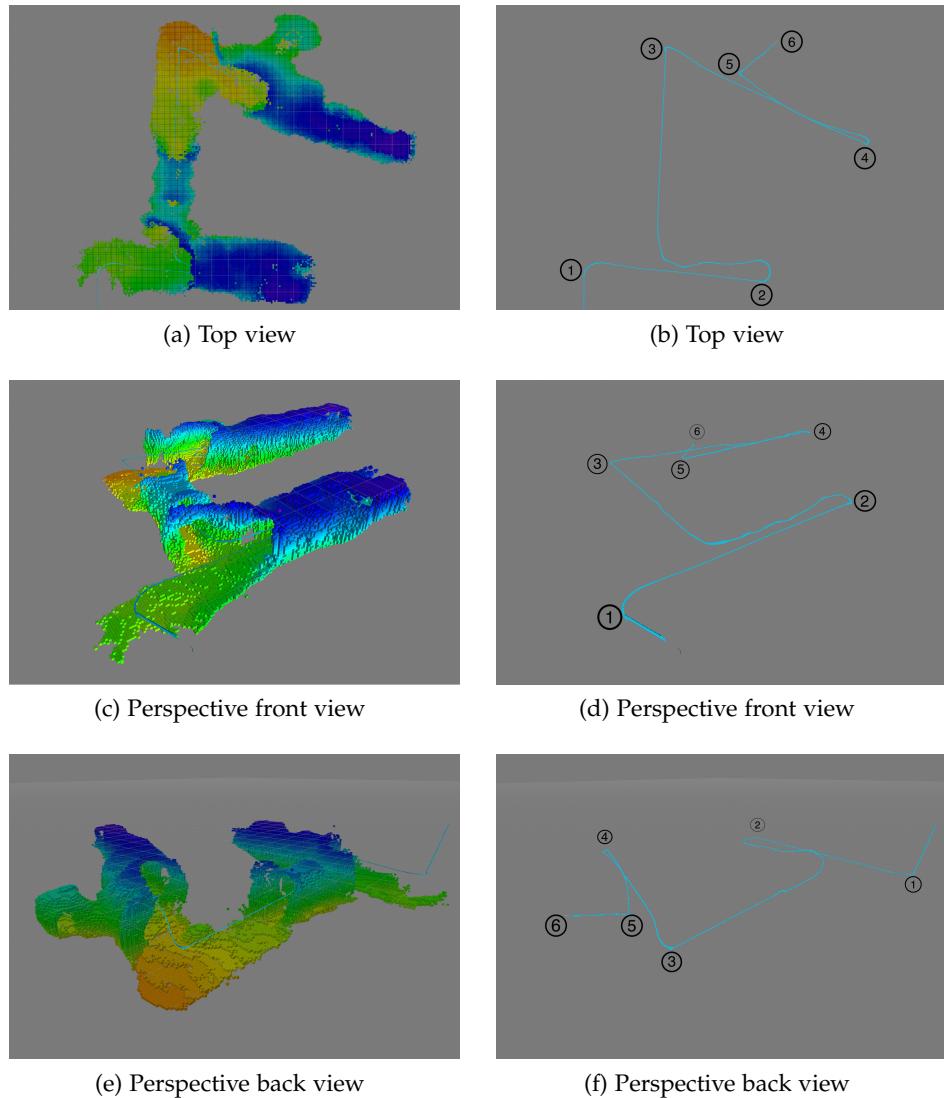


Figure 72: Simulated mission in the caves complex. The whole mission was composed of six start-to-goal queries that were executed consecutively.

6.4 GAP FILLING AND POTENTIAL TARGET INSPECTION

The experiments presented in the previous sections have attempted to prove the capabilities of the proposed path/motion planning framework. But as it was also mentioned before, in order to make some of these kind of missions a reality, there are still additional technical aspects that have to be tackled. For this reason, some of the framework's validation tests have been limited to conduct simulations over real-world datasets. One example of this is the capability to plan feasible 3D paths, which was evaluated with the exploration of confined natural environments. In order to complement the validation of this feature, this section presents another experiment that has been carried out with the AsterX AUV.

In most of the current operational and commercial AUVs, a mission plan is generally defined in advanced by an operator. Such a mission seeks to cover a predefined area from a constant and safe altitude. To do so, the mission commonly consists of the path to be followed and a series of payload commands. These commands establish when and where sensors must acquire different data along the specific route. Therefore, the nature of the data mainly depends on the mission objectives, as well as the sensors installed in the vehicle, e.g., sonars, optical cameras and even physico-chemical sensors.

In this approach of covering a predefined area, the preplanned route is generally not modified during the survey. Instead, the objective is to sweep the area in order to detect and localize potential targets, which can include geological formations [38], shipwrecks [93], and other artificial or natural underwater structures [31]. A common approach in this kind of missions is to conduct a first exhaustive survey, and then extract and analyse the gathered data to program a second and more specific survey. This latter one attempts to obtain more details about the potential targets, as well as to cover possible gaps resulted during the first exploration. However, this two-survey strategy can be inefficient, since it requires establishing a communication link between the AUV and its operator (located in the mother ship) for retrieving the data and reprogramming a new mission. This is particularly unnecessary in modern AUVs with long-term autonomy.

Aiming to overcome some of the aforementioned limitations, one alternative is to endow an AUV with a mission planner, or high-level controller, that extends its decision-making capabilities. This additional control layer must allow the vehicle not only to conduct predefined surveys, but also to autonomously detect and inspect potential subareas of interest without the need of resurfacing. To do so, the AUV is assumed to be equipped with a looking-downward multibeam sonar, which gathers data from the sea bottom along the initial mission. This data can be then automatically processed onboard in order to detect anomalies and gaps, which are marked as regions that require further inspection, thus enlarging the original survey. In the case of anomalies (i.e., objects that protrude above the surrounding seafloor), the vehicle must conduct closer explorations; while in the case of gaps, it must complete covering the area that was initially defined.

To implement this approach over a particular AUV, it is necessary to first understand its control architecture. In what concerns the AsterX, it is composed of three main functional blocks (see Appendix A, Fig. 81): 1) the

low-level controller, or *frontseat*, which runs over QNX; 2) the high-level controller, or *backseat*, which runs over robot operating system ([ROS](#)); and 3) the *payload* controller². Having this in mind, the proposed mission planner can be developed within the *backseat*, which allows including different specialized processes, or nodes as referred to in [ROS](#). Figure 73 depicts the *backseat* controller together with the required nodes. It is also important to notice that these nodes not only are capable of exchanging information between each other, but also can externally communicate with the *frontseat* and payload controllers.

The first of these nodes is called the *target/gap detector*, and it uses the multibeam data gathered along the initial survey to detect targets (anomalies) and gaps. The second and central node is called *mission handler*, and it receives a list of the targets and gaps' positions. Around these positions, this node establishes subareas of interest. The third node is the *path planner*, and it is based on the *planning* module presented in Chapter 5. This node calculates 3D feasible paths to approach and conduct further exploration or coverage of the subareas of interest. Finally, once the planner has provided the paths, the *mission handler* send them to the *path-tracking controller*. This latter node generates the corresponding *frontseat* controller setpoints.

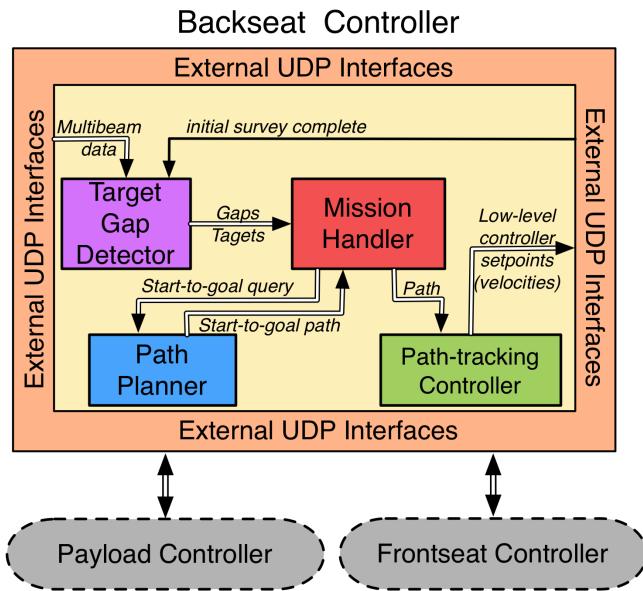


Figure 73: Main modules of the *backseat* controller

An important aspect of the *path planner* used for this experiment is the necessity to take into account the vehicle motion constraints involved. Apart from those constraints associated with a torpedo-shaped [AUV](#), the AsterX can also be limited to navigate at a constant surge speed. This leaves the turning rate as the variable that determines the [AUV](#) turning radius. This formulation coincides with the approach presented in the Chapter 3, thus allowing the use of the *planning* module presented in Chapter 5. In order to validate this new one-survey approach, the AsterX [AUV](#) conducted a real-world inspection by using its *backseat* controller with the

² For more details, the interested readers are encouraged to look into Appendix A

functional nodes mentioned before. Figure 74 depicts the deployment of the AUV before starting the mission.

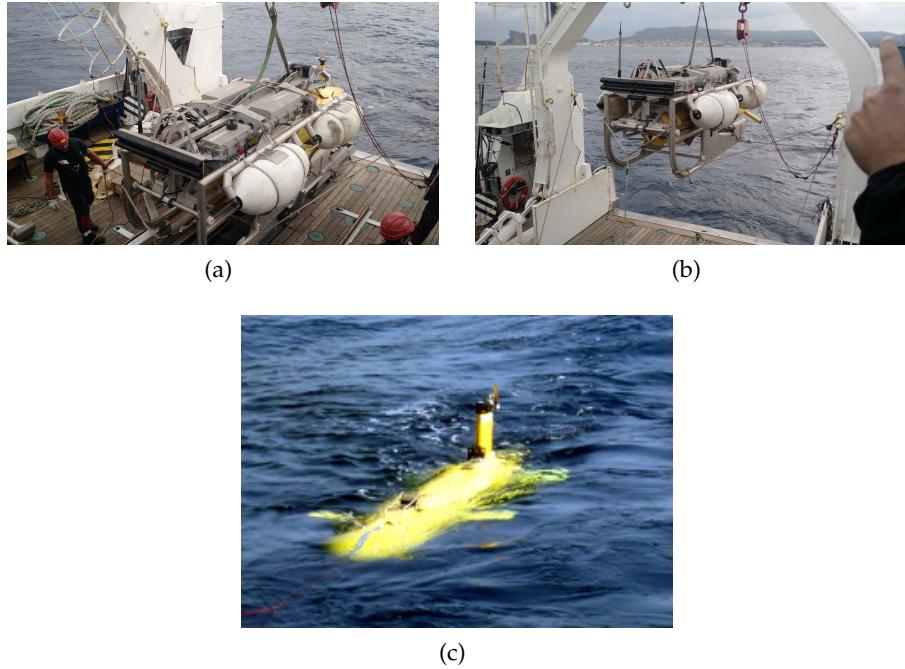


Figure 74: AsterX AUV deployment from its mother ship L'Europe

The mission consisted in detecting and navigating in close proximity to a plane wreck that is located in the bay of La Ciotat, France. Although the approximate location was known, it was not considered accurate enough to send the vehicle closer. Before starting the mission, the AsterX was programmed to follow a predefined coverage survey of the area of interest. To do so, the AUV navigated at 10m deep and with a constant surge speed of 1.5m/s. This initial survey was defined within the *frontseat* controller, which, once completed this first part, handed over the control of the AUV to the *backseat*. Once the *backseat* was informed, it guided the vehicle over a straight line trajectory while keeping the same speed and safe altitude. This maneuver was conducted while the *target/gap detector* processed the multibeam data to detect the potential targets and gaps, if any of them exist. Figure 75 depicts the area of interest, the initial survey, and the potential target and gaps detected from the multibeam data.

After the *detector* provided the locations of the potential targets and gaps, the *path planner* calculated a complementary 3D trajectory to both explore the potential targets and cover the gaps. For the former case, the planner calculated a path 15m over than highest point detected in the target. For the latter case, the planner defined paths to cover the gaps at the deep of the initial survey, i.e., 10m for this particular mission. As the extended mission required navigating at different altitudes, the *planner* calculated 3D paths that met the ascending and descending motion constraints, as it was explained in Chapter 4. Once this calculation was complete, the whole path was sent to *path-tracking controller*. Figure 76 depicts the extended mission to navigate over the potential target and to cover the gaps.

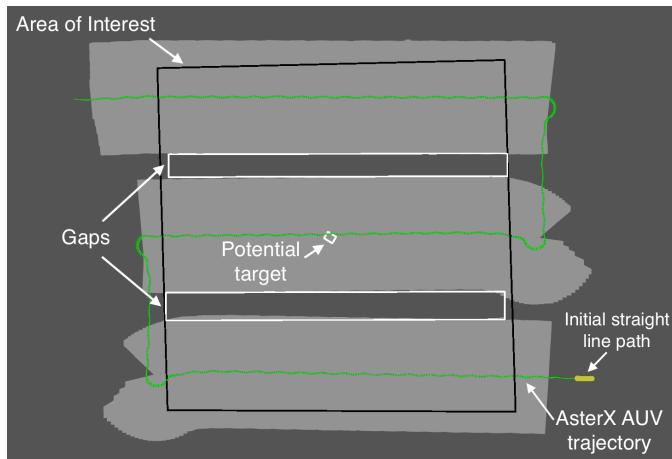


Figure 75: Potential target and gaps detected after completing the initial survey with the AsterX AUV. It can be observed the area of interest in black, the AUV trajectory in green, the area covered with multibeam sonar with light grey, and two gaps and one potential target in white.

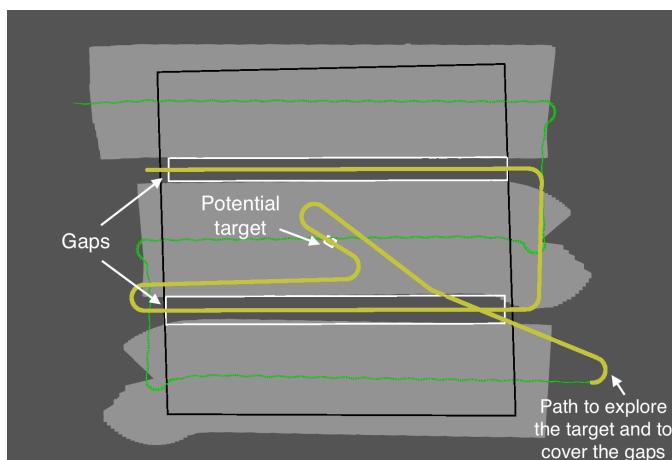


Figure 76: Path planned to further cover the potential target and gaps with the AsterX AUV. It can be observed the new extended mission in yellow. Although the view is from the top, the path is 3D since it requires different altitudes to inspect targets and to cover gaps.

As mentioned in the beginning of this section, this experiment allows further proving the capability of planning 3D AUV paths by using the approach proposed in this thesis. Figure 77 depicts how the AsterX AUV further inspected the target by following a calculated path. To do so, the vehicle descended to a specify altitude with respect to the detected anomaly (target), while conducting turning maneuvers. Both horizontal and vertical motions met the vehicle motion capabilities. Finally, Figure 78 shows the final AUV trajectory after completing the whole mission. It can be observed how the vehicle not only traveled over the target, but also covered the gaps.

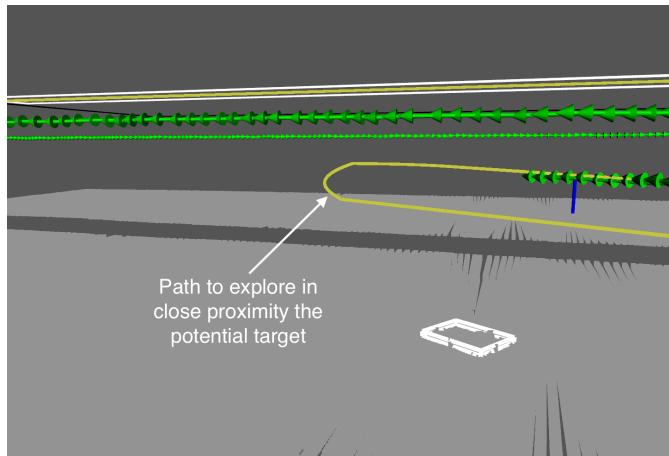


Figure 77: The AsterX AUV approaches to further inspect a potential target. The vehicle trajectory is presented in green, while the calculated path appears in yellow.

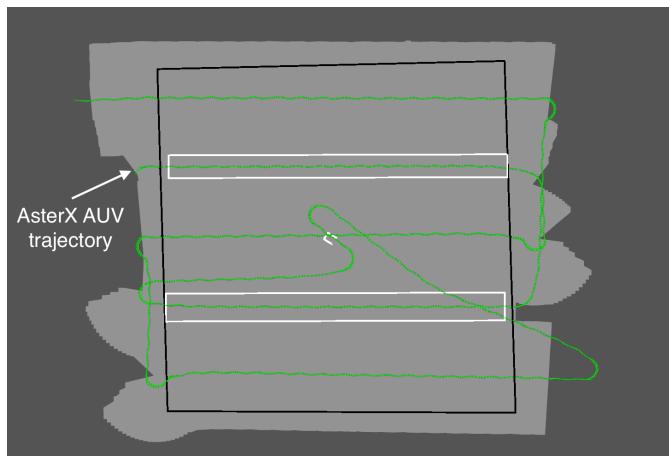


Figure 78: The AsterX AUV completed the mission by inspecting the potential target and covering the gaps from the initial survey. It can be observed the area of interest in black, the AUV trajectory in green, the area covered with multibeam sonar with light grey.

CONCLUSIONS

7.1 SUMMARY OF COMPLETED WORK

This thesis has addressed the problem of planning paths online for an autonomous underwater vehicle ([AUV](#)), which navigates under motion constraints through unexplored environments. This kind of missions require the vehicle to incrementally map the surroundings, while simultaneously replanning a feasible and safe path. After providing an overview of the general problem and the proposed objectives to tackle it, Chapter 2 presented the state of the art of path/motion planning for robotic systems. This review started with discussing the general planning problem, then it classified the available techniques, and finally it discussed their most common extensions. For this latter part, the review put special emphasis on those methods that have been used with underwater vehicles.

Chapter 3 presented the use of sampling-based methods to plan feasible and constant-depth paths for a torpedo-shaped [AUV](#). Here the term feasible refers to paths that take into consideration the vehicle's limitations, either kinematic or dynamic ones. In the case analysed in this thesis, torpedo-shaped [AUVs](#) are kinematically constrained, since they cannot conduct pure lateral motion. Instead, they are required to move forward or backward while conducting turning maneuvers. Two different approaches to consider such limitations were presented and discussed. The first one directly uses the [AUV](#) kinematic equation of motion. The second one employs Dubins curves to characterize the [AUV](#) motion with straight line segments and circular arcs. Both approaches were evaluated and compared in different simulations. This allowed identifying their advantages and drawbacks, which led to establish the one using Dubins curves as the best approach.

Chapter 4 proposed an alternative formulation to plan feasible and variable-depth paths for a torpedo-shaped [AUV](#). This formulation extended the Dubins curves by including the [AUV](#) vertical motion, and it was also used with a sampling-based planning method. This sought to define a general strategy to plan 3D paths, which took into account both the vehicle's lateral and vertical motion constraints. This approach was evaluated in different simulated scenarios, where it was compared with another approach that does not include the vehicle's motion constraints involved. Results proved that the proposed approach allows an [AUV](#) to follow more accurately 3D paths.

While simulation tests in Chapters 3 and 4 assumed fully mapped environments, the main objective of this thesis was to endow an [AUV](#) with the capability to move through unexplored environments. In order to do so, Chapter 5 introduced an online mapping and path/motion planning framework for [AUVs](#). The framework uses an Octomap to incrementally build a representation of the surroundings. To calculate the [AUV](#) path, it includes a sampling-based planner that employs the extended Dubins curves formulation for dealing with 2D and 3D missions. Furthermore, the plan-

ner not only computes a feasible path, but also attempts to obtain a safe one by minimizing its associated risk. Finally, the framework incorporates new strategies that seek to reduce the running time, which is a critical requirement for the intended applications and their online computation limitations. The framework was evaluated with simulations in different scenarios.

In order to completely validate the proposed framework and its properties, Chapter 6 presented different experiments that were conducted by torpedo-shaped AUVs. This included four different real-world scenarios:

1) Planning constant-depth paths to move through artificial marine structures.

In this case, the Sparus II AUV traversed multiple times a breakwater structure, which is composed of a series of concrete blocks separated by four-meter gaps. At the beginning of the mission, the vehicle was not provided with a map of the surroundings. This required the vehicle to incrementally build a map, while planning a feasible and safe path to the different specified goals. Furthermore, the vehicle was equipped with optical cameras to gather images, which were used to create a 3D reconstruction of the traveled area.

2) Planning constant-depth paths to move through natural marine structures.

In this case, the Sparus II navigated through a natural underwater canyon made by rocky formations. This experiment sought to prove the framework's capabilities in a more challenging scenario with non-regular shape obstacles. As occurred in the previous experiment, the vehicle was not provided with an initial map of the area, and it was also equipped with optical cameras. The result of the mission also included a photo-realistic 3D reconstruction of the natural scenario.

3) Planning variable-depth paths to move through confined marine environments. In the previously mentioned experiments, the vehicle navigated at a constant depth due to a sensor limitation. However, in order to validate the capability to plan 3D paths, a simulated Sparus II conducted missions over a real-world dataset of a cave complex. This scenario required solving consecutive start-to-goal queries of different depths. As occurred with the previous experiments, the vehicle was not provided with an initial map of the surroundings.

4) The autonomous survey replanning for gap filling and target inspection. In this case, the AsterX AUV conducted an inspection of an area of interest. The mission was composed of two phases. The first one required the AUV to follow a preplanned coverage path. The second one guided the vehicle to further inspect either potential targets or gaps (i.e., areas not correctly covered). For this latter phase, the extended Dubins curves formulation was used to plan 3D paths to guide the AUV closer from the potential targets. This experiment sought to complement the validation of planning 3D paths with a real-world vehicle.

These tests allowed proving the viability of the proposed approach under real-world conditions. This included the assessment of the computational efficiency, in which a single embedded computer was capable of conducting simultaneously mapping, planning, and control tasks. Furthermore, along the development of the proposed framework, the success rate increased from 10 – 20% during the initial tests, to 80 – 90% when using

the final version of the framework. However, it is also important to mention that those failing cases were mainly due to sensor limitations.

7.2 REVIEW OF CONTRIBUTIONS

Aiming to endow **AUVs** with the capabilities required to operate in unexplored environments, this thesis has proposed an online mapping and path/motion planning framework. This led to extend and develop different strategies that contribute to the current state-of-the-art for underwater vehicles. Such contributions have been presented and peer-reviewed along different **Publications**, and they can be gathered in four main aspects:

PATH / MOTION PLANNING ONLINE This thesis established that in order to move through unexplored environments, an **AUV** has to be capable of incrementally mapping the surroundings, while simultaneously planning collision-free paths. However, an initial approach in this thesis was to have two vehicles to conduct missions in unexplored environments. The first vehicle had to navigate at a safe altitude, and it was assumed to be equipped with a downward-looking multi-beam sonar that allowed to build online a map. Such a map was used to simultaneously plan collision-free paths for a second vehicle that navigated in close-proximity to the sea bottom [NGCUV'15]. Although the previous approach was limited to simulation tests, it did contribute to define a first mechanism for simultaneous mapping and planning [OCEANS'15]. This mechanism later became the framework thoroughly explained in Chapter 5. In its first version, the framework used an anytime tree-pruning strategy that *opportunistically check states for collision* [ICRA'15], but it was later improved by *reusing the last best known solution* [IROS'16].

PLANNING FEASIBLE MOTION FOR AUVS Although a simultaneous mapping and planning mechanism allowed an **AUV** to move through an unexplored environment [ICRA'15], results also showed multiple re-planning maneuvers. The main reason was that the **AUV** was not capable of accurately following the provided paths. This thesis identified the necessity to plan not only collision-free paths, but also feasible ones. This means that the calculated paths must take into consideration the vehicle's motion constraints, thus minimizing unexpected vehicle's trajectories when attempting to follow the calculated path. This thesis proposed and validated the use of Dubins curves to characterize the constant-depth **AUV** motions [IROS'16], which was later extended to consider full 3D trajectories [JFR'17].

PLANNING SAFE PATHS **AUVs** operate in complex environments, where they can be affected by external perturbations such as waves and currents. Furthermore, **AUVs'** navigation system has a position error that cannot be corrected while submerged, especially in environments where a mother ship cannot be used with an **USBL** system. These operation conditions can lead the vehicle to risky situations, especially when moving in close-proximity to nearby obstacles. This thesis evaluated different alternatives to plan not only collision-free

and feasible paths, but also ones that attempt to minimize the risk of collision. A first alternative was to maintain a safe distance from the surroundings [NGCUV'15]. However, it was later replaced with an optimization function that combines the length and the risk associated with a calculated path [IROS'16].

EXPERIMENTAL EVALUATION This thesis has extensively proved the proposed framework and its newly introduced capabilities. This validation mainly involved the Sparus II **AUV**. The experiments included simulated and in-water trials in different scenarios, such as artificial marine structures (breakwater) [ICRA'15, IROS'16], natural marine structures (underwater canyon) [SENSORS'16], confined natural environments (caves complex) [JFR'17]. Furthermore, the capability of planning feasible **3D** paths was validated with the AsterX **AUV** [OCEANS'17].

7.3 FUTURE WORK

This thesis cannot be considered a final and definitive solution for the problem of planning feasible and safe paths for **AUVs**. However, it does contribute a further step towards better and more reliable underwater vehicles. In doing so, this thesis has established the basis for challenging future work that will continue extending the **AUVs'** capabilities.

IN-WATER TRIALS FOR 3D MAPPING AND PLANNING Even though the framework proposed in this thesis supports both **2D** and **3D** missions, its capability of mapping **3D** environments was only evaluated with simulations over virtual and real-world datasets. This was mainly due to technical limitations, such as the absence of a mechanism that allowed the **AUV** to rotate a forward-looking multibeam sonar. Therefore, the next and immediate step to continue this work could be to conduct in-water trials in **3D** scenarios, such as the caves complex mentioned in Section 6.3.

PLANNING VARYING SPEED MOTIONS The framework proposed in this thesis uses a sampling-based method with Dubins curves as a steering function. This formulation assumes that the **AUV** moves with a constant surge speed and a maximum turning rate. In this way, the **AUV** paths can be parameterized with straight line segments and circular arcs of constant radius. However, there are situations in which the vehicle should be capable of conducting tight turns. Some of these situations were presented in Section 6.3, where the **AUV** surge speed was reduced to allow the planner finding a way out of narrow tunnels. Therefore, another possible extension for future work is to analyse alternatives to plan feasible and safe paths with varying speeds, thus permitting maneuvers with different turning radius.

VIEW/INSPECTION PLANNING As it was mentioned when discussing the experiments presented in Chapter 6, this thesis did not have as an objective to provide a strategy to inspect an area or structure of interest. Another possible future work from this thesis can be the

development of methods to efficiently conduct such inspections. In this case, the proposed framework can be used as a low-level layer for a high-level control pipeline. This latter will have to establish the trajectory that AUV should follow in order to fully inspect an area or structure. As an example of this possibility, some of the characteristics of the proposed framework have been already used in a view planning framework for AUVs [RA-LETTERS'17].

PATH / MOTION PLANNING UNDER UNCERTAINTY One of the requirements identified for moving through unexplored environments was to attempt minimizing the risk of collision. Having this mind, this thesis presented an optimization function that allows combining the length and the risk associated with the path. This approach, however, is heuristically established. Another possible future work is to propose an alternative methodology to calculate safe paths. Such a new approach should consider the different sources of uncertainty to establish the validity of the vehicle's states. In this case, the uncertainty might include the one associated with the model, the controller, the navigation system, and/or the exteroceptive sensors that detect nearby objects. Most of the current work on this matter is still computationally expensive, although there are some promising formulations that could be extended for AUVs. Some of these alternatives were analysed during the development of this work, and they are currently under study as an ongoing work at University of Girona [ICRA'18].

APPENDIX

A

EXPERIMENTAL PLATFORMS

The experimental validation of the framework proposed in this thesis has involved two **AUVs**. While most of those tests were conducted with the Sparus II **AUV**, the experiments that proved one of the framework capabilities were done with the AsterX **AUV**. The following sections explain the main software and hardware aspects of both vehicles.

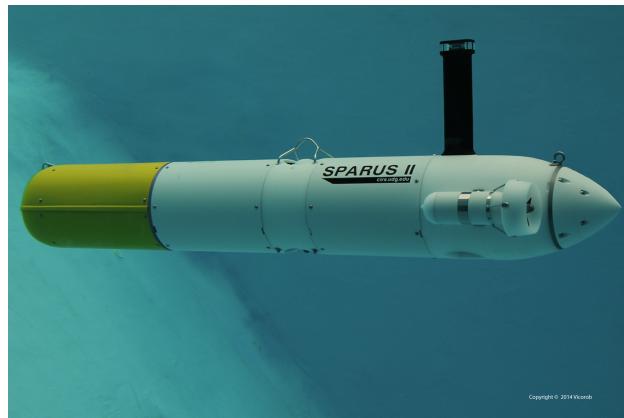
A.1 SPARUS II AUV

The Sparus II is a torpedo-shaped **AUV** with hovering capabilities, which has been designed and developed at the Underwater Vision and Robotics Research Center (**CIRS**)¹. The vehicle is rated for depths up to 200m, and is equipped with three thrusters; two of them are located in the back, and are used for motion on the horizontal plane; the third one is located in the middle, and is dedicated to vertical motion. This implies that the **AUV** can be actuated in surge, heave and yaw **DOF**. Furthermore, the Sparus II is equipped with a navigation sensor suite that includes a pressure sensor, a doppler velocity log (**DVL**), an inertial measurement unit (**IMU**) and a GPS to receive position fixes while at surface.

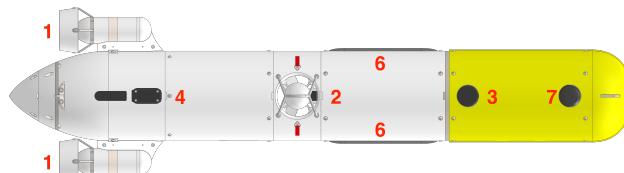
The Sparus II **AUV** also has communication devices such as an acoustic modem for underwater communication with other vehicles or surface stations (e.g., by using an **USBL** system), and a Wi-Fi antenna that can be used when the **AUV** is at surface. Moreover, the vehicle includes a configurable payload area in the front, which contains a set of exteroceptive sensors to perceive and detect the surroundings. This latter group of sensors can be modified according to the mission's requirements, and may include optical cameras, single-beam echosounders, mechanical-scanning (profiler and imaging) sonars, multibeam sonars, etc. Figure 79 depicts different views of the Sparus II **AUV**, including one where a possible payload configuration can be observed.

In what software concerns, the Sparus II **AUV** is controlled through the component oriented layer-based architecture for autonomy (**COLA2**) [101], which is a control architecture that is completely integrated with the robot operating system (**ROS**). Besides operating aboard real robots, **COLA2** can interact with the underwater simulator (**UWSim**) [105], which can import 3D environment models and simulate the vehicle's sensors and dynamics with high fidelity. Furthermore, the use of **ROS** allows to easily integrate third party tools, such as the open motion planning library (**OMPL**) that offers a convenient framework that can be adapted to specific path/motion planning problems [115].

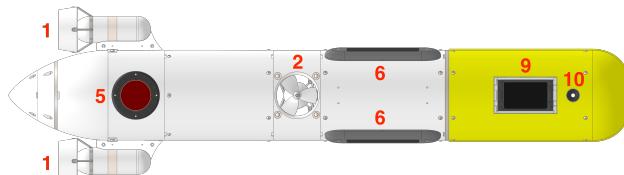
¹ **CIRS** is part of the Computer Vision and Robotics Institute (**ViCOROB**) in Girona (Spain)



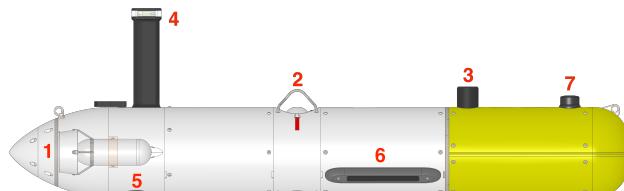
(a) Sparus II in a water tank at CIRS



(b) Top view



(c) Bottom view



(d) Lateral view

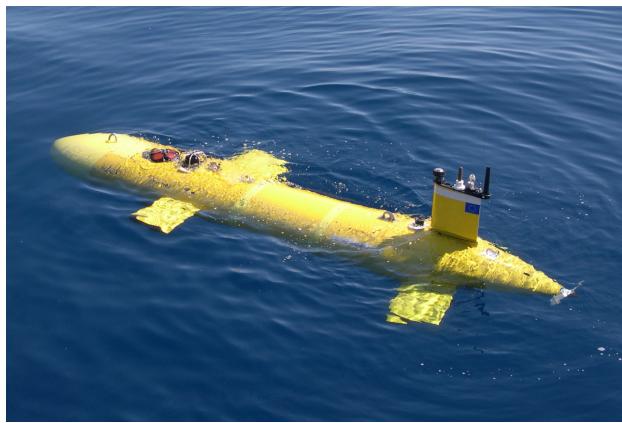


(e) 3D view

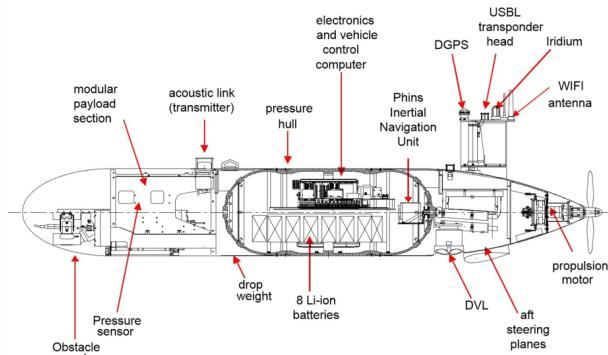
Figure 79: Sparus II AUV: top, bottom, lateral and 3D views. Different hardware parts can be observed, including the thrusters (1,2), the acoustic modem (3), the Wi-Fi and GPS (4), as well as interoceptive and exteroceptive sensors such as the DVL (5), side-scan sonar (6), mechanically-scanning imaging sonar (7), single-beam echosounders (8), multibeam sonar (9), and an optical camera (10).

A.2 ASTERX AUV

The AsterX is a torpedo-shaped [AUV](#) from the French Research Institute for Exploitation of the Sea ([Ifremer](#)). This vehicle is based on the Explorer 3000 [AUV](#), which is built by International Submarine Engineering ([ISE](#)), from Canada. The vehicle is rated for depths up to 3000m, and is equipped with one back propulsion motor, three aft steering planes, and two fore planes. Similarly as the Sparus II, AsterX is equipped with a navigation sensor suite that includes a pressure sensor, a [DVL](#), an [IMU](#) and a GPS to receive position fixes while at surface. It also has an acoustic modem, and a Wi-Fi antenna. This [AUV](#) also includes a modular payload area in the front, which may carry different sensors, e.g., a multibeam sonar. Figure 8o depicts the AsterX [AUV](#) at sea surface, and a schematic with its main inner hardware devices distribution.



(a) AsterX [AUV](#) at sea surface during in-water trials



(b) Lateral view

Figure 8o: AsterX AUV at sea surface and its lateral view with a description of the different hardware elements.

On the other hand, the AsterX's software architecture is composed of three main functional blocks (see Fig. 81). Firstly, the [AUV](#) low-level controller, also referred as *frontseat*, guides the vehicle using the automated control engine ([ACE](#)) middleware. This controller is executed over QNX operating system, thus guaranteeing real-time computation constraints. Furthermore, this functional block also handles the vehicle's navigation and safety routines. Secondly, the *backseat* controller extends the vehicle's capabilities and applications by easing the implementation of high-level

routines, such as algorithms for path/motion planning, path-tracking, and docking. These routines can directly send low-level control setpoints. This functional block has its own dedicated computer that works under the ROS (over Linux). Finally, the *payload* controller acts as a bidirectional interface between the interoceptive and exteroceptive sensors, and, on the other hand, both the *frontseat* and *backseat* controllers.

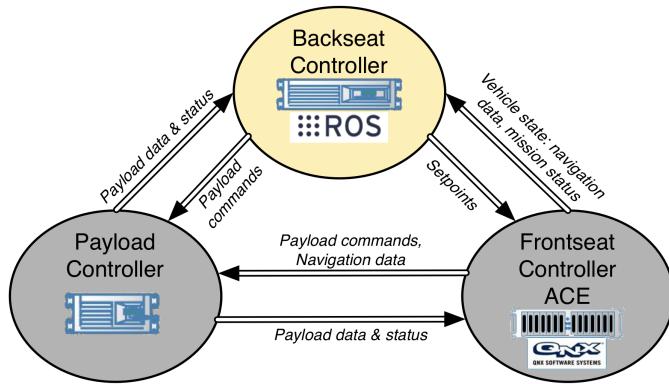


Figure 81: AsterX AUV software architecture

BIBLIOGRAPHY

- [1] Alberto Alvarez, Andrea Caiti, and Reiner Onken. "Evolutionary Path Planning for Autonomous Underwater Vehicles in a Variable Ocean." In: *IEEE Journal of Oceanic Engineering* 29.2 (2004), pp. 418–429.
- [2] Shinji Arinaga, Shoji Nakajima, Hideo Okabe, Akira Ono, and Yutaka Kanayama. "A motion planning method for an AUV." In: *Symposium on Autonomous Underwater Vehicle Technology*. 1996, pp. 477–484.
- [3] Franz Aurenhammer. "Voronoi diagrams—a survey of a fundamental geometric data structure." In: *ACM Computing Surveys* 23.3 (1991), pp. 345–405.
- [4] Jerome Barraquand and J.-C. Latombe. "A Monte-Carlo algorithm for path planning with many degrees of freedom." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1990, pp. 1712–1717.
- [5] Jerome Barraquand and Jean-Claude Latombe. "Robot Motion Planning: A Distributed Representation Approach." In: *The International Journal of Robotics Research* 10.6 (1991), pp. 628–649.
- [6] Jerome Barraquand, Lydia E. Kavraki, Jean-Claude Latombe, Rajeev Motwani, Tsai-yen Li, and Prabhakar Raghavan. "A Random Sampling Scheme for Path Planning." In: *The International Journal of Robotics Research* 16.6 (1997), pp. 759–774.
- [7] Kostas E. Bekris and Lydia E. Kavraki. "Greedy but Safe Replanning under Kinodynamic Constraints." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 704–710.
- [8] Khaled Belhith, Froduald Kabanza, Leo Hartman, and Roger Nkambou. "Anytime dynamic path-planning with flexible probabilistic roadmaps." In: *IEEE International Conference on Robotics and Automation (ICRA)*. May. 2006, pp. 2372–2377.
- [9] Jur van den Berg, Dave Ferguson, and James Kuffner. "Anytime path planning and replanning in dynamic environments." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2006, pp. 2366–2371.
- [10] Pierre Bessiere, Juan Manuel Ahuactzin, El-Ghazali Talbi, and Emmanuel Mazer. "The "Ariadne's clew" algorithm: global planning with local methods." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 2. IEEE, 1993, pp. 1373–1380.
- [11] Brian Bingham, Brendan Foley, Hanumant Singh, Richard Camilli, Katerina Delaporta, Ryan Eustice, Angelos Mallios, David Mindell, Christopher Roman, and Dimitris Sakellariou. "Robotic tools for deep water archaeology: Surveying an ancient shipwreck with an

- autonomous underwater vehicle." In: *Journal of Field Robotics* 27.6 (2010), pp. 702–717.
- [12] Robert Bohlin and Lydia E. Kavraki. "Path planning using lazy PRM." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. 2000, pp. 521–528.
 - [13] Robert Bohlin and Lydia E. Kavraki. "A Randomized Algorithm for Robot Path Planning Based on Lazy Evaluation." In: *Handbook on Randomized Computing*. Kluwer Academic Publishers, 2001, pp. 221–249.
 - [14] Charmane V Caldwell, Damion D. Dunlap, and Emmanuel G. Collins. "Motion planning for an autonomous Underwater Vehicle via Sampling Based Model Predictive Control." In: *MTS/IEEE OCEANS*. 2010, pp. 1–6.
 - [15] Ricard Campos, Rafael García, Pierre Alliez, and Mariette Yvinec. "Splat-based surface reconstruction from defect-laden point sets." In: *Graphical Models* 75.6 (2013), pp. 346–361. ISSN: 15240703.
 - [16] John Canny. "Computing Roadmaps of General Semi-Algebraic Sets." In: *The Computer Journal* 36.5 (1993), pp. 504–514.
 - [17] Junliang Cao, Junjun Cao, Zheng Zeng, and Lian Lian. "Optimal path planning of underwater glider in 3D dubins motion with minimal energy consumption." In: *MTS/IEEE OCEANS*. IEEE, 2016, pp. 1–7.
 - [18] Kevin P. Carroll, Stephen R. McClaran, Eric L. Nelson, David M. Barnett, Donald K. Friesen, and Glen N. William. "AUV path planning: an A* approach to path planning with consideration of variable vehicle speeds and multiple, overlapped with, time-dependent exclusion zones." In: *Symposium on Autonomous Underwater Vehicle Technology*. 1992, pp. 79–84.
 - [19] Chi-Tsun Cheng, Kia Fallahi, Henry Leung, and Chi K Tse. "An AUVs path planner using genetic algorithms with a deterministic crossover operator." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 2995–3000.
 - [20] Howie Choset, Kevin M Lynch, Seth Hutchinson, George A Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.
 - [21] Thomas Dean and Mark Boddy. "An Analysis of Time-Dependent Planning." In: *Seventh AAAI National Conference on Artificial Intelligence (AAAI)*. Vol. 4954. 6. 1988, pp. 49–54.
 - [22] Didier Devaurs, Thierry Simeon, and Juan Cortes. "Optimal Path Planning in Complex Cost Spaces With Sampling-Based Algorithms." In: *IEEE Transactions on Automation Science and Engineering* 13.2 (2016), pp. 415–424.
 - [23] Edsger W. Dijkstra. "A note on two problems in connexion with graphs." In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

- [24] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. "Practical Search Techniques in Path Planning for Autonomous Driving." In: *First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*. 2008.
- [25] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments." In: *The International Journal of Robotics Research* 29.5 (2010), pp. 485–501.
- [26] Bruce Donald, Patrick Xavier, John Canny, and John Reif. "Kinodynamic Motion Planning." In: *Journal of the ACM* 40.5 (1993), pp. 1048–1066.
- [27] Lester Dubins. "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents." In: *American Journal of mathematics* 79.3 (1957), pp. 497–516.
- [28] Matthew Dunbabin and Lino Marques. "Robots for environmental monitoring: Significant advancements and applications." In: *IEEE Robotics and Automation Magazine* 19.1 (2012), pp. 24–39.
- [29] Brendan Englot and Franz S. Hover. "Inspection planning for sensor coverage of 3D marine structures." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010, pp. 4412–4417.
- [30] Brendan Englot and Franz S. Hover. "Three-dimensional coverage planning for an underwater inspection robot." In: *The International Journal of Robotics Research* 32 (2013), pp. 1048–1073.
- [31] Javier Escartin, Ricard Garcia, T Barreyre, Mathilde Cannat, Nuno Gracias, and ASM Shihavuddin. "Optical methods to monitor temporal changes at the seafloor: The Lucky Strike deep-sea hydrothermal vent field (Mid-Atlantic Ridge)." In: (2013).
- [32] *FCL, Flexible Collision Library*. <https://github.com/flexible-collision-library/fcl>.
- [33] Cheng Fang and Stuart Anstee. "Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle." In: *MTS/IEEE OCEANS*. Sydney, 2010.
- [34] Dave Ferguson and Anthony Stentz. "Anytime RRTs." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. line 3. 2006, pp. 5369–5375.
- [35] Dave Ferguson and Anthony Stentz. "Anytime, Dynamic Planning in High-dimensional Search Spaces." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 1310–1315.
- [36] Christina Forney et al. "Surface Reconstruction of Maltese Cisterns Using ROV Sonar Data for Archeological Study." In: *Advances in Visual Computing: 7th International Symposium*. Vol. 6938. Springer Berlin Heidelberg, 2011, pp. 461–471.
- [37] Enric Galceran. "Coverage Path Planning for Autonomous Underwater Vehicles." PhD's Thesis. Universitat de Girona, 2014.

- [38] Enric Galceran and Marc Carreras. "Efficient seabed coverage path planning for ASVs and AUVs." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 88–93.
- [39] Enric Galceran and Marc Carreras. "A survey on coverage path planning for robotics." In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1258–1276.
- [40] Enric Galceran and Marc Carreras. "Planning Coverage Paths on Bathymetric Maps for In-Detail Inspection of the Ocean Floor." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, 2013.
- [41] Enric Galceran, Sharad Nagappa, Marc Carreras, Pere Ridao, and Albert Palomer. "Uncertainty-driven survey path planning for bathymetric mapping." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013, pp. 6006–6012.
- [42] Enric Galceran, Ricard Campos, Narcís Palomeras, David Ribas, Marc Carreras, and Pere Ridao. "Coverage Path Planning with Real-time Replanning and Surface Reconstruction for Inspection of Three-dimensional Underwater Structures using Autonomous Underwater Vehicles." In: *Journal of Field Robotics* (2014).
- [43] Mark Grasmueck, Gregor P. Eberli, David A. Viggiano, Thiago Correa, Glenda Rathwell, and Jiangang Luo. "Autonomous underwater vehicle (AUV) mapping reveals coral mound distribution, morphology, and oceanography in deep water of the Straits of Florida." In: *Geophysical Research Letters* 33.23 (2006), p. L23616.
- [44] Peter Hart, Nils Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.
- [45] Young Jin Heo and Wan Kyun Chung. "RRT-based path planning with kinematic constraints of AUV in underwater structured environment." In: *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2013, pp. 523–525.
- [46] Juan David Hernández, Klemen Istenic, Nuno Gracias, Rafael García, Pere Ridao, and Marc Carreras. "Autonomous Seabed Inspection for Environmental Monitoring." In: *ROBOT2015: Second Iberian Robotics Conference*. Lisbon, Portugal: Springer International Publishing, 2016, pp. 27–39.
- [47] Juan David Hernández, Klemen Istenič, Nuno Gracias, Narcís Palomeras, Ricard Campos, Eduard Vidal, Rafael García, and Marc Carreras. "Autonomous Underwater Navigation and Optical Mapping in Unknown Natural Environments." In: *Sensors* 16.8 (2016), p. 1174.
- [48] Geoffrey A. Hollinger, Brendan Englot, Franz S. Hover, Urbashi Mitra, and Gaurav S. Sukhatme. "Active planning for underwater inspection and the benefit of adaptivity." In: *The International Journal of Robotics Research* 32.1 (2013), pp. 3–18.

- [49] Wang Hong-jian, Bian Xin-qia, Zhang Xu, Fu Ming-yu, and Li Juan. "Two approaches for autonomous underwater vehicle global path planning in large range ocean environment." In: *International Conference on Intelligent Mechatronics and Automation (ICIMA)*. August. 2004, pp. 224–227.
- [50] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. "OctoMap: an efficient probabilistic 3D mapping framework based on octrees." In: *Autonomous Robots* 34.3 (2013), pp. 189–206.
- [51] Sarah E Houts and Stephen M Rock. "Trajectory planning for motion-constrained AUVs in uncertain environments." In: *MTS/IEEE OCEANS*. 2014, pp. 1–5.
- [52] Sarah E Houts, Stephen M Rock, and Rob McEwen. "Aggressive terrain following for motion-constrained AUVs." In: *IEEE/OES Autonomous Underwater Vehicles (AUV)*. 2012, pp. 1–7.
- [53] Franz S. Hover, Ryan M. Eustice, Ayoung Kim, Brendan Englot, Hordur Johannsson, Michael Kaess, and John J. Leonard. "Advanced perception, navigation and planning for autonomous in-water ship hull inspection." In: *The International Journal of Robotics Research* 31.12 (2012), pp. 1445–1464.
- [54] David Hsu. "Randomized Single-query Motion Planning in Expansive Spaces." PhD's Thesis. Stanford University, 2000, p. 118.
- [55] David Hsu, Jean-Claude Latombe, and Rajeev Motwani. "Path planning in expansive configuration spaces." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3. April. 1997, pp. 2719–2726.
- [56] David Hsu, Jean-Claude Latombe, and Rajeev Motwani. "Path planning in expansive configuration spaces." In: *International Journal of Computational Geometry & Applications* 09.04no5 (1999), pp. 495–512.
- [57] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. "Randomized Kinodynamic Motion Planning with Moving Obstacles." In: *The International Journal of Robotics Research* 21.3 (2002), pp. 233–255.
- [58] Natàlia Hurtós, David Ribas, Xavier Cufí, Yvan Petillot, and Joaquim Salvi. "Fourier-based Registration for Robust Forward-looking Sonar Mosaicing in Low-visibility Underwater Environments." In: *Journal of Field Robotics* 32.1 (2015), pp. 123–151.
- [59] Léonard Jaillet, Juan Cortés, and Thierry Simeon. "Sampling-Based Path Planning on Configuration-Space Costmaps." In: *IEEE Transactions on Robotics* 26.4 (2010), pp. 635–646.
- [60] Ishay Kamon, Ehud Rivlin, and Elon Rimon. "A new range-sensor based globally convergent navigation algorithm for mobile robots." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1996, pp. 429–435.
- [61] Sertac Karaman and Emilio Frazzoli. "Incremental Sampling-based Algorithms for Optimal Motion Planning." In: *Robotics: Science and Systems (RSS)*. Zaragoza, Spain, 2010.

- [62] Sertac Karaman and Emilio Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning." In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. arXiv: [1105.1186](https://arxiv.org/abs/1105.1186).
- [63] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. "Anytime Motion Planning using the RRT*." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 1478–1483.
- [64] Lydia E. Kavraki. "Random Networks In Configuration Space For Fast Path Planning." PhD's Thesis. Stanford University, 1994.
- [65] Lydia E. Kavraki, Mihail N. Kolountzakis, and Jean-Claude Latombe. "Analysis of probabilistic roadmaps for path planning." In: *IEEE Transactions on Robotics and Automation* 14.1 (1998), pp. 166–171.
- [66] Lydia E. Kavraki and Jean-Claude Latombe. "Randomized preprocessing of configuration for fast path planning." In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Comput. Soc. Press, 1994, pp. 2138–2145.
- [67] Lydia E. Kavraki, Pter Švestka, Jean-Claude Latombe, and Mark H. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.
- [68] Faisal I. Khan and Paul R. Amyotte. "Inherent safety in offshore oil and gas activities: A review of the present status and future directions." In: *Journal of Loss Prevention in the Process Industries* 15.4 (2002), pp. 279–289.
- [69] Oussama Khatib. "Real-time obstacle avoidance for manipulators and mobile robots." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. Institute of Electrical and Electronics Engineers, 1985, pp. 500–505.
- [70] Oussama Khatib. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." In: *The International Journal of Robotics Research* 5.1 (1986), pp. 90–98.
- [71] Hangeun Kim, Taehwan Lee, and Hyun Chung. "Any-angle path planning with limit-cycle circle set for marine surface vehicle." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2012, pp. 2275–2280.
- [72] James J. Kuffner and Steven M. LaValle. "RRT-connect: An efficient approach to single-query path planning." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. April. 2000, pp. 995–1001.
- [73] Yoshiaki Kuwata, Sertac Karaman, Justin Teo, Emilio Frazzoli, Jonathan P. How, and Gaston Fiore. "Real-Time Motion Planning With Applications to Autonomous Urban Driving." In: *IEEE Transactions on Control Systems Technology* 17.5 (2009), pp. 1105–1118.
- [74] Steven M. LaValle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. Computer Science Dept., Iowa State University, 1998.

- [75] Steven M. LaValle. *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.
- [76] Steven M. LaValle and James J. Kuffner. "Randomized kinodynamic planning." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. 1999, pp. 473–479.
- [77] Steven M. LaValle and James J. Kuffner. "Rapidly-exploring random trees: Progress and prospects." In: *Algorithmic and Computational Robotics: New Directions*. 2000, pp. 293–308.
- [78] Steven M. LaValle and James J. Kuffner. "Randomized Kinodynamic Planning." In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400.
- [79] Katherine Lam, Paul K S Shin, Robin Bradbeer, David Randall, Kenneth K K Ku, Paul Hodgson, and Siu Gin Cheung. "A comparison of video and point intercept transect methods for monitoring subtropical coral communities." In: *Journal of Experimental Marine Biology and Ecology* 333.1 (2006), pp. 115–128.
- [80] Jean-Claude Latombe. *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.
- [81] Maxim Likhachev and Dave Ferguson. "Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles." In: *The International Journal of Robotics Research* 28.8 (2009), pp. 933–945.
- [82] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. "ARA*: Anytime A * with Provable Bounds on Sub-Optimality." In: *Advances in Neural Information Processing Systems*. 2003.
- [83] Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun. "Anytime Dynamic A*: An Anytime, Replanning Algorithm." In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2005.
- [84] Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun. "Anytime search in dynamic graphs." In: *Artificial Intelligence* 172 (2008), pp. 1613–1643.
- [85] Yuecheng Liu and Yongjia Zhao. "A virtual-waypoint based artificial potential field method for UAV path planning." In: *IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*. 2016, pp. 949–953. ISBN: 978-1-4673-8318-9.
- [86] Tomás Lozano-Pérez. "Spatial Planning: A Configuration Space Approach." In: *IEEE Transactions on Computers* C-32.2 (1983), pp. 108–120.
- [87] Tomás Lozano-Pérez. "A simple motion-planning algorithm for general robot manipulators." In: *IEEE Journal on Robotics and Automation* 3.3 (1987), pp. 224–238.
- [88] Tomás Lozano-Pérez, Mattehew T Mason, and Russell H Taylor. "Automatic synthesis of fine-motion strategies for robots." In: *The International Journal of Robotics Research* (1984).

- [89] Vladimir Lumelsky and Alexander Stepanov. "Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape." In: *Algorithmica* (1987), pp. 403–430.
- [90] Toshihiro Maki, Hayato Mizushima, Hayato Kondo, Tamaki Ura, Takashi Sakamaki, and Masao Yanagisawa. "Real time path-planning of an AUV based on characteristics of passive acoustic landmarks for visual mapping of shallow vent fields." In: *MTS/IEEE OCEANS*. Vancouver, 2007.
- [91] Angelos Mallios, Pere Ridao, David Ribas, Marc Carreras, and Richard Camilli. "Toward autonomous exploration in confined underwater environments." In: *Journal of Field Robotics* 33.7 (2015), 994—1012.
- [92] Angelos Mallios, Eduard Vidal, Ricard Campos, and Marc Carreras. "Underwater caves sonar and vision data set." In: *International Journal of Robotic Research* (2017), (under review).
- [93] "Mapping the Moon: Using a lightweight AUV to survey the site of the 17th century ship 'La Lune'." In: *OCEANS 2013 MTS/IEEE Bergen: The Challenges of the Northern Dimension* (2013).
- [94] Emmanuel Mazer, Juan Manuel Ahuactzin, and Pierre Bessiere. "The Ariadne's Clew Algorithm." In: *Journal of Artificial Intelligence Research* 9 (1998), pp. 295–316.
- [95] James McMahon and Erion Plaku. "Mission and Motion Planning for Autonomous Underwater Vehicles Operating in Spatially and Temporally Complex Environments." In: *IEEE Journal of Oceanic Engineering* 3 (2016), pp. 1–20.
- [96] Nikola Mišković. "Use of Self-Oscillations in Guidance and Control of Marine Vessels." PhD thesis. University of Zagreb, 2010, p. 189.
- [97] Jörg Müller, Norman Kohler, and Wolfram Burgard. "Autonomous miniature blimp navigation with online motion planning and re-planning." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011, pp. 4941–4946.
- [98] Kiran Murthy and Stephen Rock. "Spline-based trajectory planning techniques for benthic AUV operations." In: *IEEE/OES Autonomous Underwater Vehicles (AUV)*. 2010, pp. 1–9.
- [99] Nils Nilsson. "A mobile automation: an application of artificial intelligence techniques." In: *1st International Joint Conference on Artificial Intelligence*. 1969, pp. 509–520.
- [100] Mark H. Overmars and Pter Švestka. *A probabilistic learning approach to motion planning*. Tech. rep. Utrecht University, 1994.
- [101] Narcis Palomeras, Andres El-Fakdi, Marc Carreras, and Pere Ridao. "COLA2: A Control Architecture for AUVs." In: *IEEE Journal of Oceanic Engineering* 37.4 (2012), pp. 695–716.
- [102] Yvan Petillot, Ioseba Tena Ruiz, and David M. Lane. "Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar." In: *IEEE Journal of Oceanic Engineering* 26.2 (2001), pp. 240–251.

- [103] Clement Petres, Yan Pailhas, Pedro Patron, Yvan Petillot, Jonathan Evans, and David Lane. "Path Planning for Autonomous Underwater Vehicles." In: *IEEE Transactions on Robotics* 23.2 (2007), pp. 331–341.
- [104] Jann Poppinga, Andreas Birk, Kaustubh Pathak, and Narunas Vaskevicius. "Fast 6-DOF path planning for Autonomous Underwater Vehicles (AUV) based on 3D plane mapping." In: *IEEE International Symposium on Safety, Security, and Rescue Robotics*. 2011, pp. 345–350.
- [105] Mario Prats, Javier Perez, J. Javier Fernandez, and Pedro J Sanz. "An open source tool for simulation and supervision of underwater intervention missions." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 2577–2582.
- [106] Z Qu and H Yuan. "Optimal real-time collision-free motion planning for autonomous underwater vehicles in a 3D underwater space." In: *IET Control Theory & Applications* 3.6 (2009), pp. 712–721.
- [107] João Sequeira and M.I. Ribeiro. "A two level approach for underwater path planning." In: *MTS/IEEE OCEANS*. Vol. 2. Brest, 1994, pp. II/87 –II/91.
- [108] Andrei M Shkel and Vladimir Lumelsky. "Classification of the Dubins set." In: *Robotics and Autonomous Systems* 34.4 (2001), pp. 179–202.
- [109] Ryan N. Smith, Peggy P. Li, David A. Caron, Burton H. Jones, and Gaurav S. Sukhatme. "Planning and Implementing Trajectories for Autonomous Underwater Vehicles to Track Evolving Ocean Processes Based on Predictions from a Regional Ocean Model." In: *The International Journal of Robotics Research* 29.12 (2010), pp. 1475–1497.
- [110] Michaël Soulignac. "Feasible and Optimal Path Planning in Strong Current Fields." In: *IEEE Transactions on Robotics* 27.1 (2011), pp. 89–98.
- [111] Jason R. Stack and Cheryl M. Smith. "Combining random and data-driven coverage planning for underwater mine detection." In: *MTS/IEEE OCEANS*. San Diego, CA, 2003.
- [112] Anthony Stentz. "Optimal and efficient path planning for partially-known environments." In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Comput. Soc. Press, 1994, pp. 3310–3317.
- [113] Anthony Stentz. "The focussed D* algorithm for real-time replanning." In: *14th International Joint Conference on Artificial intelligence (IJCAI)*. Vol. 14. August. Morgan Kaufmann Publishers Inc., 1995, pp. 1652–1659.
- [114] Ioan Alexandru Sucan. "Task and motion planning for mobile manipulators." PhD's Thesis. Rice University, 2011.
- [115] Ioan Alexandru Sucan, Mark Moll, and Lydia E. Kavraki. "The Open Motion Planning Library." In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

- [116] Kazuo Sugihara and Junku Yuh. "GA-based motion planning for underwater robotic vehicles." In: *10th International Symp. on Unmanned Untethered Submersible Technology*. Durham, NH, USA, 1997, pp. 406–415.
- [117] Chiew Tan, Robert Sutton, and John Chudley. "An incremental stochastic motion planning technique for autonomous underwater vehicles." In: *IFAC Control Applications in Marine Systems Conference*. 2004, pp. 483–488.
- [118] David R Thompson, Steve Chien, Peggy Li, Bronwyn Cahill, Julia Levin, Oscar Schofield, Arjuna Balasuriya, Stephanie Petillo, Matthew Arrott, and Michael Meisinger. "Spatiotemporal path planning in strong, dynamic, uncertain currents." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 4778–4783.
- [119] Sebastian Thrun et al. "Stanley: The robot that won the DARPA Grand Challenge." In: *Journal of Field Robotics* 23.9 (2006), pp. 661–692.
- [120] Konstantinos I. Tsianos, Ioan Alexandru Sucan, and Lydia E. Kavraki. "Sampling-based robot motion planning: Towards realistic applications." In: *Computer Science Review* 1.1 (2007), pp. 2–11.
- [121] Nikolaus Vahrenkamp, Tamim Asfour, and Rüdiger Dillmann. "Efficient motion planning for humanoid robots using lazy collision checking and enlarged robot Models." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2007, pp. 3062–3067.
- [122] Eduard Vidal, Juan David Hernández, Klemen Istenič, and Marc Carreras. "Online View Planning for Inspecting Unexplored Underwater Structures." In: *IEEE Robotics and Automation Letters* PP.99 (2017), pp. 1–1. ISSN: 2377-3766.
- [123] Sabiha Wadoo and Pushkin Kachroo. *Autonomous Underwater Vehicles: Modeling, Control Design and Simulation*. CRC Press, 2010, p. 143.
- [124] Charles W. Warren. "A technique for autonomous underwater vehicle route planning." In: *Symposium on Autonomous Underwater Vehicle Technology*. Vol. 15. 3. 1990, pp. 201–205.
- [125] Bilal Wehbe, Elie Shammas, Joseph Zeaiter, and Daniel Asmar. "Dynamic modeling and path planning of a hybrid autonomous underwater vehicle." In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2014, pp. 729–734.
- [126] Wei Li, Jay A. Farrell, Shuo Pang, and Richard M. Arrieta. "Moth-inspired chemical plume tracing on an autonomous underwater vehicle." In: *IEEE Transactions on Robotics* 22.2 (2006), pp. 292–307.
- [127] Louis L. Whitcomb. "Underwater robotics: out of the research laboratory and into the field." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. IEEE, 2000, pp. 709–716.
- [128] David P Williams. "On optimal AUV track-spacing for underwater mine detection." In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 4755–4762.

- [129] Kwangjin Yang and Salah Sukkarieh. “3D smooth path planning for a UAV in cluttered natural environments.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2008, pp. 794–800.
- [130] Nan Ying Nan Ying, L Eicher, Wang Xunzhang Wang Xunzhang, G G L Seet, and M W S Lau. “Real-time 3D path planning for sensor-based underwater robotics vehicles in unknown environment.” In: *MTS/IEEE OCEANS*. Vol. 3. 65. 2000, pp. 2051–2058.
- [131] Shlomo Zilberstein and Stuart Russel. “Approximate reasoning using anytime algorithms.” In: *Imprecise and Approximate Computation*. Ed. by Swaminathan Natarajan. Vol. 318. The Springer International Series in Engineering and Computer Science. Boston, MA: Springer US, 1995.