



UNIVERSIDAD DE MÁLAGA

DOCTORAL THESIS

Extreme Path Planning for Exploration Mobile Robots

Author:

Jose Ricardo SÁNCHEZ IBÁÑEZ

Supervisors:

Dr. Carlos Jesús PÉREZ DEL PULGAR MANCEBO

Prof. Dr. Alfonso GARCÍA CEREZO

*A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy*

in the

Space Robotics Laboratory
Department of Systems Engineering and Automation
Escuela de Ingenierías Industriales
Universidad de Málaga


Doctorado en Ingeniería Mecatrónica

2022



UNIVERSIDAD
DE MÁLAGA

AUTOR: José Ricardo Sánchez Ibáñez

 <https://orcid.org/0000-0002-5130-3808>

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:

<http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>

Cualquier parte de esta obra se puede reproducir sin autorización pero con el reconocimiento y atribución de los autores.

No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): riuma.uma.es



UNIVERSIDAD
DE MÁLAGA



Escuela de Doctorado

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D./Dña JOSE RICARDO SÁNCHEZ IBÁÑEZ

Estudiante del programa de doctorado EN INGENIERÍA MECATRÓNICA de la Universidad de Málaga, autor/a de la tesis, presentada para la obtención del título de doctor por la Universidad de Málaga, titulada: EXTREME PATH PLANNING FOR EXPLORATION MOBILE ROBOTS

Realizada bajo la tutorización de ALFONSO GARCÍA CEREZO y dirección de CARLOS J. PÉREZ DEL PULGAR MANCEBO Y ALFONSO GARCÍA CEREZO (si tuviera varios directores deberá hacer constar el nombre de todos)

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante a la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente.

En Málaga, a 25 de ABRIL de 2022

Fdo.: JOSE RICARDO SÁNCHEZ IBÁÑEZ Doctorando/a	Fdo.: ALFONSO GARCÍA CEREZO Tutor/a
Fdo.: CARLOS J. PÉREZ DEL PULGAR MANCEBO Y ALFONSO GARCÍA CEREZO Director/es de tesis	



EFQM AENOR



Edificio Pabellón de Gobierno. Campus El Ejido.
29071
Tel.: 952 13 10 28 / 952 13 14 61 / 952 13 71 10
E-mail: doctorado@uma.es

Dedicado a mis hermanas y a mi padre.

Agradecimientos - Acknowledgements

Una parte del camino es el final. Mientras escribo estas palabras la producción de esta tesis que lees está llegando a su fin. Este es el desenlace de una aventura que ha ido desarrollándose durante ya más de un lustro. Ha coincidido con unos años que han sido tremendamente difíciles para mí, sobretodo a nivel emocional. Salgo de este proceso como una persona cambiada, evolucionada. Este viaje también me ha proporcionado un crecimiento enorme como persona, y ha sido muy en gran parte posible gracias a personas maravillosas que me han apoyado tanto en los buenos como en los malos momentos. Esta sección va para esas personas.

En primer lugar quiero agradecer a mi familia todo el amor, el apoyo y el cariño que me han brindado siempre. Hemos pasado durante estos últimos años difíciles momentos. Para mí, momentos de los más duros en toda mi vida. Aún con todo, hemos sabido permanecer juntos, formando una maravillosa piña que sigue adelante. Quiero agradecerles a mis hermanas Mireya y Alba y a mi padre Jose todo el cariño, el amor y el apoyo que me han brindado. Os quiero muchísimo.

Aunque no posean lazos de sangre también he estado rodeado de personas que han estado a mi lado, en mi círculo más cercano e íntimo, formando parte de mi mundo. Quiero darle gracias a mi grupito de amigos de Málaga de siempre. Gracias Stefi, María, Gabri, Belén, Mario, por todos estos años y los que vienen, por habernos visto crecer y vivir. Muchas gracias Sylvia por haber compartido conmigo muchos momentos donde hemos reído, disfrutado y encontrado un apoyo maravilloso en el otro. Supongo que tengo que darle gracias al universo por habernos conocido. Muchas gracias a mis amigos de la carrera Ángel, Ángela y MariCarmen, tenemos un grupo apañadísimo y me encanta ver cómo vamos tomando rumbos distintos pero aún así no perdemos esos lazos.

Quiero darle gracias a todos esos compañeros que han estado currando a mi lado, con los que he compartido no solo grandes momentos profesionales sino también ratos divertidos. Nombraré a aquellos que han estado más cerca de mí, pero no me olvido de aquellos con los que he compartido aunque sean momentos más breves. Muchas gracias a la tripulación del Space Robotics Lab, Gonzalo, Nacho, Raúl, Laura, Emilio. También quiero agradecer a Andrés, que fue de los primeros con los que trabajé codo con codo. Muchas gracias a 'los del taller', Juanma, Curro, Pablo. Fuera de España he conocido también a gente maravillosa. Quiero darle gracias a mis compis de la ESA. Muchas gracias Karine y Levin por haber estado a mi lado y haber sido mis amigos más cercanos estando en Holanda. Muchas gracias Martin, ha sido una gran experiencia trabajar contigo y con los rovers de la ESA. También quiero agradecerles a mis compañeros de Luxemburgo del joven grupo SpaceR. En especial muchas gracias Sofía, Pedro, Jota, Abhishek, Dave. Aunque hayamos pasado juntos menos tiempo del que nos hubiera gustado me ha encantado conocerlos y pasar momentos no solo divertidos pero también con mucho significado.

Por último, quiero darle también las gracias a mis directores de tesis. Muchas gracias Alfonso por el apoyo y los consejos que me has brindado durante el doctorado. Muchas gracias por haber estado ahí dando soporte y apoyando el crecimiento de un laboratorio con mucho potencial como es el Space Robotics Lab. Carlos, qué te voy a decir. Muchas, muchas gracias. Me siento tremendamente afortunado de que hayas sido no solo director de mi tesis, sino también un amigo con el que poder contar en los buenos y en los malos momentos. Contigo he aprendido valiosas lecciones no solo de trabajar sino también de vivir, de disfrutar. Podría decir incluso que ha sido gracias a que te he tenido como director de tesis que he podido acabar este doctorado a pesar de todas las dificultades que he ido encontrando a nivel personal.

Espero no haberme dejado a nadie. De todas formas el finalizar esta tesis no es más que un paso más en mi camino. Ahora se abre una nueva etapa en mi vida, y espero de todo corazón que siga rodeado de personas tan maravillosas que hagan que este viaje que es la vida merezca mucho eso, vivirla. Se atisban los primeros rayos de luz del alba. El amanecer de una nueva aventura está ya aquí.

Drawing Robotic Souls.

Ryk.

Sumario

Es esencial aumentar la autonomía de los robots móviles, dotarles de la capacidad de realizar operaciones por sí mismos. Ello es deseable en campos como la exploración planetaria o las operaciones de búsqueda y rescate. Los robots móviles podrían realizar un mayor número de tareas en un mismo intervalo de tiempo sin requerir intervención humana. Estos podrían, por ejemplo, desplazarse largas distancias y llegar a más lugares. Para ello se emplean algoritmos de planificación de caminos. Estos generan un camino que guía al robot hacia la localización objetivo. Sin embargo, los entornos en exteriores y desestructurados presentan un desafío a su locomoción. Por este motivo el planificador de caminos debe tener en cuenta este tipo de entornos a la vez que la capacidad motriz del robot. De esta manera, el algoritmo puede encontrar el camino que conlleve el menor consumo de energía posible. No obstante, no siempre el robot dispone de información precisa y completa acerca del entorno. En estos casos el planificador de caminos debe actualizar de forma dinámica el camino siempre que sea necesario. El robot podría tener que lidiar in situ con elementos del terreno que no fueron considerados previamente.

Esta tesis trata problemas que pueden surgir en la navegación autónoma sobre terrenos irregulares. Como paso previo, presenta un resumen de los algoritmos de planificación de caminos existentes hoy en día. Este resumen se centra en aquellos algoritmos que son compatibles con robots móviles de tierra. Además, sirve para construir una clasificación de los métodos existentes. Esta clasificación se basa en las distintas maneras que tiene un planificador de caminos de modelar y procesar el entorno y la movilidad del robot. Esta tesis pone el foco en aquellos métodos capaces de producir el camino globalmente óptimo dado un mapa de coste. Estos métodos son llamados algoritmos PDE Solving (Solucionadores de ecuaciones en derivadas parciales). La piedra angular de esta tesis es la adaptación de este tipo de algoritmos a la navegación autónoma sobre terrenos irregulares. Primero presenta un planificador de caminos enfocado a robots reconfigurables. Este tipo de robots es capaz de ejecutar varios modos de locomoción. Esta habilidad les permite adaptarse a un mayor número de terrenos, lo cual es tenido en cuenta por el planificador. En segundo lugar, esta tesis presenta una arquitectura de planificación de caminos basada en métodos PDE Solving que sirve no sólo para encontrar un camino óptimo inicial sino también actualizarlo. Esta arquitectura trata con información del entorno proveniente de dos fuentes. Una de ellas es un mapa global inicial que cubre un área extensa con baja resolución. La otra fuente es el robot mismo, el cual detecta obstáculos que puedan aparecer en su camino. Tercero, otra contribución considera el uso de un planificador de caminos en escenarios con superficies inclinadas. Este planificador considera el efecto de la gravedad en el robot y cómo su orientación afecta al coste energético. Finalmente, esta tesis presenta los resultados de varios experimentos de simulación y de campo, los cuales permiten comprobar la funcionalidad de los planificadores descritos y sirven para validarlos.

Abstract

Increasing autonomy on mobile robots is essential. The main reason is to provide them with the ability to perform operations on their own. This skill is desirable in fields such as planetary exploration or search and rescue operations. Mobile robots could carry out more tasks in a time window by not relying on human intervention. These systems could, for instance, drive longer distances and hence reach more places. Autonomous navigation rests on the use of path planning algorithms. These algorithms generate a path that guides the robot towards a target location. Yet, off-road and unstructured environments can pose a challenge to its locomotion capabilities. Thus, the path planner must address them at the same time as the robot mobility skills. In this way, this algorithm finds the path that minimizes a metric such as energy consumption. Not always precise information describing the environments is complete. The path planner must hence dynamically update the generated path whenever necessary. The robot may deal with terrain elements that were not addressed before in situ.

This thesis tackles problems that arise in autonomous navigation on irregular terrains. As a first step, it presents an overview of the existing path planning algorithms. This overview focuses on those algorithms that are compatible with ground mobile robots. It serves to build a classification of the existing approaches. This classification rests on their functionality. In a few words, it tackles the different ways in which path planner model and process the environment and the robot mobility. This thesis puts the focus on some of them that are capable to produce the globally optimal path given a cost map. These are called PDE (Partial Derivative Equation) solving algorithms. How to adapt these algorithms to the autonomous navigation of irregular terrains is the cornerstone of this thesis. It deals first with the use of a path planner along with a reconfigurable robot. This kind of robot is capable to perform many modes of locomotion. This skill allows them to adapt to a larger variety of terrains, and the planner must acknowledge it. Second, this thesis presents a path planning architecture based on PDE solvers that serve to not only find an initial path but also update it. This architecture addresses environment information coming from different sources. One of them is an initial global map that covers a large area but with low resolution. The other source is the robot itself, which detects obstacles placed on its way. Third, another contribution considers the use of a path planner in scenarios with inclined surfaces. Such a planner acknowledges the effect of gravity on the robot and how the orientation affects the cost. Finally, this thesis presents the results of simulation and field experiments. They check the functionality of the described planners and serve to validate them.

"Any planet is *Earth* to those that live on it."

Isaac Asimov, *Pebble in the sky*, 1950

Contents

Declaración de Autoría y Originalidad	iii
Agradecimientos - Acknowledgements	vii
Sumario	ix
Abstract	xi
List of Figures	xxi
List of Tables	xxiv
List of Abbreviations	xxv
1 Introduction	1
1.1 Extreme Path Planning for Exploration Mobile Robots	1
1.2 Contributions	8
1.3 Context and Motivation	10
1.4 Thesis Outline	16
2 State of the Art in Path Planning for Ground Mobile Robots	19
2.1 Introduction	19
2.2 Path Planning Workspace Modelling	20
2.3 General Classification	25
2.4 Reactive Computing Based Path Planning Algorithms	28
2.5 Soft Computing Based Path Planning Algorithms	31
2.6 C-Space Search Based Path Planning Algorithms	34
2.7 Optimal Control Based Path Planning Algorithms	38
2.8 Summary and Conclusions	41
3 Isotropic Optimal Path Planning for Reconfigurable Rovers	45
3.1 Introduction	45
3.2 Reconfigurable Locomotion Cost Model	46
3.3 Optimal Path Planning using the Fast Marching Method	55
3.4 Summary and Conclusions	64

4	Dynamic Multilayered Path Planning	67
4.1	Introduction	67
4.2	Multi-scale Planning Architecture	70
4.3	Multi-layered Grid	73
4.4	Local Path Repairing	76
4.5	Summary and Conclusions	84
5	Anisotropic Cost Model for Traversing Inclined Terrains	87
5.1	Introduction	87
5.2	Problem Formulation	89
5.3	Optimal Path Planning using the Ordered Upwind Method	91
5.4	Anisotropic Cost Function for Planning on Inclined Surfaces	96
5.5	Summary and Conclusions	101
6	Experiments	103
6.1	Introduction	103
6.2	Global Path Planning with Reconfigurable Rovers	104
6.3	Local Path Repairing Tests	111
6.4	Anisotropic Path Planning for Traversing Slopes	119
6.5	Summary and Conclusions	138
7	Conclusions and Future Work	141
7.1	Introduction	141
7.2	Thesis Conclusions	142
7.3	Future Work	145
	Bibliography	149
	Resumen de la Tesis Doctoral	169
	Planificación Extrema de Caminos para Robots Móviles de Exploración . . .	169
	Contribuciones	174
	Contexto y motivación	177
	Estructura de la tesis	181
	Conclusiones de la Tesis Doctoral	185

List of Figures

1.1	Real scale mock-ups of rovers sent by NASA to Mars. From smaller to bigger size: Sojourner (left below), Opportunity (left above) and Curiosity (right). Credit: NASA / JPL-Caltech.	4
1.2	3D reconstruction of the ESA rovers that are planned to be sent in the upcoming years.	5
1.3	SAR robots used in real (a-b) and simulated (c-d) operations.	7
1.4	Diagram showing the contributions of this thesis. It details how these contributions relate to the agent and the environment in the context of the path planning problem.	8
1.5	Facilities where the work of this thesis was held.	11
1.6	Diagram showing the publications that support this thesis, together with those that precede them and those that extend them.	13
2.1	Different types of environment cell decomposition (a-d) and roadmap graphs (e-f).	21
2.2	Examples of ground mobile robots with different kinematic configurations. (a) and (c) were reproduced in (Sánchez-Ibáñez et al., 2021) with permission of the University of Tohoku and the German Research Center for Artificial Intelligence (DFKI) respectively.	23
2.3	Locomotion models used for wheeled ground mobile robots along with path planners: Differential drive (a), Front-Ackermann (b), Skid-Steering (c), Full-Ackermann (d), Crabbing (e) and Point-Turn (f). . . .	24
2.4	Path Planning Roadmap	26
2.5	Graphical representations of concepts used in the Artificial Potential Fields (APF) (a) and Velocity Obstacle (b) algorithms.	29
2.6	Graphical representations of concepts used in Bug algorithm (a) and Elastic Bands (b) algorithm with Bubble bounds.	30
2.7	Examples of Evolutionary algorithms: Genetic (a) and Swarm Optimizer ACO (b).	32
2.8	Main difference between the paths (in red) produced by Edge-restricted (a) and Any-angle algorithms (b): in the first case, waypoints can be placed only on consecutive (neighbouring) nodes.	35
2.9	Overview of many different approaches to Graph Search path planning. The arrows indicate how most of them rest on older approaches yet introduce significant improvements.	36

2.10	Example cases of single-query (a) and multiple-query (b) Sampling Based algorithms. Samples are created in an iterative way until the destination is reached. They can still create more to improve the quality of the path.	38
2.11	PDE Solving based algorithms can calculate a continuous and smooth path in non-uniform cost maps. The cost assigned to each cell can be a scalar value or isotropic (a) or in the form of a vector, i.e. anisotropic (b).	39
3.1	Nodes indexation and neighbourhood in square (a) and hexagonal (b) grids.	47
3.2	3D render image of a rover capable to execute Wheel-walking, with indications of the main parts of its kinematic configuration. Credit: (Pérez-del-Pulgar et al., 2017).	49
3.3	Step-by-step depictions of virtual rover performing the Wheel-walking locomotion using the side-by-side gait.	50
3.4	Schematic of the main forces, torques and position parameters involved in the execution of the Wheel-walking locomotion mode.	51
3.5	Schematic of the main forces, torques and position parameters involved in the execution of the Normal-driving locomotion mode.	54
3.6	An example case in which the FMM is used in a scenario with different kinds of terrain (named 'Compact soil', 'Loose soil' and 'Obstacles' to be representative of a real scenario). The rate at which the wave propagates depends on the assigned values of cost to each terrain. Changing the cost results in changing the wave propagation and, as a consequence, the shape of the optimal path.	57
3.7	Different paths $\Gamma(\tilde{x}_o, \tilde{x}_g, s)$ connecting \tilde{x}_o and \tilde{x}_g , all of them located within the 2D closed region Ω . The path planner must find the discretized version $\tilde{\Gamma}$ that approximates one of them: the optimal.	58
3.8	Graphical step-by-step exposure of the functioning of FMM for path planning.	61
3.9	Virtual scenario with the terrain of the example case.	64
4.1	Representation of the functioning of GESTALT: a set of predefined paths is evaluated and thereafter one of them is selected according to safety conditions. Credits: Bajracharya et al. (2008).	69
4.2	Schematic showing the flow of information in DyMu, the proposed dynamic path planning solution. This architecture rests on the multi-layered grid to represent information at different scales. The path planner makes use of this grid as well as a cost function to generate and update the optimal path. This path serves to guide the rover in its autonomous drive.	70

4.3	Illustrative image of the multi-layered grid. Each global node indicates the type of terrain that is contained in its area and occupies the same area as a finite number of local nodes, each of them providing an indication of its state relative to nearby obstacles.	74
4.4	The GPP generates a global path (a)(c). Then, whenever an obstacle is detected in the way of the rover, it computes an alternate path, either completely new (b) or just partially, reconnecting with the original one at some point (d).	76
4.5	The LPR process is composed by several operations. First, when obstacles are in the way of the rover the process is triggered (a). Local nodes are created (b) and the values of r_{ab}^{ij} are calculated (c). Finally, the FM* uses these values to get a new path (d).	77
4.6	Admissible error due to path discretization after the GPP. In this extreme case the path does not trigger the LPR, although the segment connecting two waypoints traverses the risky area.	79
5.1	Different paths $\Gamma = \Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$ connecting \tilde{x}_o and \tilde{x}_g can be defined upon different tangent directions $\vec{u}(\Gamma)$. The role of the anisotropic path planner is to find the optimal path among them.	90
5.2	Schematic of the functioning of biOUM. Two expanding waves start from the goal and the origin nodes (\tilde{x}_g and \tilde{x}_o) respectively. The loop controlling each wave is colored in purple and orange. The red arrows indicate the characteristic direction $\vec{u}(\tilde{x}_{ij})$ that passes through each node.	92
5.3	Update process of a Considered node \tilde{x}_{ij} . Its values of total cost and characteristic direction are computed taking into account Accepted-Front nodes within the distance $\zeta(\tilde{x}_{ij})$ expressed in Equation (5.7).	94
5.4	Graphical representation of the slope model used to build up the anisotropic cost function. This model encompasses multiple variables such as the slope gradient α_{ij} , the aspect direction $\vec{\gamma}_{ij}$, the normal vector \vec{v}_{ij} , the heading direction $\vec{\psi}$ and the relative angle $\beta(\vec{\psi}, \vec{\gamma}_{ij})$. All of them are marked in this conceptual depiction.	96
5.5	Elliptical inverse of the anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$. The descent direction is the one coincident with the slope aspect $\vec{\gamma}_{ij}$. The robot has a heading direction $\vec{\psi}$ that determines which value of this anisotropic cost is returned.	97
5.6	Use of Bezier curve function $R_b(\tilde{x}_{ij})$ to comply with the non-zero positive cost specification in a smooth way. $\rho_{ij} = 0.3$, $\alpha_\Delta = 15^\circ$ and $\alpha_{ij}^{zero} = \arctan \rho_{ij}$	100
6.1	ExoMars Testing Rover (ExoTeR), a prototype that mimics the locomotion subsystem of ExoMars <i>Rosalind Franklin</i> rover.	105

6.2	Estimation of the energy per meter cost consumed by the ExoTeR rover for each locomotion mode according to the values of slip ratio σ_{ij} and the specific resistance ρ_{ij} . This model does not address the effect of the slip ratio on the wheel-walking locomotion. Here it is considered the velocity v_{ij} as 0.02 m s^{-1}	106
6.3	Calculated values of total cost $T(\tilde{x}_o, \tilde{x}_{ij})$ and obtained paths when using only Normal-driving or $L = \{nd\}$ (a) and both Normal-driving and Wheel-walking or $L = \{nd, ww\}$ (b).	108
6.4	Overview of virtual models describing the shape of the experimental terrain located near ESA facilities.	110
6.5	The distribution of the different terrains is shown in (a). The FMM is executed twice: one considering just the Normal-driving mode (c), $L = \{nd\}$, and other also taking the Wheel-walking mode into account (d), $L = \{nd, ww\}$. In this way, the total cost to arrive at the destination is reduced for some areas (b).	111
6.6	Resulting Paths from the LPR simulation test using different values of λ	112
6.7	Results from the LPR simulation test. The shape of the repaired paths after executing the LPR process 4 times is shown (a), as well as information relative to the computational power used, in the form of the number of processed nodes (b), the number of times the eikonal equation is used in total and the elapsed time to do the computation.	113
6.8	The Heavy Duty Planetary Rover (HDPR), shown in (a), performing its traverse through the experimental field populated by obstacles shown in (b).	114
6.9	Paths computed during the field test.	115
6.10	More traverses performed by HDPR during field tests.	117
6.11	Showcase of the computation times that were taken by the HDPR on-board computer to execute the Local Path Repairing. Figures (a), (b), (c), and (d) depict the values for all path updates along the respective path. In (e), the computed mean and standard deviation of each traverse is shown, together with the mean and standard deviation of the computation times for all traverses.	118
6.12	Preparation of the map used for the simulation tests with the anisotropic planner. It is based on the shape of a real crater on the Martian surface. The resolution of the DEM is 1 m.	120
6.13	Slip ratio function of the two models (Wheel and Track) found in the literature (Sutoh et al., 2015), used in the numerical simulation tests.	122
6.14	Values of cost returned by each anisotropic cost function.	123

6.15	Results from the first test using anisotropic and isotropic cost functions. The resulting paths connecting two locations, \tilde{x}_o to \tilde{x}_g , are depicted (a). The origin is located at (10 10) m while the goal is at (55 50) m. Note that the 3d view is rotated to provide a better perspective of the obtained paths. The total cost calculated by bi-OUM (a) and bi-FMM (b).	125
6.16	Comparison between the use of bi-OUM and bi-FMM to plan paths on inclined surfaces. The number of times the total cost is updated serves to check the computational load of both approaches (a). The omission of the information in the descent and lateral directions of the isotropic cost function used by bi-FMM is relevant or not according to the terramechanic parameters (b).	128
6.17	Showcase of pictures showing the terrain using different perspectives (a)(b)(c) and a screenshot of Pix4Dmapper showing its virtual reconstruction (d).	129
6.18	The skid-steering four-wheeled Cuádriga robot.	130
6.19	Extraction of ρ_{ij} and σ_{ij} based on preliminary drives of Cuádriga. The plots (b) and (c) are accompanied by histograms that indicate the density of samples along the slip ratio (vertical) and pitch (horizontal) axes. The pitch Θ is negative when the robot is ascending.	131
6.20	Slope data describing the shape of the experimental terrain after smoothing the DEM with an average filter.	133
6.21	Paths traversing the slope with certain configurations of weight values. Here $w_\phi = w_{ij}^\Phi$ and the positions of interest are grid nodes, i.e. $x_a = \tilde{x}_a$, $x_b = \tilde{x}_b$, $x_c = \tilde{x}_c$ and $x_d = \tilde{x}_d$	133
6.22	Roll penalization functions used to minimize the roll angle.	134
6.23	Orientation angles produced by each round-trip travel from the three models.	134
6.24	Traversed distance with an absolute value of roll Φ higher than the threshold indicated in the horizontal axis.	135
6.25	Paths resulting from the second test with anisotropic cost and the placement of the position samples of Cuádriga, measured with a RTK GNSS antenna.	136
6.26	Comparative analysis of the orientation angles and the cost between planning estimations and Cuádriga experience.	136

List of Tables

1.1	ECSS levels of autonomy	5
2.1	Main surveys and reviews of global path planning algorithms found in the literature.	27
2.2	Intelligent models used for some Swarm Optimizer algorithms, together with some of the behaviours they perform inspired by nature. .	33
2.3	Overview of characteristics of the presented path planning algorithms.	42
3.1	Optimization criteria for optimal isotropic path planning according to how the cost function is defined. Here T is the total cost.	59
6.1	Summary of experiments presented in this thesis.	104
6.2	Simulation values of isotropic tests	106
6.3	Terrain parameters used in the first two simulation tests, where $P_{ij} = P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$	107
6.4	Results from the execution of the GPP to produce the path that is later repaired in Figure 6.9b.	116
6.5	Results from the execution of the GPP to produce the path that is later repaired.	116
6.6	Results of the first anisotropic test with the Wheel model. Each algorithm is executed using its bi-directional version. The first row is the number of times a total cost is updated for a node. The second row is the total cost estimated by the planner. The third row is the corresponding total cost after integrating the anisotropic cost function along the resulting path, even for those paths obtained using bi-FMM. The fourth row is the error committed in the estimation with respect to the result from the integration.	126
6.7	Results of the first anisotropic test with the Track model. Each algorithm is executed using its bi-directional version. The first row is the number of times a total cost is updated for a node. The second row is the total cost estimated by the planner. The third row is the corresponding total cost after integrating the anisotropic cost function along the resulting path, even for those paths obtained using bi-FMM. The fourth row is the error committed in the estimation with respect to the result from the integration.	126
6.8	Specifications of the Cuádriga robot	130

6.9	Mean Absolute Error (MAE) of experimental data results with respect to planned data	138
6.10	Root Mean Squared Error (RMSE) of experimental data results with respect to planned data	138

List of Abbreviations

ABIT*	Advanced Batch Informed Trees. 37
ACO	Ant Colony Optimizer. xvii, 21, 32
ADE	Autonomous Decision making for very long traverses. 12, 14, 16, 178, 179, 181
APF	Artificial Potential Fields. xvii, 27–29
ARES	Autonomous Routing on Extreme Surfaces. 12–14, 178, 179
ASTRA	Advanced Space Technologies in Robotics and Automation. 14, 15, 180, 181
BIT*	Batch Informed Trees. 37
CAMIS	Continuous Anisotropic Model for Inclined Surfaces. 88, 89, 98–102, 104, 119–122, 130, 132, 135, 137, 139, 144–147, 179, 187, 188
D*	Dynamic A*. 35, 36
DEM	Digital Elevation Map. xx, 22, 33, 109, 120, 121, 128, 132
DFKI	German Research Center of Artificial Intelligence. 16
DP	Differential Dynamic Programming. 41
DPP	Dynamic Programming Principle. 39, 56, 58, 91
DSN	Deep Space Network. 3, 171
DWA	Dynamic Window Approach. 25, 26, 29
DyMu	Dynamic Multilayered path planner. xviii, 70, 84, 109, 114, 116, 143–146, 186–188
ECSS	European Cooperation for Space Standardization. 6, 173
ESA	European Space Agency. xx, 5, 6, 11, 12, 15, 110, 143, 173, 177, 178, 186
ESTEC	European Space Research and Technology Center. 11
ExoTeR	Exomars Testing Rover. 106, 107, 145
FIM	Fast Iterative Method. 41

FIRST-ROB	Multi-Robot System for Cooperation with First Response Human and Canine Rescue Teams in Catastrophe Scenarios. 11, 14, 177, 179
FM*	Heuristic Fast Marching. xix, 43, 69, 72, 76–78, 80, 81, 83, 84
FMM	Fast Marching Method. xviii, xx, xxi, xxiii, 27, 38, 40, 43, 46, 55–57, 59–61, 63–65, 69, 72, 73, 76, 78–81, 84, 88, 91, 95, 97, 105, 111, 119, 121, 122, 124–128, 138, 143, 144, 147, 187
FMT*	Fast Marching Tree. 38
FSM	Fast Sweeping Method. 40, 41, 65, 146
GPP	Global Path Planning. xix, xxiii, 72, 75, 76, 79, 80, 82, 84, 85, 109, 113–116, 146
HDPR	Heavy Duty Planetary Rover. xx, 103, 114–118, 143, 144, 186, 187
HiRISE	High Resolution Imaging Science Experiment. 109, 121
HJB	Hamilton-Jacobi-Bellman. 39, 40, 90, 94, 101, 147
ISA	Department of Systems Engineering and Automation. 10
iSAIRAS	International Symposium on Artificial Intelligence, Robotics and Automation in Space. 14, 15, 180, 181
LPA*	Lifelong Planning A*. 36
LPR	Local Path Repairing. xix, xx, xxxi, 69, 72, 76–79, 84, 85, 103, 104, 109, 111–114, 116, 119, 139, 143, 146, 186
LQR	Linear Quadratic Regulator. 41
MER	Mars Exploration Rover. 4, 172
MILP	Mixed-Integer Linear Programming. 41
MPC	Model Predictive Control. 38, 41
MSL	Mars Science Laboratory. 5, 172
MSR	Mars Sample Return. 5, 173
NASA	National Aeronautics and Space Administration. xvii, 4, 6, 68, 172, 173
OUM	Ordered Upwind Method. xxi, 40, 43, 85, 88, 89, 91, 92, 94–97, 100–102, 119, 122, 124–128, 144, 146, 147, 187

PDE	Partial Derivative Equation. 9, 10, 16, 17, 39, 43, 46, 55, 64, 65, 68, 69, 72, 84, 88, 89, 91, 101, 105, 138, 142–144, 147
PRM	Probabilistic Roadmap Method. 38, 41
PSO	Particle Swarm Optimizer. 32
PUTB	Planetary Utilisation Testbed. 64
RAP	Red Andaluza de Posicionamiento. 145, 188
RDT	Rapidly Deterministic Tree. 37
RL	Reinforcement Learning. 34
RRT	Rapidly Random Tree. 37, 38, 41
RRT*	Heuristic Rapidly Random Tree. 37, 88
SAR	Search And Rescue. xvii, 2, 7, 8, 10, 19, 87, 141, 142
SBMPO	Sampling Based Model Predictive Optimization. 88
SFR	Sample Fetch Rover. 5, 6, 15, 173, 181
SRL	Space Robotics Laboratory. 10, 11
TEB	Timed Elastic Bands. 30, 31
THEMIS	Thermal Emission Imaging System. 109
TRUST-ROB	Towards Resilient UGV and UAV Manipulator Teams for Robotic Search and Rescue Tasks. 11, 14, 177, 180
UMA	University of Malaga. 10–12
VFH	Vector Field Histogram. 28, 29

List of Symbols

B_m	Motor rotational damper	$\text{N} \cdot \text{m} \cdot \text{s} \cdot / \text{rad}$
$C(\tilde{x}_{ij})$	Cost Function at \tilde{x}_{ij}	$\text{W} \cdot \text{s}/\text{m}, \text{A} \cdot \text{s}/\text{m}, \text{s m}^{-1}$
$c(\tilde{x}_{ab}^{ij})$	Cost Function at \tilde{x}_{ab}^{ij}	s m^{-1}
d_l	Body-wheel bar lenght	m
d_r	Wheel radius	m
d_{step}	Gradient descent method step	m
d_{risk}	Risky area maximum distance from obstacles	m
F_{dpbk}	Drawbar Pull Force on walking joint k	N
F_{dpwk}	Drawbar Pull Force on wheel joint k	N
F_N	Normal Force	N
F_{Nwk}	Normal Force on wheel k	N
F_r	Resistive Force opposed to motion	N
g	Gravity acceleration	m s^{-2}
h_{ab}^{ij}	Heuristic function at node \tilde{x}_{ab}^{ij}	
I	Electric charge function	$\text{A} \cdot \text{s}$
I_d	Normal-driving electric charge function	$\text{A} \cdot \text{s}$
I_{ww}	Wheel-walking electric charge function	$\text{A} \cdot \text{s}$
i	Horizontal index to access a node	
J_b	Body inertia in Motor axis	kg m^2
J_w	Wheel inertia in Motor axis	kg m^2
j	Vertical index to access a node	
K_b	Walking motor constant	$\text{N} \cdot \text{m}/\text{A}$
K_w	Wheel motor constant	$\text{N} \cdot \text{m}/\text{A}$
k	Counter index for multiple purposes	
L	Set of available locomotion modes	
l	A locomotion mode	
m_b	Robot body mass	kg
m_w	Robot wheel mass	kg
nd	Normal-driving locomotion mode	
$neigh(\tilde{x}_{ij})$	Neighbourhood of (global) node \tilde{x}_{ij}	
$neigh(\tilde{x}_{ab}^{ij})$	Neighbourhood of local node \tilde{x}_{ab}^{ij}	
n	Number of wheels/legs	
n_{nodes}	Number of nodes in the grid $\tilde{\Omega}$	
P	Instantaneous power function	$\text{W (J s}^{-1}\text{)}$
P_d	Normal Driving instantaneous power function	$\text{W (J s}^{-1}\text{)}$

P_{ww}	Wheel-Walking instantaneous power function	$W \text{ (J s}^{-1}\text{)}$
p_{bx}	X-component of robot body position	m
\dot{p}_{bx}	X-component of robot body speed	m s^{-1}
\ddot{p}_{bx}	X-component of robot body acceleration	m s^{-2}
p_{bz}	Z-component of robot body position	m
\dot{p}_{bz}	Z-component of robot body speed	m s^{-1}
\ddot{p}_{bz}	Z-component of robot body acceleration	m s^{-2}
p_{wkx}	X-component of wheel k position	m
\dot{p}_{wkx}	X-component of wheel k speed	m s^{-1}
\ddot{p}_{wkx}	X-component of wheel k acceleration	m s^{-2}
$p_{w kz}$	Z-component of wheel k position	m
$\dot{p}_{w kz}$	Z-component of wheel k speed	m s^{-1}
$\ddot{p}_{w kz}$	Z-component of wheel k acceleration	m s^{-2}
$Q(\tilde{x}_{ij}, \vec{\psi})$	Anisotropic cost function	$W \cdot \text{s/m}, A \cdot \text{s/m}$
$R_b(\tilde{x}_{ij})$	Bezier curve function	$W \cdot \text{s/m}, A \cdot \text{s/m}$
r_{ij}	Collision risk of global node \tilde{x}_{ij}	
r_{ab}^{ij}	Collision risk of local node \tilde{x}_{ab}^{ij}	
S_{ij}	State of planning calculation on node \tilde{x}_{ij}	
S_{ab}^{ij}	State on local node \tilde{x}_{ab}^{ij}	
S_{ij}^o	State on node \tilde{x}_{ij} with respect to origin	
S_{ij}^g	State on node \tilde{x}_{ij} with respect to goal	
s	Path length	m
$T(\tilde{x}_o, \tilde{x}_{ij})$	Total cost at \tilde{x}_{ij} from \tilde{x}_o	$W \cdot \text{s}, A \cdot \text{s}, \text{s}$
$T(\tilde{x}_{ij}, \tilde{x}_g)$	Total cost at \tilde{x}_{ij} remaining to \tilde{x}_g	$W \cdot \text{s}, A \cdot \text{s}, \text{s}$
$t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij})$	Local total cost of local node \tilde{x}_{ab}^{ij}	s
$\vec{u}(\tilde{x}_{ij})$	Characteristic direction at \tilde{x}_{ij}	
V	Motor Voltage Supply	V
v_c	Commanded robot speed	m s^{-1}
v_{ij}	Real robot speed at node \tilde{x}_{ij}	m s^{-1}
ww	Wheel-walking locomotion mode	
w_{ij}^Φ	Roll weight function of node \tilde{x}_{ij}	
\tilde{x}_{ab}^{ij}	2D Position of (local) grid node	$\text{m} \times \text{m}$
\tilde{x}_g	2D Position of goal	$\text{m} \times \text{m}$
\tilde{x}_{ij}	2D Position of (global) grid node	$\text{m} \times \text{m}$
\tilde{x}_o	2D Position of origin	$\text{m} \times \text{m}$
α_{ij}	Slope gradient at node \tilde{x}_{ij}	rad, °
ϵ	Coefficient to iteratively calculate the total cost	
$\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$	Continuous path curve	
$\Gamma(\tilde{x}_o, \tilde{x}_g, s)$	Position of path at length s	$\text{m} \times \text{m}$
$\tilde{\Gamma}$	Discretized path, set of waypoints	
$\tilde{\Gamma}_k$	Waypoint k in $\tilde{\Gamma}$	$\text{m} \times \text{m}$

$\tilde{\Gamma}_{reference}$	Reference waypoint used by LPR	$m \times m$
$\tilde{\Gamma}_{start}$	Starting waypoint of LPR	$m \times m$
$\tilde{\Gamma}_{triggerer}$	Waypoint that triggers the LPR	$m \times m$
$\vec{\gamma}_{ij}$	Aspect direction of a slope at node \tilde{x}_{ij}	
Λ	Global Grid Resolution	m
λ	Local Grid Resolution	m
\vec{v}_{ij}	Normal vector of a slope at node \tilde{x}_{ij}	
Φ	Roll orientation angle	$rad, ^\circ$
ρ_{ij}	Specific resistance at node \tilde{x}_{ij}	
σ_{ij}	Slip ratio at node \tilde{x}_{ij}	
τ_{bk}	Torque of walking joint motor k	$N \cdot m$
τ_{wk}	Torque of wheel joint motor k	$N \cdot m$
Θ	Pitch Orientation angle	rad
θ_{bk}	Rotation angle of wheeled leg bar k	rad
$\dot{\theta}_{bk}$	Rotation speed of wheeled leg bar k	$rad s^{-1}$
$\ddot{\theta}_{bk}$	Rotation acceleration of wheeled leg bar k	$rad s^{-2}$
θ_{wk}	Rotation angle of Wheel k	rad
$\dot{\theta}_{wk}$	Rotation speed of Wheel k	$rad s^{-1}$
$\ddot{\theta}_{wk}$	Rotation acceleration of Wheel k	$rad s^{-2}$
Ω	2D region of interest for path planner	
$\tilde{\Omega}$	Grid discretization of Ω	
$\vec{\psi}$	Heading direction of the robot	
$Y(\tilde{x}_{ij})$	Anisotropy at node \tilde{x}_{ij}	
$\zeta(\tilde{x}_{ij})$	OUM search radius centered at node \tilde{x}_{ij}	m

Chapter 1

Introduction

"Exploration is in our nature. We began as wanderers, and we are wanderers still. We have lingered long enough on the shores of the cosmic ocean. We are ready at last to set sail for the stars."

Carl Sagan
Cosmos, 1980

1.1 Extreme Path Planning for Exploration Mobile Robots

Since ancient times humanity has invested numerous efforts to expand the limits of its knowledge. Humans have explored more and more locations around the globe, trying to understand the ins and outs of the world that surround us. They have settled in some of these new places, forming colonies. In the current situation, with a globalized world and human populations being placed almost everywhere, the next frontier is the deep space. We dream of colonizing other worlds like Mars. Nevertheless, to make this aspiration become reality several technological, ethical and legal issues must be solved first (Levchenko et al., 2019). Meanwhile, space agencies have sent in the last years multiple kinds of spacecraft to explore extraterrestrial environments. These scenarios are unstructured, irregular and the information describing them is limited. Unlike flat, even surfaces, the terrain in these places can be difficult to traverse due to its composition and/or shape. The use of advanced technologies to tackle the existing inconvenient conditions is mandatory to preserve the safety of robotic explorers and promote the efficiency of their operations. In other words, these robots must navigate making use of extreme autonomous navigation capabilities. These technologies can be benefited from similar applications on Earth, and vice-versa. For instance, in disaster scenarios, the role of exploration agents is important to raise awareness of the situation. This knowledge may prove helpful in the search of possible victims, as it is considered when planning the course of action of first responders (Seppänen et al., 2015).

Exploration, reconnaissance, inspection. All these words refer to the act of adventuring into partially or fully unknown environments and reducing uncertainty about something. This action may return many benefits as result. The newly acquired information could increase the potential to plan any further action. We exploit this fact from the moment we are born: as we grow we adventure, we learn new information and, as a result, our possibilities to survive and interact with the environment increase enormously. Moreover, the witnessing of phenomena that are not comprehended by us as infants motivates us to seek a better understanding of them (Köster et al., 2020). One important remark here is that exploration entails actively acting. This can be in the form of motion, i.e., displacing from one spatial state to another. As a consequence of performing this motion, the individual has a new viewpoint. From this viewpoint, such individual changes his/her perspective, and may perceive things better. The novel incoming information improves our understanding and awareness of the situation. This helps in making more options to become available to act next.

Humans are not longer alone to explore new places. In the last decades more and more autonomous robotic systems are being put on the scene to perform this kind of task. There are many reasons to opt for this. One of the most important ones is to avoid risking human lives in certain dangerous scenarios. The atmosphere (or the lack of it), the radiation, the fragility of the terrain, ... many are the factors that justify the substitution of humans by robots to operate. The design and structure of these systems differ according to the medium in which they are meant to move: sea, air, space, ground. The latter is where the focus of this thesis is on. This thesis presents the results from the work carried out to improve the autonomous navigation capabilities of ground mobile robots. In particular, it presents the work done on those algorithms and techniques that determine the way the vehicle decides where to drive and how. As a matter of better delimiting the scope of this thesis, the term *ground* refers to those robots who can propel themselves by interacting with a surface, e.g. using wheels. An important clarification is made here: it is assumed the action of gravity makes these vehicles be in permanent contact with the surface. This statement entails that this thesis does not consider any manoeuvre that implies the separation of the vehicle from the ground. An example of this is the action of jumping that hopping robots can make. In this way, this thesis has a special focus on wheeled robots designed for exploration purposes. This is the case of rovers: robotic vehicles used to explore the surface of other planets and satellites beyond Earth. These systems must navigate through areas where human presence is still unfeasible. Besides, in a similar way to wheeled robots used for Search And Rescue (SAR) purposes, autonomously addressing the particularities of an unstructured and complex terrain is desired to come up with a proper course of action minimizing or even dismissing human intervention.

In many navigation schemes, one of the most important parts is the generation of

a path. This path is meant to be followed by the vehicle and connects its starting position (the origin) with the desired destination (the goal). Also known as *Guidance*, this thesis refers to this path generation as *path planning*. Furthermore, the scope of application of this path planning includes off-road, unstructured scenarios. This kind of location may pose a challenge for the navigation system of any mobile robot. This system must be effective, taking the robot safely to another place, and efficient, minimizing the invested resources (e.g. energy). However, the irregularities in the terrain may in some cases penalize the robot motion. For this reason, the path planner must account for this given the locomotion capabilities of the robot. Moreover, the *extreme* nature of these scenarios may also limit the perception capabilities of not only the robot itself but also any external medium such as drones or satellites. This is translated into the limited availability of the information describing the scenario. The planner must hence account for this and dynamically update the plan (or *replan*) whenever necessary according to the updates in this information. Next, brief discussions about two fields of application with extreme conditions are provided. These are extraterrestrial planetary environments and disaster scenarios on Earth. In the first case, rovers must navigate through partially known scenarios with rough terrain. In the second case, robots may have to traverse off-road scenarios in situations where minimizing time and/or energy is critical.

Planetary exploration

Hints about the origin of life and the universe may be hidden on the surface of other celestial bodies beyond Earth. The study of their geological history and the materials located there could help us to understand better some of the processes that occur on Earth (T. Zhang et al., 2019). Furthermore, there exist valuable resources that may be exploited by the industry and space agencies in future space missions. These resources, such as water on the Moon, would hopefully help in the creation of human colonies in extraterrestrial places. Nevertheless, to go and inspect these places requires the use of exploration agents. Nowadays there are plenty of difficulties to safely transport humans during space travels. For this reason, their prolonged presence in extraterrestrial environments is still unfeasible. Therefore, the use of robotic agents (rovers), cheaper to maintain and not living beings, arises as a good and more realistic alternative in the meantime. They can accomplish the role of carrying and operating scientific instruments in planetary and space scenarios (Gao et al., 2017).

Although the use of robots for extraterrestrial exploration purposes is justified, there are still many underlying issues. One of them is related to the communications with the ground stations located on Earth. The delay in sending and receiving messages with robotic systems on the Moon can approximately take from 3 to 10 seconds. This allows the use of direct teleoperation at a certain level (Y. Wang, 2021). However, the communications using the Deep Space Network (DSN) between the ground stations on Earth and the rovers on Mars can take much longer: up to 40 minutes approximately for a two-way message (Lester et al., 2017). This is a huge delay

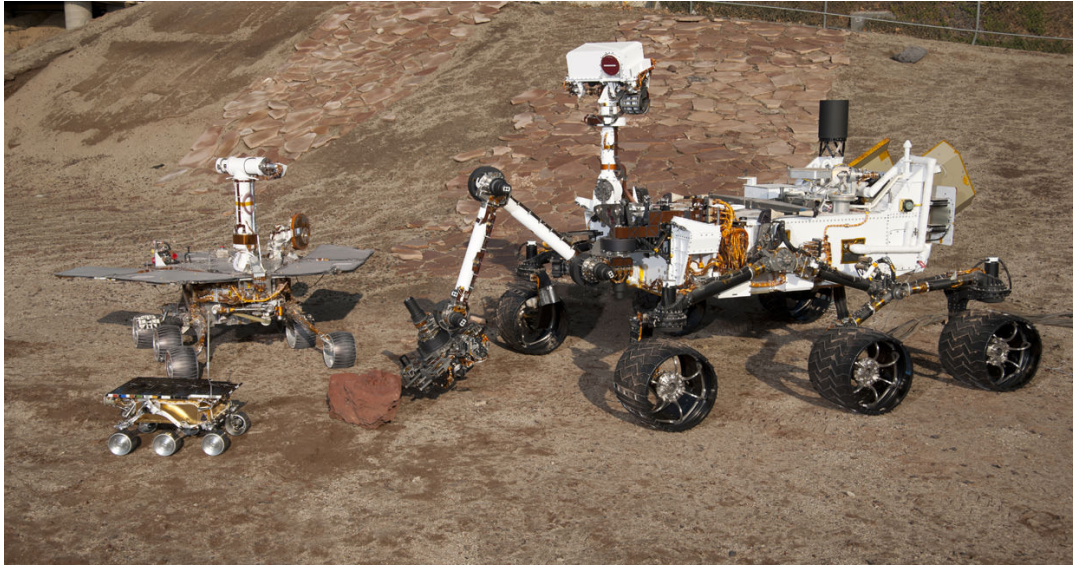
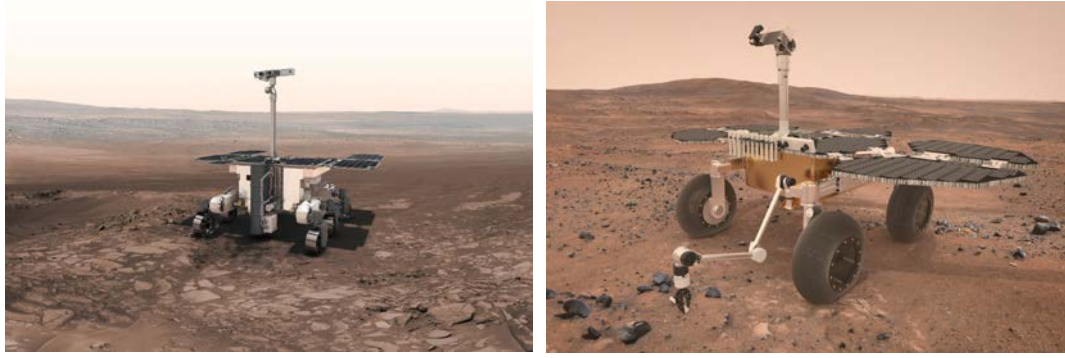


FIGURE 1.1: Real scale mock-ups of rovers sent by NASA to Mars. From smaller to bigger size: Sojourner (left below), Opportunity (left above) and Curiosity (right). Credit: NASA / JPL-Caltech.

that makes unfeasible the direct control of these robots: the ground operator must wait for the next incoming message, analyze the arriving information and thereafter send the next command or series of commands back to Mars (Bresina et al., 2005). On top of that, communications between Mars and Earth are limited by the passing of the Mars Relay Network orbiters. This means that these communications can only happen a few times per *sol* (martian day, similar to a day on Earth). As a result, the distance that rovers drive is restricted to a limited amount of meters.

With regards to the rovers that were sent in the last missions to Mars, the autonomy software installed on them has been progressively increased in the last decades. This was made to compensate for the mentioned problems in the communications between this planet and Earth. Figure 1.1 shows the real scale replicas of some of these rovers. In 1997, the National Aeronautics and Space Administration (NASA) sent to Mars the first rover capable of moving with certain autonomy: Sojourner (Bajracharya et al., 2008). This relatively small robot was capable of autonomously performing simple operations like going to a certain waypoint (Mishkin et al., 1998). It achieved a total distance of around 100 meters, serving as a proof of concept for sending more robotic mobile explorers to Mars. The immediate next mission was the Mars Exploration Rover (MER), in which two twin rovers were sent to different spots of the red planet. One of them was Spirit (Arvidson, Bell, et al., 2010) and the other one was named Opportunity (Arvidson, Ashley, et al., 2011). For this occasion NASA had augmented the autonomous capabilities of both robots, providing them with a local path planner called GESTALT (Biesiadecki et al., 2006). During their mission, it was figured out that a local planner was not enough for achieving complete autonomous navigation: the rover could get stuck in different situations involving encounters with rocks. Because of this, a global path planner was also included in



(a) The Rosalind Franklin rover. Credit: ESA.

(b) The Sample Fetch Rover (SFR). Credit: ESA.

FIGURE 1.2: 3D reconstruction of the ESA rovers that are planned to be sent in the upcoming years.

the autonomy software. In this way, a long-traverse could be initially planned while the local planner would dynamically produce safe paths avoiding obstacles (Maimone et al., 2007). This autonomy system was kept for the Mars Science Laboratory (MSL) Curiosity rover in 2011 (Lele, 2014). Nevertheless, manual commanding from the ground operators is still the most common strategy used to navigate on Mars. Usually, for each martian day (or *sol*) a series of tasks is previously planned and later sent to these vehicles.

Future missions include sending to the red planet the Rosalind Franklin rover (see Figure 1.2a) as part of the ExoMars campaign, leaded by the European Space Agency (ESA). It was initially planned for 2020 (Vago et al., 2017) but delayed to 2022 at the time this thesis is written. Its main purpose is to take and analyze samples from underground to find any signs of current or past life on the red planet. One of the main particularities of this rover is the disposition of extra joints on top of each of its legs. They allow the rover to deploy and retrieve each leg in a similar way as if it was *walking*. Thus, this mode is referred to as Wheel-walking (Patel et al., 2010; Woods et al., 2009). It makes the rover increase its traction on sandy soils as demonstrated in previous experiments (Azkarate, Zwick, et al., 2015). In this way, the rover could avoid getting stuck as happened to the Spirit rover in 2009 (Arvidson, Bell, et al., 2010). Another future mission is the Mars Sample Return (MSR) mission (Muirhead et al., 2019). It consists of taking several tubes with samples inside from the Martian surface and return them to Earth. These tubes are expected to be previously

TABLE 1.1: ECSS levels of autonomy

Level	Description
E1	Direct commanding from ground operators
E2	Execution of pre-planned operations
E3	Reactivity, the vehicle adapts to external events
E4	Goal-oriented, fully autonomous navigation

left by the Perseverance NASA rover, which arrived at Mars in February 2021. Later on, another rover that will be built by ESA, the Sample Fetch Rover (SFR) (see Figure 1.2b), will be tasked to retrieve them. For this kind of missions, efficiency is denoted by the number of samples that can be successfully retrieved within a time window, so higher capabilities in both mobility and autonomy are desirable. As a side note, a better understanding of the levels of autonomy of a rover can be done thanks to Table 1.1. It contains a division into four levels proposed by the European Cooperation for Space Standardization. Reaching the last one, E4, will probably be crucial for complying with the expectations imposed on the SFR, as well as any other future rover with increasingly complex tasks. An example of the latter is the mobile robotic systems that will intervene in the construction of infrastructure on the Moon surface (Govindaraj et al., 2019).

To summarize, increasing the autonomy of rovers is justified by two reasons. First, as mentioned, direct teleoperation of these systems is difficult or even not feasible and any other kind of commanding strategy will delay any operation, compromising the efficiency of the mission as a consequence. Second, the complexity of the tasks that are foreseen to be performed by future rovers is increasing. SFR, for instance, shall fetch samples, pick them with a manipulator, store them and later place them on another spacecraft, all of this within a limited time window restriction.

Search And Rescue (SAR) Robotics

According to previous research, many actions can influence how a disaster is managed: mitigation, preparedness, response and recovery (Jorge et al., 2019). The third one, response, corresponds to those tasks carried out just after the disaster happens. Since the tragedy of the World Trade Center, which collapsed in 2001, there have been tens of cases in which the use of robotic mobile platforms has proven beneficial (Murphy, 2004). They have assisted in both natural (e.g. earthquakes) and man-made (e.g. mines) disaster scenarios. In these cases, a robot or team of robots must perform exploration operations to, for example, look for potential victims or inspect the state of any damaged structure (Murphy et al., 2016). For instance, in 2013 a tsunami hit the east coast of the main island of Japan and had as consequence serious damage to some of the buildings of a nuclear power plant in Fukushima, causing a meltdown as result. Mobile robots, depicted in Figure 1.3a, were sent as part of the emergency response to this disaster, being teleoperated to reach areas contaminated with radiation and thereby hostile to humans (Nagatani et al., 2013). Another example is the use of the robot shown in Figure 1.3b to inspect a pair of churches that were damaged after an earthquake that occurred in Italy in 2016 (Kruijff-Korbayová et al., 2016). The main reason to use robots in this case was the elevated risk of both churches collapsing, which would put human operators inside any of the buildings in a very dangerous situation. A distinct time-critical rescue operation consist of assisting people that are trapped in mountains due to accidents or even external actors such as an avalanche (Silvagni et al., 2017).



(a) *Quince* robots, used during the response to the nuclear accident in Fukushima. Credits: (Nagatani et al., 2013).



(b) Robot used after the 2016 earthquake in central Italy. Credits: *TRADR* project.



(c) Legged robot *ANYMAL*. Credits: (Hutter et al., 2017).



(d) *RAMBLER* assisting a dummy device that emulates a victim. Credits: *FIRST-ROB* project.

FIGURE 1.3: SAR robots used in real (a-b) and simulated (c-d) operations.

Although teleoperating these systems is demonstrated to be safe and reliable, it requires the presence of an already limited number of human operators in the loop (Birk et al., 2006). Moreover, wireless communication can be compromised in scenarios such as mines or nuclear plants (Murphy, 2012), motivating the increase of the autonomous capabilities of rescue robots (Bogue, 2019). In this way, a single operator could command a higher number of robots thanks to the low-level navigation tasks being automated. Another difficulty in directly tele-operating rescue mobile robots is properly addressing the motion of robots with a high number of DoF. Interaction with an irregular terrain is not trivial as preserving the stability of the system or avoiding collisions are difficult tasks for the user and demand complex interfaces such as exoskeletons (Klamt et al., 2018). Unstructured environments may present extreme difficulties in the form of rocks, slopes or terrains with different terramechanic properties. This arises the need of developing path planning solutions that acknowledge these features and let exploration vehicles find safe and optimal trajectories (Delmerico et al., 2019). For these applications, legged robots, as the one portrayed in Figure 1.3c, are promising due to their adaptability to multiple kinds of uneven terrain (Miki et al., 2022). Moreover, providing robots with autonomous first response capabilities to quickly assist victims is also of interest (see Figure 1.3d)¹.

¹https://www.uma.es/robotics-and-mechatronics/info/107721/FIRST-ROB/?set_language=en Accessed on 4th February 2022

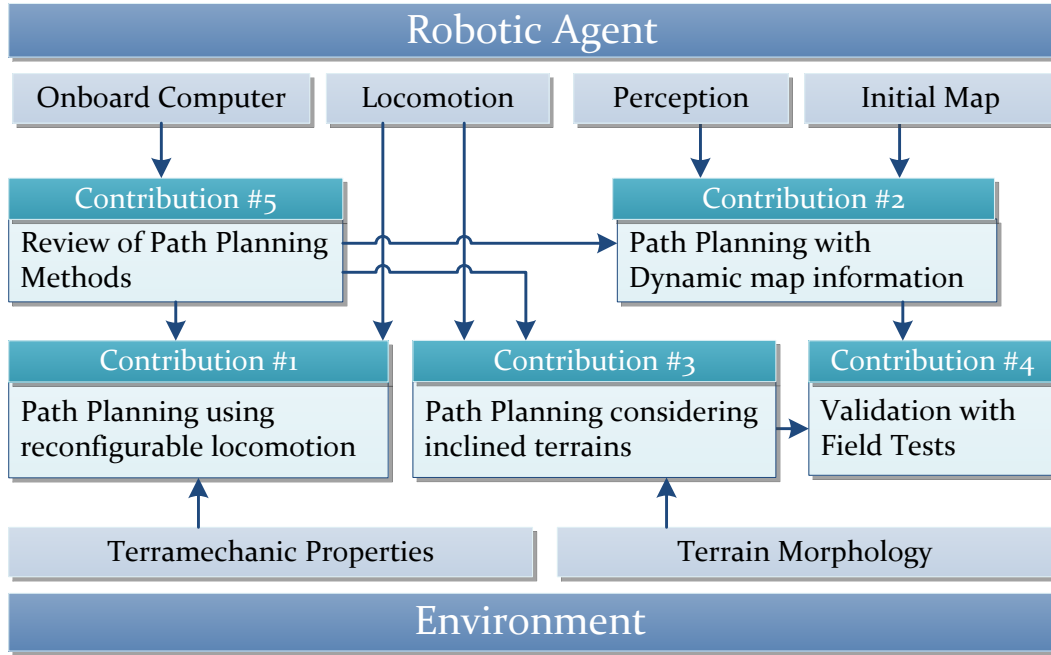


FIGURE 1.4: Diagram showing the contributions of this thesis. It details how these contributions relate to the agent and the environment in the context of the path planning problem.

1.2 Contributions

This thesis presents and details the work carried out to create a series of contributions to the state of the art of path planning for autonomous ground mobile robots. In particular, the presented work focuses on the use of robots working in the fields of planetary exploration and SAR. As previously seen, it is of high importance that these robots manage to navigate through unstructured outdoors environments in a fully autonomous fashion, without being teleoperated at all. As a preliminary step, an exhaustive review of the literature was made. This review is discussed in the next chapter of this thesis. As a result, it was found in the literature some room for improvement. It encompasses the use of different locomotion modes, the combination of maps with different formats and the consideration of the irregularity of the terrain. Figure 1.4 shows a conceptual schematic containing the contributions of this thesis, a total of five. These are exposed together with their implications with the system that moves, the robotic agent, and its surrounding environment. The five contributions are detailed next:

- **Contribution 1: Optimal path planning considering multiple locomotion modes.** Unstructured environments like that of a disaster scenario or an extraterrestrial one may pose a challenge for the locomotion capabilities of any mobile ground robot. The composition of the terrain may not only increase the energetic cost of the robot but also restrict its movement. In an extreme case, the terrain may even entrap it, causing the failure of the mission. Furthermore, the shape of the terrain may also make it more difficult to drive on top of it. For

this reason, providing robots with a kinematic configuration capable to perform multiple locomotion modes can result in making them adapt better to the terrain. This is translated into providing the robot with more room to navigate. A path planner that acknowledges this feature is a must-have to ensure the advantage in question. Besides, the path planner itself should as well be capable to find optimal paths given a criterion based on using multiple locomotion modes. Having this in mind, the first contribution of this thesis is the use of an optimal path planner together with a cost function based on this locomotion reconfiguration capability. In this way, the dynamic modelling of the involved locomotion modes feeds this cost function. In particular, this contribution has as reference the Rosalind Franklin rover, which as mentioned before is capable to perform two locomotion modes: Normal-driving and Wheel-walking.

- Contribution 2: Multi-scale path planning combining initial long-traverse planning and dynamic local path replanning.** The information that describes the terrain can come from different sources and in different formats. For instance, in planetary exploration, the initial information about an environment comes from satellite imagery. It gives a rough estimation of what the rover may find on its way. However, there may be still some uncertainty as the sources are not accurate enough. For this reason, the initial information may be dynamically complemented with the information the rover perceives while it is driving. An example of this is the case in which the rover detects obstacles on its way that were not considered in the initial planning. This translates into the need of making use of a dynamic path planner that acknowledges these updates in the environment data. Besides, it has to address new map information coming with different values of resolution and size compared to the initial information. This thesis presents a contribution where a novel dynamic global-local path planning scheme is proposed. The main novelty in this approach is the use of Partial Derivative Equation (PDE) algorithms, which are globally optimal by definition.
- Contribution 3: Creation of a direction-dependant cost function for planning the optimal traverse on slopes.** The third contribution of this thesis puts the focus on the traverse of scenarios including inclined terrains. The main consideration for this contribution is the fact that the robot consumes more or less energy on a slope depending on its orientation. This is because gravity affects differently according to the direction of the robot. Therefore, it is necessary to use a cost function based on a dynamics model that acknowledges this. In this way, the cost function is categorized as direction-dependent or anisotropic. In particular, this contribution combines the use of a PDE path planner, anisotropic in this case unlike in the previous contribution, with the cost function in question based on robot-terrain interaction. Moreover, it is included in this anisotropic approach the possibility to use a risk function that maximizes

the lateral stability of the robot by minimizing the roll angle. In this way, there is a trade-off between minimizing energy consumption and the risk of turning over, and it is up to the user to opt for prioritizing one of them.

- **Contribution 4: Field tests using experimental mobile platforms.** The fourth contribution presented in this thesis is the description of how the second and third contributions were experimentally validated. To do this, a rover prototype and a skid-steering mobile platform were used on outdoors irregular terrains. For both cases, it was carried out several field tests involving the emulation of complete path planning and navigation operations. This thesis deepens into the setup of these experiments and includes discussions about their execution.
- **Contribution 5: General classification of most relevant existing path planning algorithms used along with ground mobile robots.** The number of approaches to path planning existing in the literature is vast. This can be overwhelming to anyone who needs to choose a path planning algorithm to integrate it into a navigation system. For this reason, the fifth and last contribution of this thesis is the analysis of most of the path planning algorithms found in the literature to later produce a classification method that makes it easier to understand the different approaches in terms of way of functioning and applicability.

1.3 Context and Motivation

The main motivation for the work presented in this thesis is to progress in the field of autonomous navigation for mobile robots. As mentioned before using Table 1.1, there are many levels of autonomy. Path planning is key to achieving full autonomy. Thus, this work focuses on improving this kind of algorithm and its use in certain situations. This progress results in the completion of a PhD in Mechatronics. This PhD is from the Industrial Engineering School (see Figure 1.5a) of the University of Malaga (UMA), whose location is marked in Figure 1.5c. The author has performed the work presented in this thesis within the Department of Systems Engineering and Automation (ISA). In particular, the author has been a member of one of the department laboratories: the UMA Space Robotics Laboratory (SRL)².

The ISA research group (TEP-119) has long experience in the development of robotic solutions in SAR applications. Every year it organizes a conference in security, emergencies and catastrophes, where a public demonstration of the robotic systems is made³. During the PhD studies, the author participated in this conference by providing the same kind of PDE path planner used in the first and second thesis

²<https://www.uma.es/robotics-and-mechatronics/info/107542/robotica-espacial/> Accessed on 4th November 2021.

³<https://www.jornadascatastrofes.com/> Accessed on 29th August 2021.



(a) UMA Industrial Engineering School. Credit: www.cti.uma.es, foto taken in 2009.



(b) ESA European Space Research and Technology Center (ESTEC). Credit: esa.int.



(c) Sites of realization of this thesis (Image taken from Google Maps and modified).

FIGURE 1.5: Facilities where the work of this thesis was held.

contributions. Moreover, two national projects funded by the Spanish Government supported the work of this thesis, providing financial resources, equipment and the needed infrastructure. They are the following:

- **Multi-Robot System for Cooperation with First Response Human and Canine Rescue Teams in Catastrophe Scenarios (FIRST-ROB).** This project has as reference number DPI2015-65186-R. It focuses on the use of multi-robot systems to gather information during the initial stages of a disaster scenario. This project has supported the first, second and fourth contributions of this thesis.
- **Towards Resilient UGV and UAV Manipulator Teams for Robotic Search and Rescue Tasks (TRUST-ROB).** With reference number RTI2018-093421-B-I00, this project focuses on improving resilience on the mechatronic systems that are part of multi-robot response teams. This project has supported the third and fourth contributions of this thesis.

The SRL started functioning with the collaboration between UMA researchers and the European Space Agency (ESA). As part of his PhD studies, the author has carried out three research stays in the latter institution. These were done in the European Space Research and Technology Center (ESTEC) (see Figure 1.5b), with a total duration of one year. The collaboration agreement between this institution

and the University of Malaga started at the beginning of the author's PhD studies. It keeps active the moment this thesis is written. Besides, the author of this thesis collaborated on a European project funded by the European Commission through the Horizon 2020 program. Both projects are detailed next:

- **Autonomous Routing on Extreme Surfaces (ARES).** This project started in 2016 with contract number 4000118072/16/NL/LvH/gp between ESA and UMA. Its original purpose was the improvement of autonomy for reconfigurable rovers, creating path planning solutions capable to work with multiple modes of locomotion. In a second stage, this project was renovated in 2018 towards improving sample fetch missions. It is driven by the need of improving the navigation capabilities of rovers for future missions on the Moon and Mars. In the context of this project, the work towards achieving the first two contributions as well as the fourth one was made. The ESA Automation and Robotics Section provided the author with the facilities and the robotic systems to perform some of the experiments described in this thesis. This project supported the first, second and fourth contributions of this thesis.
- **Autonomous Decision making for very long traverses (ADE).** This project started in March 2019 and lasted until May 2021⁴. Funded by the European Commission under the Horizon2020 programme, the main objective of this project was to advance in the autonomy architecture of a rover prototype equipped with a robotic manipulator. The University of Malaga participated as one of the members of the consortium with the responsibility of developing the methods to make the system autonomously fetch a sample. In this project, the scope of such methods was to produce the motion of the arm needed to place the end effector on some location. The project finished with a series of field tests where the software provided by each of the partners was merged. This project supported the fifth contribution of this thesis. Moreover, the algorithms presented in this thesis served as the basis of the methods used in this project to autonomously plan the paths for the rover prototype.

Publications

All the contributions that support this thesis were presented as a result of the mentioned projects through both conference and journal publications. The Figure 1.6 presents a diagram containing not only these publications but also other complementary ones in which the author of this thesis was involved. The publications containing the contributions of this thesis are the following:

- **Path Planning for Reconfigurable Rovers in Planetary Exploration.** Authors: Pérez-del-Pulgar, C. J., Sánchez, J. R., Sánchez, A. J., Azkarate, M., Visentin, G. Published in *IEEE International Conference on Advanced Intelligent Mechatronics*

⁴<https://www.h2020-ade.eu/> Accessed on 6th October 2021.

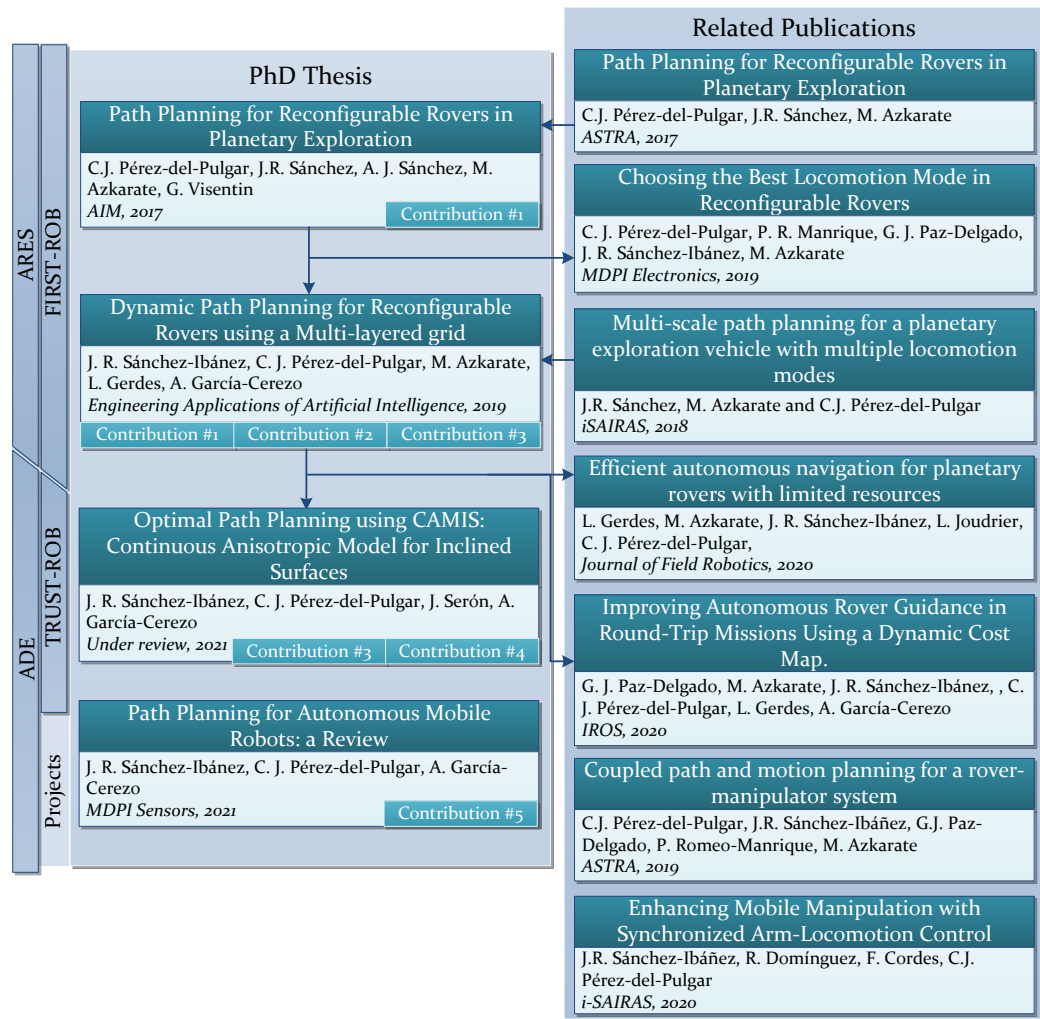


FIGURE 1.6: Diagram showing the publications that support this thesis, together with those that precede them and those that extend them.

(AIM), pp. 1453-1458, in 2017. This conference publication presents the first contribution of this thesis. It details the use of a path planner that considers multiple locomotion modes for a planetary wheeled mobile robot. A V-REP simulation environment is used to validate the proposed path planning solution, and it is demonstrated how the energy consumption can be minimized by considering these locomotion modes in areas containing different types of terrain. It was supported by the ARES project.

- **Dynamic Path Planning for Reconfigurable Rovers using a Multi-layered Grid.** Authors: Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., Azkarate, M., Gerdes, L., García-Cerezo, A. Published in *Engineering Applications of Artificial Intelligence* (2019 Impact factor: 4.201, Q1), vol. 86, pp. 32-42, in 2019. Based on the previous conference publication, this journal publication not only considers the use of reconfigurable rovers but also the information that describes the environment coming from different sources. This is possible thanks to the use of a grid made up of two layers, along with a path planning strategy that

combines an initial planning process on one layer with a dynamic update on the other. Numerical results, as well as results from field tests, are provided to validate this solution. In this way, this publication supports three of the contributions of this thesis, all of them except the third and the fifth ones. Besides, it is supported by the ARES and FIRST-ROB projects.

- **Path Planning for Autonomous Mobile Robots: a Review.** Authors: Sánchez-Ibáñez, J.R., Pérez-del-Pulgar, C.J., García-Cerezo, A. (2021). Published in *Sensors* (2020 Impact Factor: 3.576, Q1), vol. 21, pp. 7898, in 2021. This journal publication results from the analysis of path planning algorithms over the years during the PhD period. As a result, a novel classification is proposed, covering most of the existing path planning approaches that can be found in the literature. It is supported by the ADE project and supports the fifth contribution of this thesis.
- **Optimal Path Planning using CAMIS: Continuous Anisotropic Model for Inclined Surfaces.** Authors: Sánchez-Ibáñez, J.R., Pérez-del-Pulgar, C.J., Serón, J., García-Cerezo, A. Preprint under review for journal publication at the time this thesis is written. As the energetic performance of mobile ground robots differ according to their orientation on top of a slope, a path planner acknowledging this is desirable. This preprint details CAMIS, the cost model used to represent this anisotropic behaviour along with a compatible path planner. Besides, the lateral inclination is also considered to reduce the probability of turning over. In a similar way to the previous publication, this one provides simulation and field test results. It supports contributions three and four and is funded by the TRUST-ROB project.

Moreover, the first two publications that support this thesis were preceded each of them by a publication in the Advanced Space Technologies in Robotics and Automation (ASTRA) symposium and the International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS) respectively:

- **Path Planning for Reconfigurable Rovers in Planetary Exploration.** Authors: Sánchez-Ibáñez, J. R., Perez-del-Pulgar, C. J., Azkarate, M. Published in *Advanced Space Technologies in Robotics and Automation (ASTRA)*, in 2017. Sharing the same title as one of the mentioned publications that supports this thesis, this conference publication focuses more on the technical aspects of the simulation environment used along with the robotics software.
- **Multi-scale path planning for a planetary exploration vehicle with multiple locomotion modes.** Authors: Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., Azkarate, M. Published in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, in 2018. This short conference publication already gives some hints about the use of a multi-layered grid for path

planning, which is fully detailed in the second publication that supports this thesis.

The contributions provided by this thesis and supported by the aforementioned publications have also influenced other ones, also indicated in Figure 1.6. Here are introduced some of them in which the PhD candidate is co-author:

- **Choosing the Best Locomotion Mode in Reconfigurable Rovers.** Authors: Pérez-del-Pulgar, C. J., Romeo-Manrique, P., Paz-Delgado, G. J., Sánchez-Ibáñez, J. R., Azkarate, M. Published in *Electronics* (2019 Impact Factor: 2.412, Q2), vol. 8, pp. 818, in 2019. This conference publication extends the modelling done for the first contribution and also provides a method to determine when to switch from Wheel-walking to Normal-driving, according to the information provided by the robot onboard sensors.
- **Efficient Autonomous Navigation for Planetary Rovers with Limited Resources.** Authors: Gerdes, L., Azkarate, M., Sánchez-Ibáñez, J. R., Joudrier, L., Perez-del-Pulgar, C. J. Published in *Journal of Field Robotics* (2020 Impact Factor: 3.581, Q1), vol. 37, pp. 1153-1170, in 2020. This journal publication is a field report of the experiments carried out near ESA facilities using an experimental prototype rover. The path planner used is the same that is described in the journal publication *Dynamic Path Planning for Reconfigurable Rovers using a Multi-layered Grid*.
- **Improving Autonomous Rover Guidance in Round-Trip Missions Using a Dynamic Cost Map.** Authors: Paz-Delgado, G. J., Azkarate, M., Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., Gerdes, L., García-Cerezo, A. J. Published in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7014-7019, in 2020. In this conference article, it was presented an improvement to the path planner developed in the previous journal publication *Dynamic Path Planning for Reconfigurable Rovers using a Multi-layered Grid*. It is focused on round-trip missions such as the one the SFR will be expected to perform in the future.
- **Coupled path and motion planning for a rover-manipulator system.** Authors: Sánchez-Ibáñez, J. R., Paz-Delgado, G. J., Romeo-Manrique, P., Pérez-del-Pulgar, C. J., Azkarate, M. Published in *Advanced Space Technologies in Robotics and Automation (ASTRA)*, in 2019. The simulation results of the motion planning of a mobile manipulator were presented in this conference paper. The algorithm used to plan the path for the mobile base is the same used in contributions 1 and 2 of this thesis.
- **Enhancing Mobile Manipulation with Synchronized Arm-Locomotion Control.** Authors: Sánchez-Ibáñez, J. R., Domínguez, R., Cordes, F., Pérez-del-Pulgar, C. J. Published in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, in 2020. This conference publication

presents the preliminary results of some tests done with the *SherpaTT* rover, owned by the German Research Center of Artificial Intelligence (DFKI). These tests consisted of validating the path and motion planning method developed under the ADE project.

1.4 Thesis Outline

The structure of this thesis rests on seven chapters. All of them are written to present and define each of the contributions that make up this thesis. Each chapter after the current one and except the last one is written following a certain format. Each of them begins with a brief introduction about the concepts that are going to be presented. Thereafter, it is provided with an elaboration of these concepts, exposing each of them in detail. Then, each chapter ends up with room for a summary and some conclusions. It shows the discussion about the involvement of the presented concepts in the overall work and the thoughts of the author with regards to them. The chapters in question that make up this thesis work are the following:

- **Chapter 1: Introduction.** It is the current chapter. It starts by presenting the main questions that motivate the realization of this thesis. Thereafter, it provides a disclosure of the key objectives. Then, this chapter introduces the contextualization of the work carried out. It ends up here, detailing the structure of this thesis.
- **Chapter 2: State of the Art in Path Planning for Ground Mobile Robots.** This chapter introduces an extensive revision of path planning, which is at the same time the fifth (and last) contribution of this thesis. It provides an analysis of most of the existing path planning algorithms that can be found in the literature. It proposes a classification method that aims to ease their understanding. Later on, this chapter presents an analysis of the main criteria used in path planning for autonomous navigation.
- **Chapter 3: Isotropic Optimal Path Planning for Reconfigurable Rovers.** It elaborates the first thesis contribution about the use of many locomotion modes in path planning. This chapter details the process to make a model that considers various modes of locomotion. In the end, it explains the conditions and constraints due to its use along with an isotropic PDE planner.
- **Chapter 4: Dynamic Multilayered Path Planning.** This chapter focuses on the second contribution of this thesis. It defines the problem of using information from many sources in autonomous navigation. Moreover, it questions how a path can be dynamically updated. Then, it details a solution in the form of a multi-layered path planning architecture. Later on, this chapter finalizes with a discussion about the use of this novel solution.

- **Chapter 5: Anisotropic Cost Model for Traversing Inclined Terrains.** The third contribution of this thesis is fully explained in this chapter. Here it is made a step further in the formulation of the path planning problematic. Now the different energetic costs of the vehicle due to its orientation on any slope must be accounted for. This entails the use of an anisotropic PDE path planner along with a direction-dependent cost model that considers this new criterion.
- **Chapter 6: Experiments.** This chapter presents the fourth contribution of this thesis. It discloses the different experiments carried out to validate all the previous contributions. Here is relevant the use of existing experimental platforms, including rovers and mobile rescue robots. Thereafter this chapter provides the results from these experiments, followed by some discussion.
- **Chapter 7: Conclusions and Future Work.** The last chapter of this thesis makes first a summary of the presented work in the previous chapters and presents the discussion about the extracted conclusions out of them. Then, it introduces a discussion about the presented contributions highlighting the remaining gaps. Those improvements that can be made to the algorithms introduced are also indicated. They may lead to novel future research work.

Chapter 2

State of the Art in Path Planning for Ground Mobile Robots

"All roads lead to Rome."

Proverb

2.1 Introduction

As stated in the previous chapter, autonomous navigation is a valuable asset for mobile robots. It helps to mitigate their dependency on human intervention. Yet, it also entails many tasks or problems to solve, e.g. path planning. This task lies in finding the best course of action to make a robot reach the desired state from its current one. For example, both states could be respectively the goal and the initial positions. This course of action comes in the form of a path, also named route in many other works. The path serves to guide the robot to the desired state in question. However, there may be numerous possible paths given the free space in which the robot can move. Path planning algorithms generally try to get the best path or at least an admissible approximation to it. Best is understood here in the same sense as optimal: the resulting path comes from minimizing one or more objective optimization functions. For instance, this path may be the one entailing the least amount of time. This is critical in missions such as those of the Search And Rescue (SAR) (Alenezi et al., 2018): victims from a disaster may ask for help in life or death situations. Another optimization function to consider could be the energy of the robot. In the case of planetary exploration, this is critical since rovers have limited energetic resources available (Ishigami et al., 2011). At the same time, the path generated by the planner must follow any imposed restrictions. These may come from the limitations in the adaptability of the robot to certain terrains. The locomotion of the robot and the characteristics of the existing terrain limit the kind of manoeuvres that can be performed. This consequently reduces the number of feasible paths that the path planner can generate.

The number of path planning approaches is immense and do not stop increasing year after year. Choosing the most suitable one for any navigation application

considering restrictions and optimality criteria is not trivial. As is explained later, existing path planning reviews and surveys tend to miss important approaches. This fact motivates the work carried out that is detailed in this chapter. It starts by describing in Section 2.2 the methods used to address the environment and locomotion information, which serve as inputs to any path planner. In particular, this chapter focuses on the use of ground mobile robots. This means the planners here introduced have worked with this kind of vehicle or at least they are compatible with them. Nevertheless, most of the path planners that are introduced later can be also applied with other kinds of mobile vehicles. This is the case of, for instance, maritime surface robots (Y. Liu and Bucknall, 2016) and underwater/aerial robots moving under plans using 2D modelled environments (Petres et al., 2005). Later on, Section 2.3 includes a detailed description of existing path planning categories. They are organized using a classification method based on their fundamentals, together with references to works that make use of them. Besides, this section also analyzes previous review and survey publications and highlights how they are not comprehensive enough to at least mention the most relevant path planning algorithms. The following four sections deal, each of them, with one of the four path planning categories: Reactive Computing (Section 2.4), Soft Computing (Section 2.5), C-Space Search (Section 2.6) and Optimal Control (Section 2.7). Finally, Section 2.8 contains a summary of the contents introduced by this chapter, together with a series of conclusions extracted from them.

2.2 Path Planning Workspace Modelling

A path planner needs to be fed with information describing the environment. This information can inform, for instance, about the presence of obstacles or the features of the surface that are relevant to the planning. Besides, the criteria used to calculate the path has to do with the way the robot interacts with this environment. For example, to just minimize path length perhaps the information of what areas can be traversed or not is enough, while to minimize energy the terramechanics and the way the robot steers shall be taken into account. This section introduces first different ways to format the map in a way usable by the planner. Later on, this section presents different approaches to address robot-terrain interaction according to the robot locomotion capabilities.

Environment Modeling

Surface mobile robots drive from one position to another within a certain region in space. Therefore, it is needed to consider how will the locomotion model interact with this surface and how the path planner will take care of it. For instance, some algorithms require the construction of a graph that represents somehow the environment in which the robot is moving. This is mostly the case of Graph Search algorithms, part of the C-Space Search category. Besides, Evolutionary algorithms like

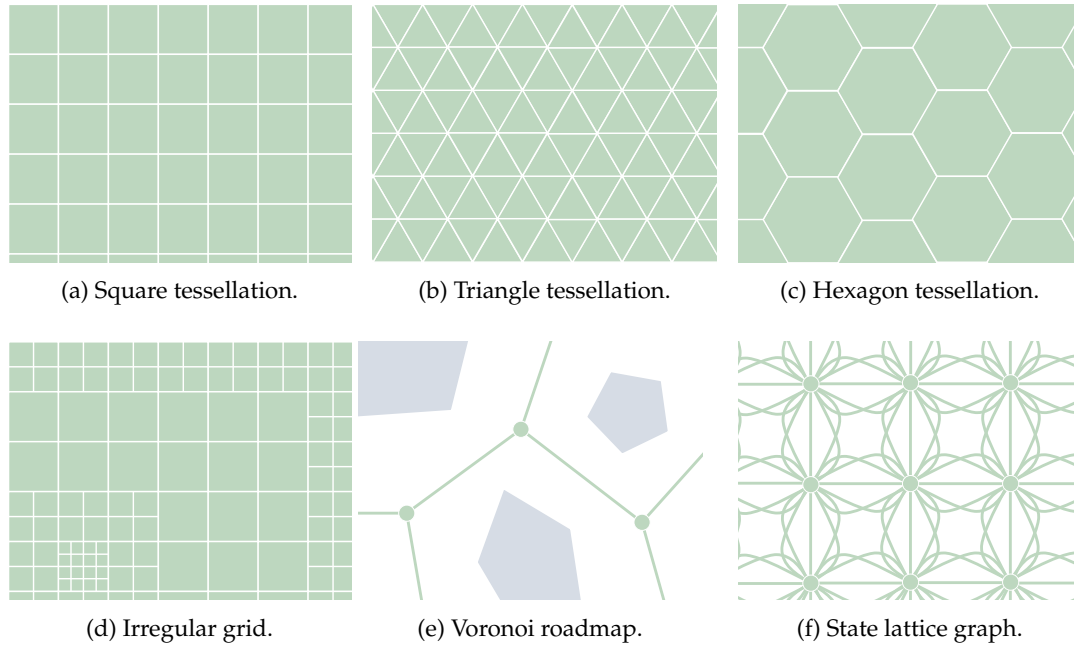


FIGURE 2.1: Different types of environment cell decomposition (a-d) and roadmap graphs (e-f).

the Ant Colony Optimizer (ACO) also can make use of a graph. This asset can represent how the terrain features that affect the robot navigation are spatially arranged in the scenario. In particular, the graph in question is assumed here to be built upon metric maps, acknowledging the existence of other kinds of maps out of the scope of this thesis, like topological and semantic (Yi et al., 2012). According to Souissi et al. (2013), there exists multiple ways to build a graph, as those shown in Figure 2.1. The work of Nash and Koenig (2013) also put some light into this classification. It distinguishes between Cell Decomposition and Roadmaps. The first of them consists of tessellating the surface into cells. These cells can be arranged using regular (P. Raja et al., 2012; Nash and Koenig, 2013; Souissi et al., 2013; H.-y. Zhang et al., 2018) or irregular grids (Nash and Koenig, 2013; Souissi et al., 2013). Figures 2.1a, 2.1b and 2.1c show how regular grids can be built using one out of three types of polygons: squares, triangles and hexagons. Its main advantage is the simple indexation of each node, which is translated into quick access to any of them and an optimized way to store them in memory (Algfoor et al., 2015). Irregular grids, like the one depicted in Figure 2.1d, allow adapting the grid better to terrain features with different values of resolution, at the expense of possibly getting worse paths (Petres et al., 2005). Other forms of Cell Decomposition are the Navigation meshes and Circle-based waypoint graphs as explained by Nash and Koenig (2013). The other form of representing the environment is, as mentioned, by using roadmaps. A roadmap is a graph built upon nodes connected by edges. Each node represents a possible state of the robot, while each edge indicates how to reach that state from another. Examples of roadmaps include Voronoi graphs (Barraquand et al., 1992) (see Figure 2.1e), Visibility graphs (Huang et al., 2004) and State-Lattice graphs (see Figure 2.1f). The latter

consists of making the edges be based upon motion primitives, so the resulting path is ensured to be feasible given the robot mobility constraints, especially when using Graph Search algorithms (Likhachev and Ferguson, 2009; Bergman et al., 2020).

The cells or the nodes from these graphs can store information regarding the surface at their location, in the form of static or dynamic elements (Patle et al., 2019). It can be for instance elevation information. A Digital Elevation Map (DEM) is a grid whose nodes have associated each of them a value of elevation. Elevation maps can be also represented by polygons, but regular grid maps are preferred (Choi et al., 2012). Shape-related features, such as the slope gradient or the surface roughness, can be extracted using convolution matrices (Papadakis, 2013). The size of the kernel and the DEM resolution will determine what kind of features are extracted. Moreover, this resolution defines the level of detail of the elements contained in the map. As shown in Figure 2.1d, this resolution can be non-uniform or multiple. The size of the grid can be chosen according to the scale at which the planning is performed: local in case of covering the immediate surroundings of the robot (more or less the reachable distance of the on-board sensors) and global if the area is bigger than that, usually using information from external sources such as satellites or drones.

With regards to how the cost is defined over the planner workspace, there are different ways. First of all, cost is here understood as the metric that the robot accumulates by moving. The objective of the path planner is to minimize this accumulation by producing the optimal path. The cost in question can be uniform, in the sense that the regions that can be accessed by the robot have always the same value. This approach can be used for doing collision avoidance path planning, where metrics such as the path length in a 2d plane are minimized. Non-uniform cost maps can be used to assign different values of cost to different accessible areas. It can be useful to, for example, define the energetic performance of the robot at each location. Moreover, the cost can be also defined according to a direction vector. This means that the robot will experience different values of cost depending on its heading. In this case, the cost is categorized as anisotropic (Shum et al., 2015), while in the contrary case the cost is isotropic. Besides, the steering manoeuvre of the robot can also have different values of cost according to its locomotion. Finally, it is worth remarking that the knowledge about the environment can be fully known, partially known or even fully unknown, requiring for the latest two a planning strategy capable to replan when this knowledge is updated.

Robot-Surface Interaction Modeling

A ground mobile robot interacts with the surface beneath it to propel itself. To perform this, there exist many different locomotion actuators, such as wheels, tracks, legs and even omnidirectional wheels. Figure 2.2 depicts six real examples of ground mobile robots using different configurations of actuators. The Koguma rover (in

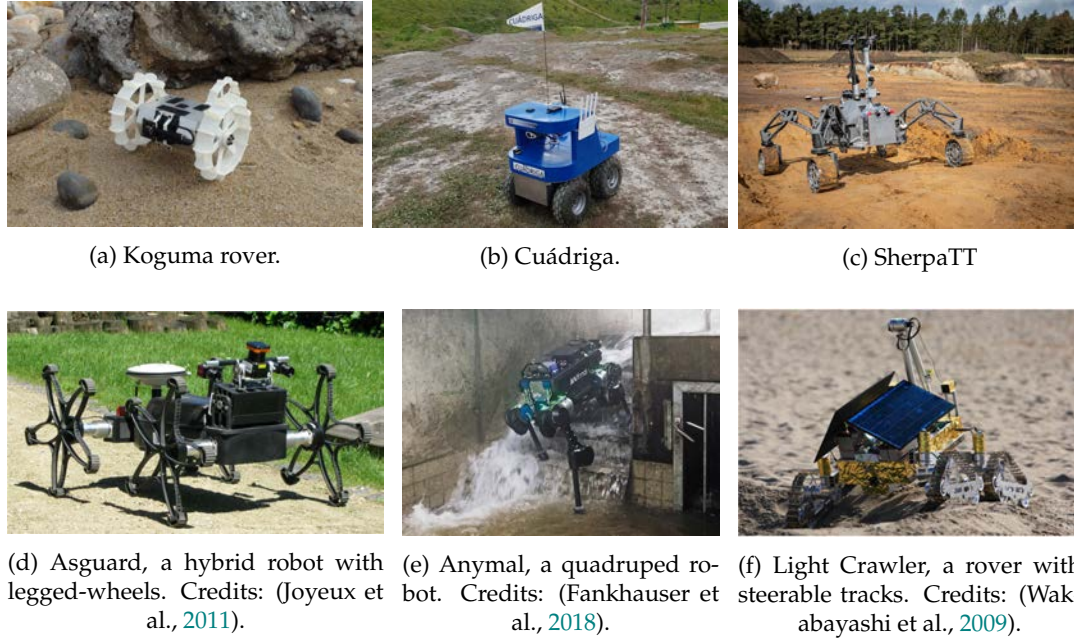


FIGURE 2.2: Examples of ground mobile robots with different kinematic configurations. (a) and (c) were reproduced in (Sánchez-Ibáñez et al., 2021) with permission of the University of Tohoku and the German Research Center for Artificial Intelligence (DFKI) respectively.

Figure 2.2a) has Differential drive locomotion. Cuádriga (in Figure 2.2b) is a four-wheeled Skid-Steering robot owned by the University of Malaga. SherpaTT (in Figure 2.2c) has four steerable wheels that allow it to execute Full-Ackermann, Crabbing and Point-Turn maneuvers. The other three robots shown in this Figure have different actuators such as legged-wheels, articulated legs and steerable tracks (Figures 2.2d, 2.2e and 2.2f respectively). All the introduced actuators, together with the joints linking them to the robot body, determine the kinematic structure and the dynamic behaviour of the robot. In other words, they determine the locomotion configuration of the robot. H. Zhang et al. (2020) summarize some kinematic and dynamic models of different well-known configurations of wheeled robots: Differential drive (Papadopoulos et al., 2007) (see Koguma robot (Laine et al., 2018) in Figure 2.2a as an example and the depiction of the model in Figure 2.3a), Ackermann (Marin et al., 2013) (see Figures 2.3b and 2.3d), Skid-Steering (Mandow, Martinez, et al., 2007) (see Figure 2.3c) and Omnidirectional (Wu et al., 2020). Some of them entail constraints relevant to path planning, such as the minimum turning radius of robots with Front-Ackermann (Takei et al., 2013) (see Figure 2.3b) or the high energy consumption of Skid-Steering robots in turning manoeuvres (Effati et al., 2020). Moreover, another model exists called Crabbing (see Figure 2.3e). It allows a robot to drive in a direction different to the one it is facing, due to having steering joints on top of all wheels (Patel et al., 2010). Furthermore, some kinematic configurations allow the robot to perform the Point-Turn manoeuvre, which makes them rotate without translating. It is worth mentioning the existence of articulated robots capable to reconfigure themselves to obtain some kind of benefit and perform multiple types of locomotion (see

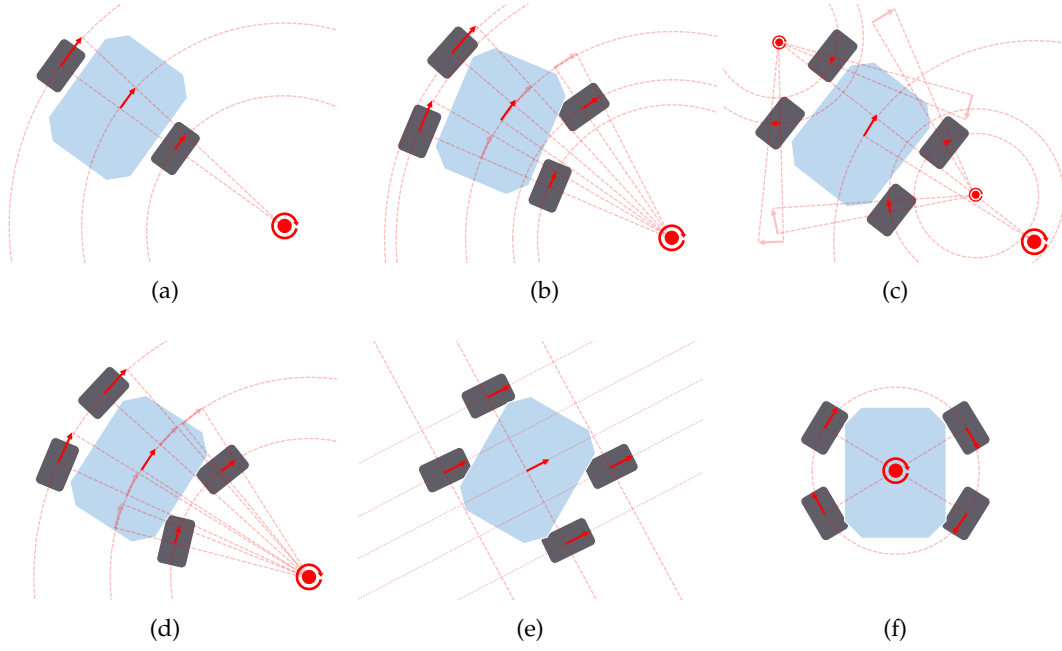


FIGURE 2.3: Locomotion models used for wheeled ground mobile robots along with path planners: Differential drive (a), Front-Ackermann (b), Skid-Steering (c), Full-Ackermann (d), Crabbing (e) and Point-Turn (f).

SherpaTT in Figure 2.2c as an example of this). For instance, articulated robots with tracks can actively control their stability while driving on rough terrains (Rohmer, T. Yoshida, et al., 2010; Brunner et al., 2015). Others use wheel-on-legs configuration to execute a locomotion mode called Wheel-walking (Azkarate, Zwick, et al., 2015; Malenkov et al., 2017). This mode is designed to overcome soft terrains where a robot could get stuck. In a similar way, Push-pull locomotion imitates the motion of a caterpillar to also increase traction in this kind of terrains (Moreland et al., 2011; Creager et al., 2012; Colin Creager et al., 2015). Path planning algorithms acknowledging this reconfiguration capability is a must-have for this kind of robot, as they can find paths that take advantage of their high adaptability (Rohmer, Reina, et al., 2010).

The robot locomotion will adapt better or worse depending on the terrain features that were briefly mentioned before. These features may be related to either the morphology (shape) or the composition of the terrain. One of them is the terrain inclination or slope gradient (Weih Jr et al., 2004). The slope gradient takes influence on the roll and pitch orientation angles of the robot, which is important to consider to preserve stability (Miró et al., 2010; Brunner et al., 2015; Norouzi et al., 2017). Besides, it can also influence the energetic performance of the robot according to its direction. This dependency on direction is due to the effect of gravity, making the robot consume different amounts of energy according to whether it is climbing, descending, going laterally or going diagonally through the slope (Rowe et al., 1990; Choi et al., 2012; Ganganath, Cheng, and Chi, 2015; Ganganath, Cheng, Fernando, et al., 2018; Gruning et al., 2020). Another relevant terrain feature is the roughness. It is

the measure of how diverse the normal vectors are (Krüsi et al., 2017; Lindsay et al., 2019) and may affect the vibration experienced by the robot. Other terrain shapes can be negotiated by the robot according to its chassis. For instance, it can overcome rocks using the body clearance, which is the space between the body lower surface and the terrain surface (Raghavan et al., 2019; Otsu et al., 2020). The presence of negative obstacles such as holes or ditches can be problematic as they are difficult to capture by the robot sensors (Hines et al., 2021). With regards to the terrain composition, it takes influence on the dynamics underlying the robot-surface interaction. This surface may be more rigid or deformable (Taghavifar et al., 2020). This affects the way the robot adheres to the surface, even restricting its motion (Arvidson, Bell, et al., 2010). The slippage is the metric that, in general terms, measures how the real speed of the robot differs from the commanded one, usually by doing a ratio between them (Ishigami et al., 2007; Sutoh et al., 2015). Some works consider both slippage and the slope gradient of the terrain to make a more accurate estimation of the robot while traversing rough terrains (Hiroaki Inotsume et al., 2016; H. Inotsume et al., 2020). The magnitude of gravity can also affect the dynamics of the interaction between the robot and the terrain (Papadakis, 2013; Niksirat et al., 2020). Finally, path planners usually make use of cost functions that encompasses multiple terrain features related to shape and composition. For instance, Ishigami et al. (2011) introduces the *dynamic mobility index*, encompassing stability, slippage, elapsed time and energy consumption. Moreover, there may exist other elements not directly related to the terrain that may still affect the performance of the robot while navigating. One of them is the solar radiation (Plonski et al., 2013; Sutoh et al., 2015), which can be modeled as a dynamic function (Kaplan et al., 2016). Groves et al. (2021) map other kinds of radiation that may harm the robot, like in scenarios of nuclear dismantlement. Besides, the concept of risk can be of high importance to prevent the robot from getting into a dangerous situation (Ono et al., 2015), like increasing the cost with the proximity to obstacles (Valero-Gomez et al., 2013).

2.3 General Classification

Figure 2.4 depicts four categories of path planning: Reactive Computing, Soft Computing, C-Space Search and Optimal Control. Each of them is split into two subcategories that will be detailed in later sections. This classification rests on the principles and fundamental mechanisms used to construct and return a path. In many past reviews, two kinds of distinction were made: according to whether the environment is dynamic or not, *Online* and *Offline* path planners respectively (P. Raja et al., 2012), and the size of this environment, local and global. Usually *Online* is associated with local and *Offline* to global. The main issue with this is that there are algorithms that can be considered in both categories. An algorithm with no replanning capabilities could be used online due to its high computational speed. The contrary also could happen. For instance, a Reactive Computing algorithm called the Dynamic Window

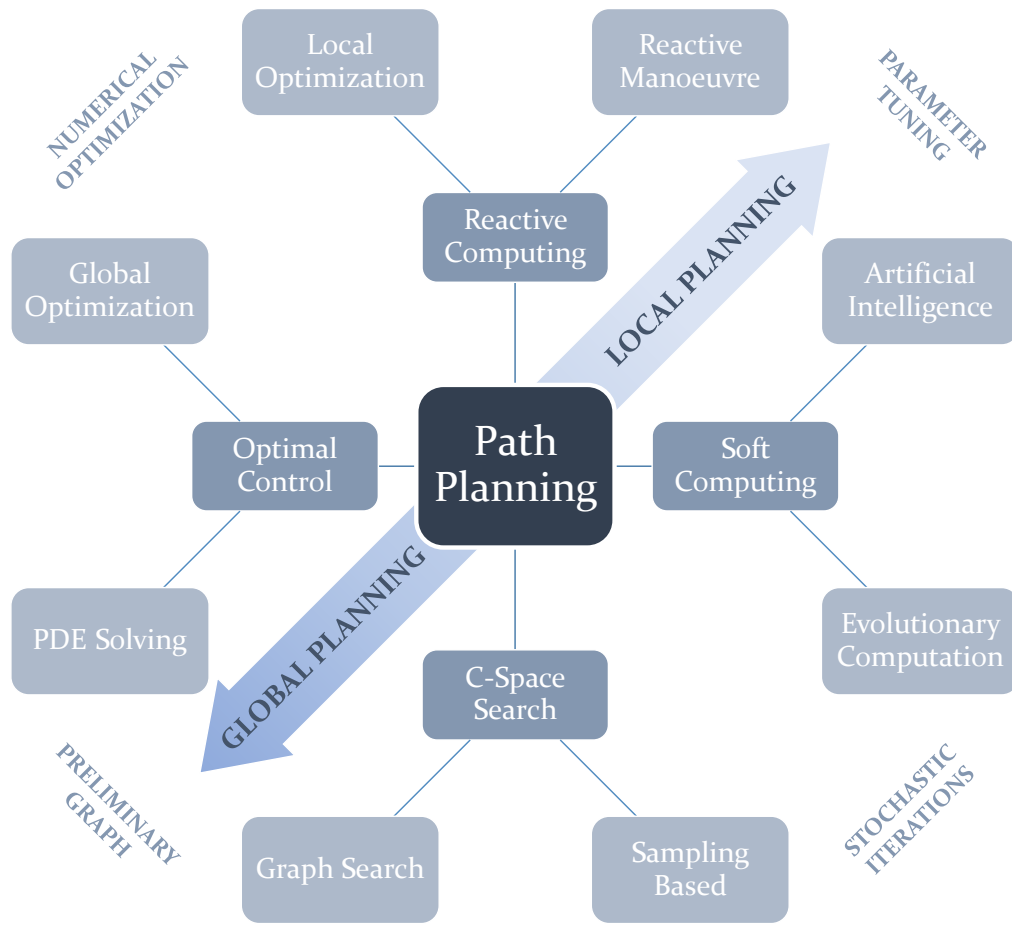


FIGURE 2.4: Schematic showing the proposed classification of existing path planning approaches. There are four main categories, each of them containing two subcategories. Two adjacent subcategories from different categories have features in common. The schematic also indicates how some subcategories are more inclined towards either *Global Planning* or *Local Planning*.

Approach (DWA), is usually used for local planning (H.-y. Zhang et al., 2018), but can also be found as a global planner (F. Zhang et al., 2019). Vagale et al. (2021) presented an interesting division between algorithms that require a preliminary map representation (*Classic*) (Souissi et al., 2013) and those which not (*Advanced*). *Classic* includes Graph Search methods while *Advanced* addresses Soft Computing and Sampling Based algorithms. Souissi et al. (2013) proposed several clear and reasonable path planning classifications: according to the robot model (holonomic, non-holonomic, kinodynamic), according to the map model requirement (needed or not needed beforehand), according to the replanning capability (offline or online) and according to whether the algorithm always calculates the same solution or not according to preliminary configuration parameters (deterministic or probabilistic).

The main purpose of the classification proposed in this chapter and depicted in Figure 2.4 is two-fold. First, this classification aims to encompass a larger variety of algorithms than those that are tackled in past reviews. Many past reviews propose

TABLE 2.1: Main surveys and reviews of global path planning algorithms found in the literature.

Publication	Reactive Computing		Soft Computing		C-Space Search		Optimal Control	
	Reactive Manoeuvre	Local Optimization	Evolutionary Computation	Artificial Intelligence	Sampling Based	Graph Search	PDE Solving	Global Optimization
P. Raja et al. (2012)	Yes	No	Yes	No	No	No	No	No
Nash and Koenig (2013)	No	No	No	No	Yes (RRT and PRM)	Yes (Any-angle and A*)	No	No
Souissi et al. (2013)	Yes (APF)	No	Only mentions	No	Yes	Yes	No	No
Elbanhawi et al. (2014)	Only mentions	No	Only mentions	Only mentions	Yes	Only mentions	No	No
González et al. (2015)	No	Yes	No	No	Yes	Yes	No	Yes (NLP)
Mac et al. (2016)	Yes	No	Yes	Yes	Yes	No	No	No
Noreen et al. (2016)	No	No	Only mentions	No	Yes	Only mentions	No	No
Injarapu et al. (2017)	No	No	Yes (GA)	Yes	No	No	No	No
Ravankar et al. (2018)	No	Yes	No	No	No	No	No	Yes (NLP)
H.-y. Zhang et al. (2018)	Yes	No	Yes	Yes (ANN)	No	Yes	No	No
Zafar et al. (2018)	Yes (APF)	No	Yes	Yes (ANN)	Yes	Yes	No	No
Costa et al. (2019)	No	No	Yes (GA)	No	Yes (RRT)	Yes (A*)	No	No
Patle et al. (2019)	Yes (APF)	No	Yes	Yes	Only mentions (PRM)	No	No	No
Gómez et al. (2019)	No	No	No	No	No	No	Yes (eikonal solvers)	No
S. Campbell et al. (2020)	Yes	No	Yes	Yes	No	No	No	No
H. Zhang et al. (2020)	Yes	No	Yes	No	Yes	Yes (A*)	Yes (FMM)	Yes (DP)
H. Sun et al. (2021)	Only mentions (APF, DWA)	No	Only mentions	Yes	Only mentions (RRT and PRM)	Only mentions	No	No
Vagale et al. (2021)	Yes	No	Yes	Yes (DRL)	Yes (RRT and PRM)	No	No	No

or claim to present a general overview on path planning, but as is seen in Table 2.1 the majority of them suffer from important omissions. In this table, *Yes* means there is significant discussion about the algorithms in question. *Only Mentions* means the publication acknowledges the existence of at least one or more algorithms in that class are acknowledged. If there are one or two algorithms between parenthesis this means only those are mentioned/referred to briefly. Second, the nomenclature to refer to path planning categories is not clear in some cases. For instance, some other reviews make the distinction between *Classical* and *Heuristic* approaches (Mac et al., 2016; Zafar et al., 2018). Patle et al. (2019) refer to the latter as *Reactive*. However, the term *Classical* can be quite an ambiguous term as the majority of planning algorithms used are based on methods with one or more decades of life. This class is also used to encompass many algorithms with completely different ways of functioning. The term *Heuristic* not only has been used to refer to Evolutionary and Artificial Intelligence algorithms (Mac et al., 2016), but it has been also used to refer to Graph Search based planners (H.-y. Zhang et al., 2018). Besides, Figure 2.4 shows how in general

terms each of these categories tend to be used mostly for either local or global planning. Moreover, each of the subcategories can also have something in common in their functioning with subcategories from other categories, such as the use of numerical methods, the existence of parameters to tune beforehand, the requirement of modelling the map with a graph or the use of stochastic iterative processes.

2.4 Reactive Computing Based Path Planning Algorithms

Reactive Computing algorithms model the environment by distinguishing between traversable and non-traversable (or obstacle) regions. These algorithms are commonly used for local path planning: they quickly tackle the dynamic incoming information describing the surroundings of the robot and update in real-time the path. For this reason, Reactive Computing algorithms are used along with robots equipped with limited perception systems. They calculate the next action or short path that makes the robot avoid close obstacles, while at the same time following a global path. An important drawback of many Reactive Computing algorithms is the creation of local minimum paths, which can make the robot get stuck. There exist two subcategories of this kind of algorithm: the Reactive Manoeuvre methods and the Local Optimization methods. The first deals with algorithms that produce the next immediate action or manoeuvre given the presence of nearby obstacles. The second encompasses algorithms that modify an existing path given the existence of obstacles contacting it.

Reactive Manoeuvre

Algorithms considered as Reactive Manoeuvre define the reaction of the robot in a situation where it navigates and finds obstacles on its way. They do not require a preliminary model of the environment in a graph format (Souissi et al., 2013). The position and shape of these obstacles are addressed by the Reactive Manoeuvre algorithms in different deterministic ways, but to produce the reaction in question, in the form of a steering or a velocity command, investing low computational resources. The omission of global environment information makes this kind of algorithm focus on local planning, doing replanning instantaneously. However, this fact makes these algorithms in turn sub-optimal and non-complete, since the robot may get stuck in local minimum points. To overcome this, some approaches combine Reactive Manoeuvre methods with algorithms from other categories that consider global information (Vadakkepat et al., 2000; R. Raja et al., 2015; Zhou et al., 2018). Moreover, Reactive Manoeuvre can be used in dynamic environments that contain moving obstacles (Ge et al., 2002).

Three kinds of formulation of the robot reaction can be distinguished:

- **Repulsive and attractive field forces.** The Artificial Potential Fields (APF) and the Vector Field Histogram (VFH) methods tackle the presence of obstacles

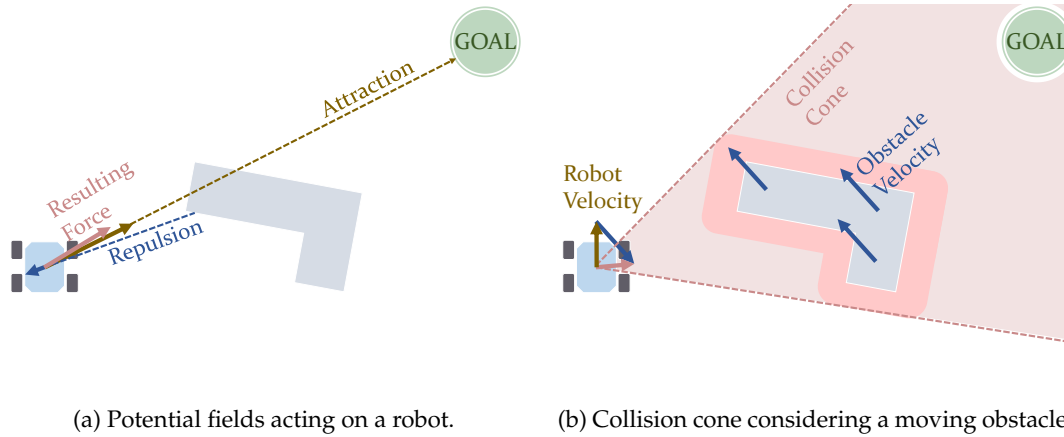
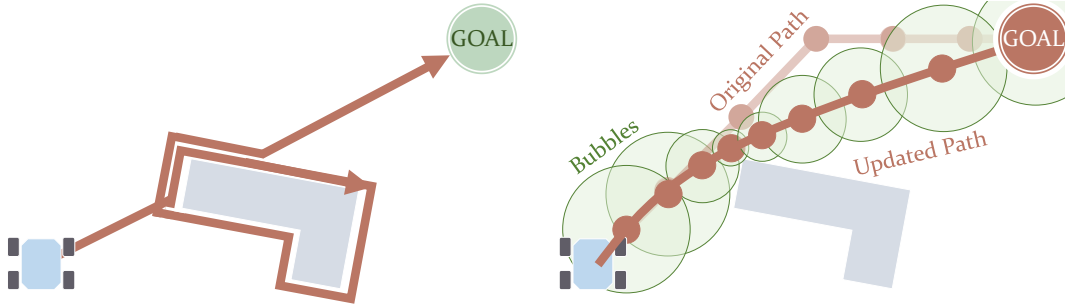


FIGURE 2.5: Graphical representations of concepts used in the Artificial Potential Fields (APF) (a) and Velocity Obstacle (b) algorithms.

using field models. Figure 2.5a depicts a graphical explanation of how the APF method works. The goal is assigned with a force that attracts the robot, while repelling forces are assigned to the locations of obstacles. In this way, the robot gets further from obstacles while driving towards the goal (O. Khatib, 1985). The VFH, proposed by Borenstein et al. (1991), creates a polar histogram to evaluate the density of obstacles around the robot, selecting the steering angle with the lowest density of obstacles.

- **Circumvention of obstacles.** The Bug algorithms, Bug1 and Bug2, make the robot circumvent any obstacle found on its way until it reaches the goal (Lumelsky et al., 1986). The main difference between them is that Bug1 makes the robot drive the full boundary of any obstacle (see Figure 2.6a), while Bug2 can drive it only partially (S. Campbell et al., 2020). They despise optimality in favour of easiness in the implementation and very minimal computation. They can be used on robots equipped only with sensors that just detect obstacles in their immediate vicinity. In this way, these robots either drive towards the goal or drive along the boundaries of obstacles they find. Q.-L. Xu et al. (2017) use the Bug algorithm considering turning radius constraints, producing paths with smooth turns.
- **Velocity space.** This kind of algorithms are adequate for dynamic environments. The following approaches are focused on producing a velocity command for the robot. Velocity Obstacle methods calculate a safe trajectory considering the velocity vectors of both the robotic agent and any other moving obstacle (Fiorini et al., 1998; Kuwata et al., 2013). Figure 2.5b depicts how this calculation is based on evaluating a cone of collision with a vertex on the robot and covering the obstacle. Moreover, other works introduced the use of the Front-Ackermann locomotion model (see Figure 2.3b) to find local paths for cars. Wilkie et al. (2009). Another Velocity Space method is the Dynamic Window Approach (DWA), an algorithm that searches in the velocity space of the



(a) Path after using Bug1 algorithm.

(b) Path after experiencing stretching.

FIGURE 2.6: Graphical representations of concepts used in Bug algorithm (a) and Elastic Bands (b) algorithm with Bubble bounds.

robot for a velocity command to follow. This velocity command must make the robot take a collision-free circular trajectory, bounded by some admissible speed values and a time window (Fox et al., 1997). This solution may not be the global optimal one, but rather a local optimal one (H.-y. Zhang et al., 2018). Other approaches use this algorithm for energy-minimization path planning (Henkel et al., 2016; Xie et al., 2018). Although commonly used as a local planner, F. Zhang et al. (2019) have proposed its use at a global scale as a global path planner.

Local Optimization

These algorithms usually start from a pre-existing path and modify it according to the existing obstacles, perceived at a local scale in general. Therefore, they do not need a preliminary map model. Here it is prioritized to keep computational use to the minimum at the expense of losing optimality or even completeness. There are different options to modify the path, ranging from the selection of velocity profiles within a velocity space to the stretching and elongation of the path under the effect of artificial forces.

The use of Elastic Bands in path planning was proposed by Quinlan et al. (1993). This method deforms an existing collision-free path according to the obstacles. The deformation consists of either a stretching (see Figure 2.6b) or an elongation. From a set of points in this path a set of overlapping subregions, called Bubbles, is created. These Bubbles cover collision-free areas and their size is bounded by the distance to obstacles. This entails that smaller and more numerous Bubbles are present in the portions of the path closer to obstacles. It can be used in dynamic environments, although big changes may lead to the failure of the method (Quinlan et al., 1993). Moreover, it has also been adapted to non-holonomic vehicles by complying with curvature constraints (M. Khatib et al., 1997) and using *Bezier* curves (Pérez et al., 2018). An extension to Elastic Bands include time constraints and is named Timed

Elastic Bands (TEB). This extended version addresses the kinodynamic constraints of the robot, which are relevant for local planning (Rösmann et al., 2012).

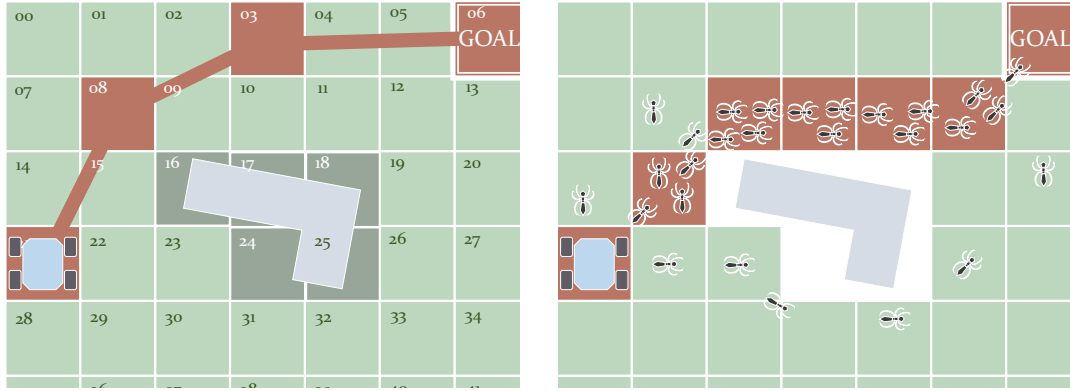
2.5 Soft Computing Based Path Planning Algorithms

This kind of algorithm does not intend to find the exact optimal solution, but rather approximate it tolerating a certain range of imprecision. In general, they require the tuning of certain parameters done by the user to work properly according to the characteristics of the environment. They can deal even with dynamic environments and are adequate for problems involving a large number of variables and high degrees of freedom (Mac et al., 2016). Yet, they in general demand a high number of computational resources and their functioning is not deterministic. This review follows the classification proposed by Mirjalili and Dong (2020), which distinguishes between Evolutionary, Fuzzy control and Machine Learning methods. The first one uses techniques inspired by biology and nature: they start with a system formed by individuals that change over time, i.e. evolve. Fuzzy control and Machine Learning methods are here together as part of a subcategory named Artificial Intelligence. They use fuzzy rules and neural networks respectively to produce controllers. These controllers are very useful for navigating through initially unknown scenarios and in general produce paths according to the obstacles the robot detects on its way. To sum up, Soft Computing algorithms allow tuning a series of repetitive elements, either nature-based individuals, fuzzy rules or artificial neurons, to generate a path.

Evolutionary Computation

The Evolutionary algorithms are also known as Meta-heuristic or Nature-inspired (Fausto et al., 2020). These algorithms generate a path that results from the evolution of a population. This population is made up of intelligent individuals whose actions are modelled after behaviours found in nature (Mirjalili and Dong, 2020). These actions may involve modifying themselves and/or interacting with other individuals. In some cases, these operations imply a motion by the individuals in the free space of the environment, i.e. in the space reachable by the robot. After performing a series of these operations, the algorithms approximate the optimal solution. The resulting path and the time invested to converge depend on the behaviour policy assigned to the individuals, the nature of the scenario (which can be dynamic (P. Raja et al., 2012)) and the values assigned by the user to certain configurable parameters. An example of the latter is the number of individuals that populate the path planning problem. Evolutionary algorithms include Genetic methods and Swarm Optimizers.

Genetic algorithms work with individuals modelled after chromosomes (Tang et al., 1996). Here, chromosomes are sets of binary numbers that encode a solution, like the grid cells forming a path as depicted in Figure 2.7a. The Genetic algorithm starts with a random set of chromosomes that evolve by reproducing (creating and



(a) Functioning of the genetic algorithms to perform path planning. The path in this figure has the form of a chromosome with genes 21 - 08 - 03 - 06.

(b) Functioning of the ACO algorithm. Simulated ants deposit more pheromones in the shortest path. Eventually the majority will follow this path.

FIGURE 2.7: Examples of Evolutionary algorithms: Genetic (a) and Swarm Optimizer ACO (b).

deleting them), crossing their information and mutating (introducing changes to incentive the exploration of the solution) (Ram et al., 1994). As a result of continuously repeating these processes, the algorithm converges, more slowly as it gets close to the optimal solution (H.-y. Zhang et al., 2018). Han et al. (1997) used Genetic algorithms for finding the shortest path in environments with dynamic obstacles. It has been used in global planning with large grids that model the environment (Alajlan et al., 2013). However, they are non-complete as it is not ensured they find the solution if it exists (H. Zhang et al., 2020).

Unlike the algorithms based in Genetic methods, Swarm Optimizers use agents modelled after animals in most cases. These agents move and actuate in the free space of the robot. After a series of iterations, the motion of these individuals towards the goal creates a pattern that eventually converges to the resulting path. Table 2.2 introduces some of the models used that can be found in the literature. The Particle Swarm Optimizer (PSO) algorithm stands out because of its simplicity. It is inspired by the behaviour of certain groups of animals like fish schools (Y. Zhang et al., 2013). It creates a series of particles that relocate themselves over time until the algorithm converges. These algorithms search for the best positions and communicate to each other, considering their previous experience (Lu et al., 2008). The PSO will find a solution if the optimal exists, so it is considered a complete algorithm (H. Zhang et al., 2020). The solution provided by PSO could be sub-optimal though (H. Sun et al., 2021). Another well-known algorithm is the Ant Colony Optimizer (ACO) which, as the name indicates, simulates the behaviour of ants. These insects move while leaving a trail of pheromones in their search for food. This trail can be tracked by the rest of the ants. Those places that contain more pheromones make up the waypoints of the best-found path. Figure 2.7b depicts this concept in a situation with an obstacle between the start and goal positions. Here, the best path is the shortest one. Following the same strategy, virtual ants can move on a grid leaving

TABLE 2.2: Intelligent models used for some Swarm Optimizer algorithms, together with some of the behaviours they perform inspired by nature.

Individuals	Behaviours	References
Particles	Best position memory Variable velocity	(Lu et al., 2008) (Y. Zhang et al., 2013) (Mac et al., 2017) (Tharwat et al., 2019)
Dragonflies	Hunting and migration	(Muthukumaran et al., 2019)
Grasshoppers	Attraction and repulsion	(Elmi et al., 2018) (Elmi et al., 2020)
Wolves	Hierarchy system Hunting for preys	(Tsai et al., 2016) (Doğan et al., 2018)
Whales	Spiral bubble-nets	(Mirjalili and Lewis, 2016) (Dao et al., 2016)
Cuckoos	Lévy flights Brood parasitism	(Prases Kumar Mohanty et al., 2013)
Bats	Echolocation	(Ghosh et al., 2017)
Bacteria	Foraging	(Hossain et al., 2015)

more or fewer pheromones according to their state concerning the goal (Cong et al., 2009; Cao et al., 2016). However, they can fall into a local minimum and provide a sub-optimal solution (Wen et al., 2006; You et al., 2016) or not find it even if one exists (Luo et al., 2020). This algorithm has been also simulated with several DEM to minimize time (L. Wang et al., 2019) and energy (Sangeetha et al., 2021). The latter also while avoiding dynamic obstacles, which is also addressed by the work of Viswanathan Sangeetha et al. (2021). There are plenty of more nature-inspired approaches. To avoid extending too much, some of them are just indicated in Table 2.2. Besides, there are also cases in which two models are combined (Saraswathi et al., 2018).

Artificial Intelligence

Soft Computing algorithms use other sets of configurable operators like fuzzy rules or neural networks to generate a path. The first is based on complying with sets of if-then rules to dynamically let the robot know what to do. The second is based on the use of interconnected artificial neurons that work with configurable weighted connections. Both fuzzy rules and neural networks could be considered for performing global and local planning (H.-y. Zhang et al., 2018). In general, the most extended choice for them is to act as local planners (Patle et al., 2019; S. Campbell et al., 2020). This is because both fuzzy rules and neural networks serve to steer the robot in highly dynamic and unstructured environments (Zavlangas et al., 2003; Engedy et al., 2010; Yan et al., 2016; S. Campbell et al., 2020). Besides, they can make the robot

constantly replan in fully unknown scenarios (Mac et al., 2016; Engedy et al., 2010). Nevertheless, those approaches based on fuzzy rules may make the robot get stuck in U-shaped obstacles in a similar way as Reactive Manoeuvre algorithms (M. Wang et al., 2005). Besides, neural networks take a relatively high computational time, the provided solution is sub-optimal (Zafar et al., 2018; Y. Zhang et al., 2013) and they are non-complete, as they may not converge to the solution (Mac et al., 2016). Other works in the literature combine fuzzy logic and neural networks especially to build up global-local navigation solutions (Zhu et al., 2009; Shi et al., 2009; Joshi et al., 2011; Prases K Mohanty et al., 2014; H. Wang et al., 2018).

Furthermore, the use of Reinforcement Learning (RL) is also studied to control the motion of a robot (Blum et al., 2020; Yu et al., 2021), even in dynamic situations (Vagale et al., 2021). Yet, according to the extensive review made by H. Sun et al. (2021), this kind of algorithm needs to be further studied in real outdoors environments (H. Sun et al., 2021). These algorithms still suffer from generalization (taking what is learnt from one environment to navigate in a different one) (H. Sun et al., 2021).

2.6 C-Space Search Based Path Planning Algorithms

Configuration space (or C-Space) Search algorithms consider the working space of the path planner as the space of all states or configurations that are reachable by the robot. For this reason, most of the works in this category refer to this working space as the C-Space. The main idea behind these algorithms is to use a discrete set of samples that are part of this C-Space. In other words, the C-Space is discretized. This set of samples includes the initial and the goal states, or at least samples relatively close to them. In this way, these algorithms execute a search operation visiting samples from this set. At a certain point, the algorithm will find and return a certain subset of samples connecting the initial and goal states: the resulting path. In other words, the waypoints forming the paths correspond, each of them, to a sample from the C-Space. This implies the generated path heavily depends on how these samples are scattered, how they are connected and how are they visited. In fact, due to this dependency, in some approaches post-processing is done to smooth the shape of the resulting path.

The C-Space Search category is subdivided into two groups of algorithms according to how do they discretize the C-Space. Graph Search algorithms do this using a pre-existent graph (as one of those depicted in Figure 2.1). Each of the nodes from this graph represents a C-Space sample and is connected to other nearby nodes, i.e., its neighbours. With regards to Sampling Based algorithms, they focus on the creation and/or modification of samples within the C-Space in an iterative way. They can keep working even after finding a feasible path to find better ones. Therefore, they must be stopped at some point, for example by using a time limit.

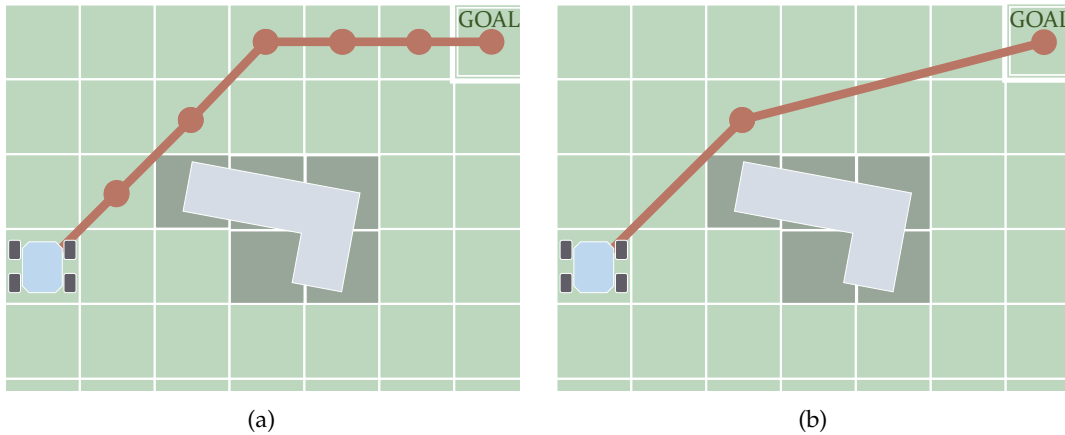


FIGURE 2.8: Main difference between the paths (in red) produced by Edge-restricted (a) and Any-angle algorithms (b): in the first case, waypoints can be placed only on consecutive (neighbouring) nodes.

Graph Search

As mentioned, the C-Space can be discretized in the form of a graph. Graph Search algorithms visit fully or partially this graph until they find a path connecting the initial and goal states. Therefore, they need this graph to be defined beforehand (Souissi et al., 2013; Nash and Koenig, 2013). The path they return is formed by waypoints that are placed on top of grid nodes/samples. These paths consequently depend on how the graph is structured. As seen in Section 2.2 there exist various graph structures in the form of Cell Decomposition and Roadmaps. According to whether these nodes, on which the waypoints are placed, must strictly be neighbours or not, the Graph Search algorithms can be subdivided into two subcategories: Edge-restricted and Any-angle.

- **Edge-restricted.** Algorithms made in this category return paths whose waypoints are placed on top of neighbouring samples. In other words, the connections between consecutive waypoints of the path are coincident with graph edges. Figure 2.8a depicts a schematic showing how in the first case the shape of the path is determined by these edges. The most known and basic Edge-restricted path planner is the Dijkstra algorithm (Dijkstra, 1959). It visits nodes in an iterative from one of interest (the goal or the start) until it reaches another if possible, as it is complete. During each visit, it propagates an estimation of the minimum amount of cost from the starting node. Years later, Hart et al. (1968) implemented a heuristic version, A^* , to speed up computation. It is the algorithm from which many variants, both Edge-restricted and Any-angle, are created, as indicated in the scheme depicted in Figure 2.9. It is not compatible with dynamic environments that contain moving elements (González et al., 2015; H. Sun et al., 2021), which applies to those variants in general. Dynamic A^* (D^*), was introduced by Stentz (1994) as an incremental version of A^* . Being incremental means this algorithm recycles previous computation to replan.

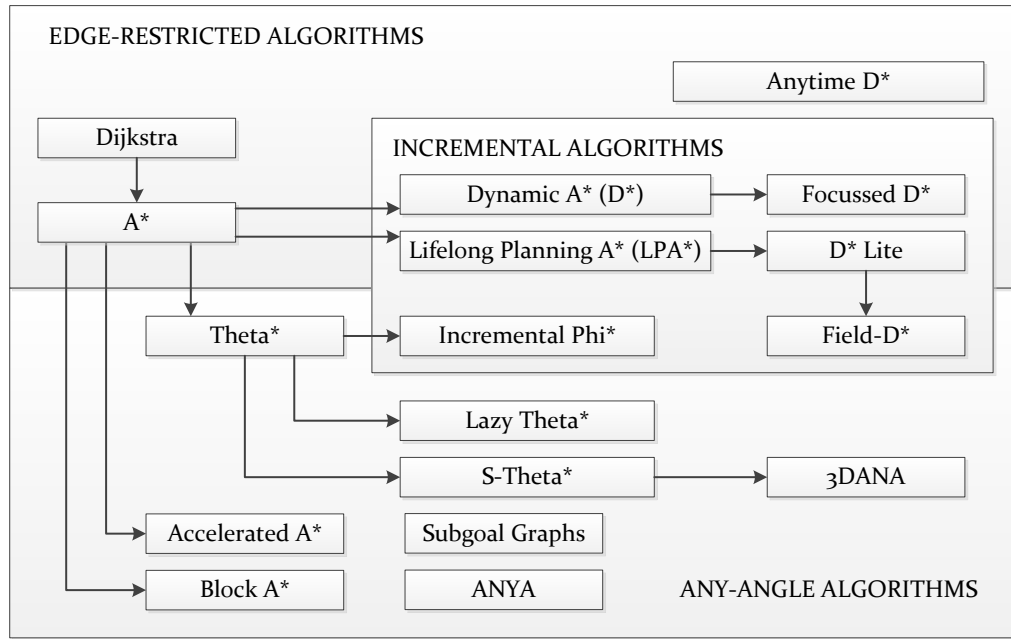


FIGURE 2.9: Overview of many different approaches to Graph Search path planning. The arrows indicate how most of them rest on older approaches yet introduce significant improvements.

Other incremental versions include Focussed D* (Stentz et al., 1995), Lifelong Planning A* (LPA*) (Koenig and Likhachev, 2002b) and D*-Lite (Koenig and Likhachev, 2002a; Koenig, Likhachev, and Furcy, 2004). Since A* and D* algorithms, including their variants, make use of heuristic functions, the optimality of the resulting paths is compromised, i.e. they are sub-optimal. Likhachev, Ferguson, et al. (2008) proposed anytime versions of these algorithms, which return the best path found given a limited time. Dolgov et al. (2010) proposed a version of A*, Hybrid-A*, that prioritizes the feasibility of the resulting paths in exchange for losing optimality and completeness. This algorithm rearranges the nodes after a path is found, in a way this path becomes kinematically feasible.

- **Any-angle.** These algorithms produce paths whose waypoints are placed in nodes that do not necessarily have to be neighbours. In this way, the orientations from waypoint to waypoint are not restricted to the grid morphology. Figure 2.8b depicts an example of this. Instead of sticking to orientations of $0, \pm 45, \pm 90, \pm 135$ or 180 degrees, the orientation from the middle to the goal waypoint is different. One of the first Any-angle algorithms was Field-D* (Ferguson et al., 2005). It is a well-known algorithm mainly due to its use on Mars NASA rovers since Spirit and Opportunity (Carsten et al., 2009). As similar to D* and D*-lite, it is an incremental algorithm, so it recycles previous computation in subsequent executions to replan. Theta* was introduced

by Nash, Daniel, et al. (2007) to find paths that avoided obstacles while reducing the sharp heading changes that could be produced by Field-D* (Daniel et al., 2010). Since these algorithms make use of heuristics they cannot ensure the produced path is the globally optimal one. For this reason there are many existing variants that improve the results of the previous one: Lazy-Theta* (Nash, Koenig, and Tovey, 2010), Accelerated A* (Šišlák et al., 2009), Block-A* (Yap et al., 2011) and more. Other approaches minimize heading changes, like S-Theta* (Muñoz and R-Moreno, 2012) and 3DANA (Muñoz, R-Moreno, and Castaño, 2016; Muñoz, R-Moreno, and Castaño, 2017), or introduce the replanning capability (Nash, Koenig, and Likhachev, 2009).

Sampling Based

Sampling Based based path planning algorithms create samples of the C-Space one after another, following different policies (Karaman et al., 2011; Elbanhawi et al., 2014). Therefore, they do not need a preliminary graph as it is constructed by them. Later on, they retrieve the path from the created samples after meeting a certain condition or set of conditions, like reaching a time limit. This kind of algorithm is asymptotically optimal. It means they can create more and more samples, attempting to find a better solution as time is running. In general, these algorithms are usually used for search in high-dimensional spaces. However, the number of samples may be relatively large to get close to the global optimal solution (Noreen et al., 2016), demanding the use of many memory resources to store all the samples as a consequence.

If only two points are considered (e.g. the starting position and the goal), the algorithm is single-query, while if more points are selected for the same environment then the algorithm is categorized as multiple-query. With regards to the single-query, one of the most famous is Rapidly Random Tree (RRT), which is also a special case of the Rapidly Deterministic Tree (RDT) (LaValle, 2006). This algorithm emulates a tree growing in the sense that from a starting point the samples are dynamically created as if they were branches. As this process is stochastic, this kind of algorithm is considered non-deterministic (Souissi et al., 2013). Figure 2.10a depicts an scheme summarizing this process. When one of the samples is closer to the goal than a certain distance then the path can be retrieved by tracking backwards until reaching the origin point. As mentioned, more iterations can be still executed to find better paths. Further modifications of RRT can be found in the literature, such as a bi-directional version (Kuffner et al., 2000). Most of them introduce improvements in the sampling operation to speed up the convergence, such as the RRT# (Arslan et al., 2013) and the Heuristic Rapidly Random Tree (RRT*). The latter has in turn numerous variants (Noreen et al., 2016) that improve results even further, like the Informed-RRT* (Gammell et al., 2014), the Batch Informed Trees (BIT*) (Gammell et al., 2015) and the Advanced Batch Informed Trees (ABIT*) (Strub et al., 2020a; Strub et al., 2020b).

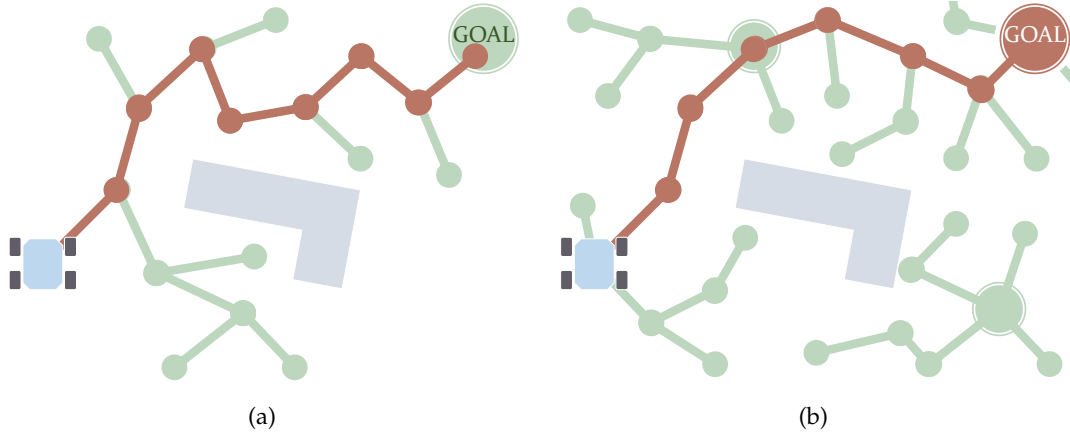


FIGURE 2.10: Example cases of single-query (a) and multiple-query (b) Sampling Based algorithms. Samples are created in an iterative way until the destination is reached. They can still create more to improve the quality of the path.

With regards to the multiple-query Sampling Based algorithms, the most famous among them is the Probabilistic Roadmap Method (PRM) (Kavraki et al., 1996). This algorithm starts with a series of samples already scattered over the C-Space. From here, new samples are created, creating a new tree from each of these initial samples. Figure 2.10b depicts a graphical explanation of this process. Thereafter, PRM depends on the use of a Graph Search method such as A* to retrieve the path from the generated graph. Other approaches improved the performance of PRM by means of heuristics (Karaman et al., 2011; Park et al., 2018). Furthermore, another Sampling Based algorithm, named the Fast Marching Tree (FMT*), was created to reduce the convergence rate of both RRT and PRM. It takes features from not only the two of them but also an Optimal Control algorithm called FMM, which will be detailed later. The main objective of FMT* is to find paths avoiding obstacles in problems involving a high number of degrees of freedom. One example of this is the motion planning of an articulated vehicle presented by Reid et al. (2019). Ichter et al. (2017) proposed the use of Group Marching Tree (GMT*), a similar algorithm to FMT* but focusing on speeding up computation with parallelization using GPUs. Finally, it is worth mentioning there are path planning algorithms that combine the Sampling Based approach with Model Predictive Control (MPC) techniques to account for kinodynamic constraints (Dunlap et al., 2010; L.-L. Wang et al., 2020; Gruning et al., 2020).

2.7 Optimal Control Based Path Planning Algorithms

The baseline of Optimal Control based algorithms is the creation of a control function that takes the robot from an initial state in the C-Space to the destination. As the name suggests, here the path planning problem is addressed using an optimal control approach (Tonon et al., 2017). The main difference with Soft Computing

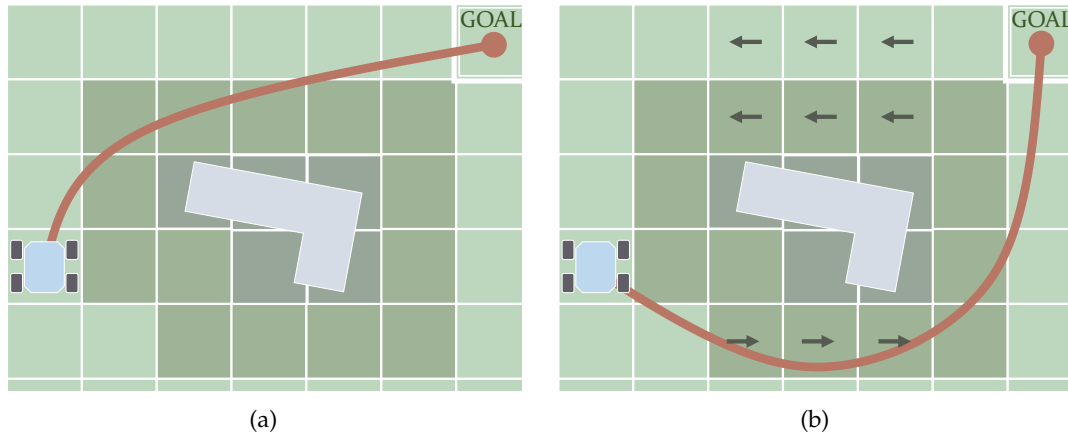


FIGURE 2.11: PDE Solving based algorithms can calculate a continuous and smooth path in non-uniform cost maps. The cost assigned to each cell can be a scalar value or isotropic (a) or in the form of a vector, i.e. anisotropic (b).

methods is that there are no configurable parameters, the problem here must be fully enclosed to be deterministic. Besides, the resulting paths are globally optimal given the formulation of the problem. There are two different subcategories of Optimal Control algorithms. The first of them, PDE Solving, solve a Partial Derivative Equation (PDE) on a grid, based on the Dynamic Programming Principle (DPP) (Bellman, 1966). The second subcategory, Global Optimization, encompasses algorithms that in general optimize a pre-existing path given the kino-dynamic restrictions of the robot to make it feasible.

PDE solving based

The optimal control approach is based here on the Dynamic Programming Principle through the resolution of the static Hamilton-Jacobi-Bellman (HJB) equation (Festa et al., 2017) using a grid. As it is static, dynamic environments are not addressed here. Since it is a Partial Derivative Equation (PDE), this sub-category is named PDE Solving. It can be seen as finding the numerical solution to the problem of calculating the propagation of a wave over a grid. A value of the wave arrival time is assigned to each of the grid nodes. The way the wave propagates will depend on how the HJB equation is formulated, including the cost function. The main drawback of this kind of algorithm is that it generally cannot deal with constraints in the form of discontinuities. For this reason, their use to plan paths for non-holonomic vehicles is limited, and, to the author knowledge, they are mostly used for planning at a global scale in an analogous way to how Graph Search methods are used.

A particular case of the HJB equation is the eikonal equation. In a few words, this equation uses cost that depends only with position and return a scalar value. Fast methods serve to solve the eikonal on a non-uniform costmap with a computational complexity similar to that of Graph Search methods (Gómez et al., 2019). The

main difference is that Fast methods produce a smooth, continuous and globally optimal path as a result. One of the most known is the Fast Marching Method (FMM), introduced by James Albert Sethian (1999). This algorithm follows the same strategy as Dijkstra to visit the nodes of a grid. Unlike Dijkstra, FMM assigns the value of the minimal amount of cost to reach each node by solving the eikonal equation. Chiang et al. (2007) compare this algorithm with A* and demonstrates how, as the path is not restricted by the grid, FMM gets shorter paths. There are many existing research works using FMM for global path planning (Kimmel et al., 2001; Garrido, Malfaz, et al., 2013; Garrido, Álvarez, et al., 2016; Y. Liu, W. Liu, et al., 2017). Some of its variants, most of them reducing the computational power requirements of the FMM, are introduced by the review of Gómez et al. (2019). Kinematic constraints are not present in the eikonal formulation, but some alternative solutions were proposed to deal with them. For example, Y. Liu and Bucknall (2016) presented a way to modify the cost around the initial position to consider the initial orientation of the vehicle. Petres et al. (2005) demonstrated how tuning repulsive potential fields around the obstacles affected the curvature radius of the path. This is the case depicted in Figure 2.11a: the obtained path smoothly gets far from the obstacle with a certain curvature thanks to the high values of cost assigned to the nodes close to it (the darker the colour, the higher the cost). Valero-Gomez et al. (2013) proposed the use of the FMM to create these potential fields prior to finding the optimal path. There is also research that is oriented to propose incremental versions of FMM (with replanning), such as E* (Philippsen, 2007; Philippsen et al., 2008; B. Xu et al., 2013), but they either use heuristic methods that may compromise the optimality of the solution or do not make a significant gain in computational processing.

To work with more general expressions of the HJB equation other kinds of methods must be used. The FMM produces sub-optimal results if used with direction-dependent (anisotropic) cost (James A Sethian and Vladimirsky, 2003). This cost is vectorial in the sense it returns a different value according to the direction of the robot. There are particular situations in which FMM produces accurate results under a certain level of anisotropy, like having a cost function formulated in a way it varies mostly in the directions parallel to the reference axes (Petres et al., 2005; J. Xu et al., 2019). This is the case depicted in Figure 2.11b. James A Sethian and Vladimirsky (2003) proposed the use of an algorithm called the Ordered Upwind Method (OUM) to deal with the static HJB equation, whose convergence rate is demonstrated by Shum et al. (2016). Its main drawback is the increase in computational cost it entails, proportional to the anisotropy existing in the scenario. Shum et al. (2015) used HJB for anisotropic path planning considering the stability of the robot on slopes. Furthermore, the Fast Sweeping Method (FSM) is a Fast method that was also demonstrated to work with general static HJB equations (Kao et al., 2005). It works by repeatedly visiting all nodes on a grid in certain directions, demanding a high number of iterations. Takei et al. (2013) used FSM to formulate the HJB equation to comply with turning radius constraints while avoiding obstacles. For the eikonal case, Bak

et al. (2010) introduced an improvement to FSM to speed up its computation when the cost varies too much, and Detrixhe et al. (2013) introduced a parallel version. Jeong et al. (2008) proposed an algorithm named the Fast Iterative Method (FIM) to solve the eikonal equation also on parallel architectures.

Global Optimization

This subcategory contains path planning algorithms that optimize an existing preliminary feasible path. Unlike Local Optimization methods, introduced in Section 2.4, Global Optimization methods make the resulting path globally optimal in exchange of investing more computational load. For this reason, they are mostly used for global offline planning. The approach presented by Ratliff et al. (2009) for instance uses Sampling Based methods like RRT or PRM as a first step. The second step consists of using gradient optimization techniques to approximate the optimal solution from this feasible path. Van Den Berg et al. (2017) also started with a trajectory computed using RRT, to later apply on it an optimization process based on Differential Dynamic Programming (DP). Plonski et al. (2013) used DP to calculate a path in a solar map that dynamically changes, considering the robot harvests solar energy and also addressing replanning. According to (H. Zhang et al., 2020), DP does not ensure completeness. Ajanović et al. (2018) combined DP with Model Predictive Control (MPC) to calculate energy minimizing paths. Other techniques include the bang-bang approach (Kalmár-Nagy et al., 2004), Mixed-Integer Linear Programming (MILP) (Ma et al., 2006) and Linear Quadratic Regulator (LQR) methods (W. Sun et al., 2016). Finally, a remarkable approach was proposed by Kogan et al. (2006), who used nonlinear optimization to plan time-optimal paths with a length between 20 and 70 metres.

2.8 Summary and Conclusions

This chapter introduced an overview of the majority of existing path planning approaches for mobile surface robots. A general classification makes any path planning algorithm fall into one out of four categories (as previously depicted in Figure 2.4): Reactive Computing, Soft Computing, C-Space Search and Optimal Control based. This classification is based on how each approach fundamentally works to generate a path. According to this way of functioning an algorithm will be suitable or not given how the path planning problem is formulated. Table 2.3 summarizes the main features of each path planning category according to the classification. It analyzes if the algorithms require a preliminary model of the environment, they are deterministic (they always provide the same solution given the same initial conditions), they can tackle with dynamic environments and replan, they are optimal, they are complete (they always return a path if it is feasible), at which planning scale are they commonly used and if they can work in dynamic environments (e.g. with moving elements). For instance, the reach of the planner and the replanning capability, i.e.

TABLE 2.3: Overview of characteristics of the presented path planning algorithms.

Category	Sub-category	Preliminary Map Model	Deterministic	Replanning	Optimality	Completeness	Planning Scale	Dynamic Environment
Reactive Computing	Reactive Manoeuvre	No	Yes	Yes	Sub-optimal	Not ensured	Local	Yes
	Local Optimization	Depend on planner used in first step	Yes	Yes	Sub-optimal	Not ensured	Local	Yes
Soft Computing	Evolutionary	Only ACO	No	Yes	Sub-optimal	Depends (e.g. GA no, PSO, yes)	Global	Yes
	Artificial Intelligence	Depends	No	Yes	Sub-optimal	Not ensured	Both	Yes
C-Space Search	Graph Search	Yes	Yes	Only incremental versions	Restricted (suboptimal with heuristics)	Yes	Global	No
	Sampling Based	No	No	Yes	Asymptotical (Globally optimal after a time)	Probabilistic	Both	Yes
Optimal Control	PDE Solving	Yes	Yes	Very rare	Globally optimal	Yes	Global	No
	Global Optimization	Depend on planner used in first step	Yes	Yes	Globally optimal	No	Global	Yes

capability to deal with updates in the environment information, will determine if it is more suitable for local planning or global planning. Local planning usually requires fast online computation and this reactivity behaviour to plan new paths in the presence of environment data changes. Global planning can even be computed offline and aims to generate paths for long traverses, having a static initial environment available.

Reactive Computing based algorithms seem suitable for local obstacle avoidance path planning as they are easy and cheap to implement. Special attention must be given to avoid falling into a local minimum. Soft Computing algorithms produce a path using multiple configurable operators, which can be inspired by nature or be based on fuzzy rules and/or neural networks. They are suitable for problems involving a large number of variables or difficult to model, such as in highly dynamic environments. With scenarios containing moving elements, in long-range (global path planning) the use of Evolutionary methods is adequate. Latest Artificial Intelligence methods including Deep Learning and Reinforcement Learning still need to be further studied to get solid conclusions, as also remarked by H. Sun et al. (2021). C-Space Search algorithms make use of samples to represent the different configurations of the robot. These samples can be provided beforehand in the form of a graph or be dynamically created. Graph Search algorithms are suitable for global path planning considering advanced graphs such as visibility graphs or space-lattice graphs, at the expense of investing time into building them (something that is admissible for offline planning). Nevertheless, it scales poorly with problems of high dimensions, which justifies the use of Sampling Based algorithms instead.

Sampling Based algorithms have also proved useful for this kind of manoeuvres and problems with a high number of dimensions. Optimal Control algorithms are outstanding for getting globally optimal results. PDE Solving based depend heavily on the formulated PDE, while Global Optimization based algorithms have to start with an already defined path.

PDE Solving based methods present good quality in the results they produce, together with relatively fast execution in 2D maps (low dimensional format) and the flexibility in the formulation of the PDE. Besides, features such as being deterministic and complete are desirable to have a reliable and understandable system onboard any rover. The FMM is chosen to perform global planning considering the cost associated with the energetic performance of a reconfigurable wheeled robot. This makes up the first thesis contribution. Although Graph Search algorithms are deterministic and complete as well, the optimality of the results is restricted by the morphology of the graph, while the FMM ensures the resulting path is globally optimal and is not constrained by such restriction. The missing replanning feature is solved by the second contribution, which uses FM* in a multi-resolution grid to locally repair a preliminary path thanks to its faster computation. Finally, the OUM is chosen to solve path planning on inclined terrains as it complies with the use of anisotropic cost functions, unlike the FMM. The third contribution of this thesis rests on this anisotropic path planning application.

Chapter 3

Isotropic Optimal Path Planning for Reconfigurable Rovers

"A wise man adapts himself to
circumstances, so as the water shapes
itself to the vessel that contains it."

Proverb

3.1 Introduction

The locomotion subsystem is a key part of any ground mobile robot. It influences the way the system drives in outdoors scenarios. Hence, it determines whether the robot is suitable or not to overcome different kinds of terrains. To arrive at a certain place, the chances of finding a preliminary workable path increase with the number of surfaces the robot can adapt to. Besides, a path planner could find more energy (and/or time) efficient paths based on the knowledge of its locomotion. This is because a refined locomotion would make a robot drive more types of terrains, reducing the energy or time invested in it. In other words, the adaptability of the robot affects the efficiency and effectiveness of the planned traverses. It takes hence even more relevance in scenarios that present many kinds of terrain features. A way to improve this adaptability is by providing the robot with the capability to reconfigure its locomotion. Thanks to this feature, the robot can switch between different locomotion modes. Each of them propels the vehicle according to how the motors actuate. In exchange, a higher number of actuators is usually needed. This may be inconvenient, as it increases the price of the robot. Besides, the higher the number of motors the higher the risk one of them fails. Still, only relying on one locomotion mode entails certain limitations. Certain terrains could be difficult or even unfeasible to traverse. Having available various locomotion modes can solve this issue. Each mode can compensate for the flaws of others. In this way, the robot can use the most convenient mode on every occasion. Even so, the robot still requires the use of a path planner that acknowledges this advantage.

The planner in question is assumed to work based on information describing the environment. This information must show in detail the distribution of the different terrains so the planner can decide where the resulting path goes. Having this information *a priori* is possible thanks to the use of external perception systems. For example, drones can be used to map the scenario from above. This makes them access areas faster than ground agents, performing initial recognisance tasks. In this way, these ground agents have available information in advance about the terrain they must face. In the case of planetary (or other celestial bodies) exploration, the use of satellites can carry out this kind of recognisance task.

This chapter elaborates on the first contribution of this thesis. This contribution consists of the creation and use of a cost function based on the models of two locomotion modes. The aim of this cost function is to be used by a path planner so it can acknowledge the adaptability feature of reconfigurable robots. In particular, these two locomotion modes are Normal-driving and Wheel-walking. They are two of the locomotion modes the *ExoMars* rover *Rosalind Franklin* is capable to execute, as indicated in Chapter 1. Here, Section 3.2 presents and details dynamic models of both modes. Thereafter, the cost function in question will be used by the FMM, an algorithm already introduced in Chapter 2. This algorithm falls into the Optimal Control category, being a PDE solving based planner. Here in this chapter this algorithm is fully explained, recalling its functioning in Section 3.3. Moreover, it is clarified which are the unavoidable constraints that arise when modelling a cost function compatible with this algorithm. As a result of using this approach, the path planner will select the most appropriate locomotion mode given any terrain, minimizing the energy required to reach a destination. Section 3.4 summarizes the contents of this chapter and presents some conclusions extracted out of its contents.

3.2 Reconfigurable Locomotion Cost Model

A PDE solving algorithm generates optimal paths given a cost function defined over a grid $\tilde{\Omega}$. This means the cost is defined according to the location of every grid node. The nature of this cost determines in which sense the paths are *optimal*. For instance, optimization criteria can be based on energy minimization. In such a case, the cost has to be built taking into account the energy consumption of the robot at the location of the grid nodes. Assuming the grid is two-dimensional and regular, the indexes i and j refer to the coordinates of a node $\tilde{x}_{ij} \in \tilde{\Omega}$ in each perpendicular axis. The images presented in Figure 3.1 graphically portrait the functioning of the i and j indexation in the two-dimensional XY-plane together with the resolution of the grid Λ . There are two options: square (see Figure 3.1a) and hexagonal (see Figure 3.1b) grids. For both of them is defined $neigh(\tilde{x}_{ij})$, which is the neighbourhood of any node \tilde{x}_{ij} . This neighbourhood, defined in Equation (3.1) for both cases, is the set that comprises the surrounding nodes that are closest to \tilde{x}_{ij} . For the case of the square grid, this set is made up by four nodes, which is also referred to as the von

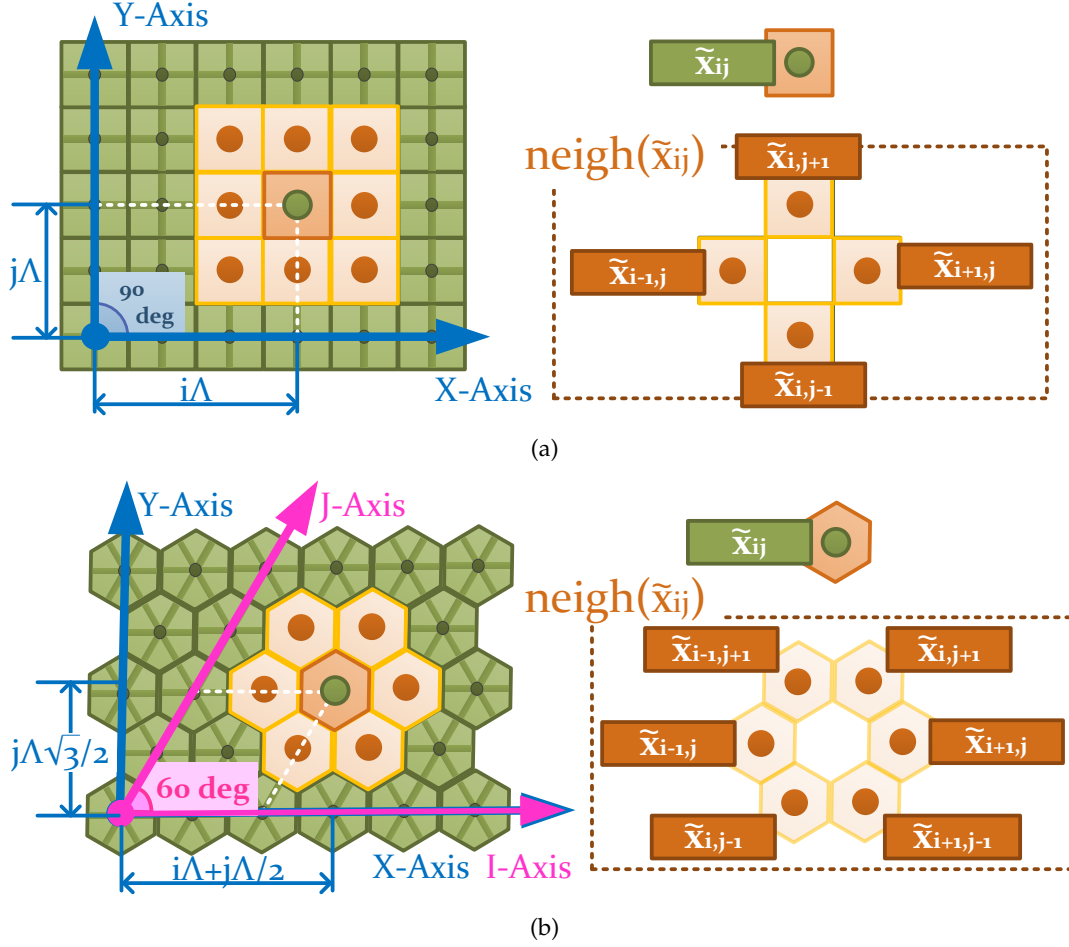


FIGURE 3.1: Nodes indexation and neighbourhood in square (a) and hexagonal (b) grids.

Neumann neighbourhood. In the hexagonal grid, every node has a neighbourhood of six surrounding nodes.

$$neigh(\tilde{x}_{ij}) = \begin{cases} \{ \tilde{x}_{i+1,j}, \tilde{x}_{i-1,j}, \tilde{x}_{i,j+1}, \tilde{x}_{i,j-1} \}, & \text{if grid is square} \\ \{ \tilde{x}_{i+1,j}, \tilde{x}_{i,j+1}, \tilde{x}_{i-1,j+1}, \tilde{x}_{i-1,j}, \tilde{x}_{i,j-1}, \tilde{x}_{i+1,j-1} \}, & \text{if grid is hexagonal} \end{cases} \quad (3.1)$$

An estimation of the power consumption while driving a terrain must account for the terramechanics involved in the traverse at the location of any node \tilde{x}_{ij} . In other words, the interaction between the robot and the terrain at \tilde{x}_{ij} must be addressed by means of dynamic models. Following the work of Canudas-de-Wit et al. (2003) and in a similar way to other previous approaches (K. Yoshida et al., 2002), two functions serve to model this interaction: ρ_{ij} and σ_{ij} . On the one hand, $\rho_{ij} : \tilde{x}_{ij} \in \tilde{\Omega} \rightarrow \mathbb{R}_+$, defined in Equation (3.2), is the function that returns the estimated value of specific resistance coefficient at any location \tilde{x}_{ij} within $\tilde{\Omega}$. This value defines the force $F_r(\tilde{x}_{ij})$ that is expected to oppose the motion of the robot. It mainly

appears due to the friction and the wheel-terrain adherence (Wong, 2008). The specific resistance coefficient can be simplified as the ratio between F_r and the normal force applied by the robot to the ground F_N (Benamar et al., 2013; Jesus Morales et al., 2009; Gillespie, 1992). On the other hand, $\sigma_{ij} : \tilde{x}_{ij} \in \tilde{\Omega} \rightarrow \mathbb{R}_+ \in (0, 1)$ is an estimation of the slip ratio. Equation (3.3) shows how this metric indicates the difference between the value of speed that is commanded to the robot actuators v_c and the expected resulting one v_{ij} that serves as reference. In the case of a robot equipped with wheels, the first speed is the one resulting from the product between the wheel radius d_r and the wheel angular speed $\dot{\theta}_w$. Both terramechanic functions, ρ_{ij} and σ_{ij} , are chosen considering they could be estimated by onboard sensors. There exist different methods to perform such estimations (Brooks et al., 2012).

$$\rho_{ij} = \frac{F_r(\tilde{x}_{ij})}{F_N} \quad (3.2)$$

$$\sigma_{ij} = \frac{v_c(\tilde{x}_{ij}) - v_{ij}}{v_c(\tilde{x}_{ij})} = \frac{d_r \dot{\theta}_w(\tilde{x}_{ij}) - v_{ij}}{d_r \dot{\theta}_w(\tilde{x}_{ij})} \quad (3.3)$$

Besides, the inclination of the terrain is also taken into account by using α_{ij} . This is the maximum gradient of a plane tangent to the surface (slope). In other words, α_{ij} is the maximum angle of inclination of the terrain at $\tilde{x}(\alpha_{ij})$ with respect to the horizontal plane. Given the terrain surface defined with an elevation function Z_{ij} , the gradient α_{ij} can be calculated using (3.4).

$$\alpha_{ij} = \arctan (||\nabla Z_{ij}||) \quad (3.4)$$

Using these terramechanic functions, the power consumption function is formulated in (3.5). It returns the value of energy consumption according to the locomotion mode that consumes less according to some terrain conditions. Here, L is the set of locomotion modes the robot can execute, and $P_l(\rho_{ij}, \sigma_{ij}, \alpha_{ij})$ is the power consumption of a locomotion mode $l \in L$.

$$P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij}) = \min_{l \in L} \{P_l(\rho_{ij}, \sigma_{ij}, \alpha_{ij})\} \quad (3.5)$$

The electric charge function can be similarly obtained using (3.6).

$$I(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij}) = \min_{l \in L} \{I_l(\rho_{ij}, \sigma_{ij}, \alpha_{ij})\} \quad (3.6)$$

Provided that the same voltage V is supplied to all the involved electric motors, the equivalence in (3.7) occurs.

$$P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij}) = V I(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij}) \quad (3.7)$$

The particular case of a rover equipped with wheeled legs is tackled here. An example of this kind of rover is the one that will be sent as part of the ESA Exomars

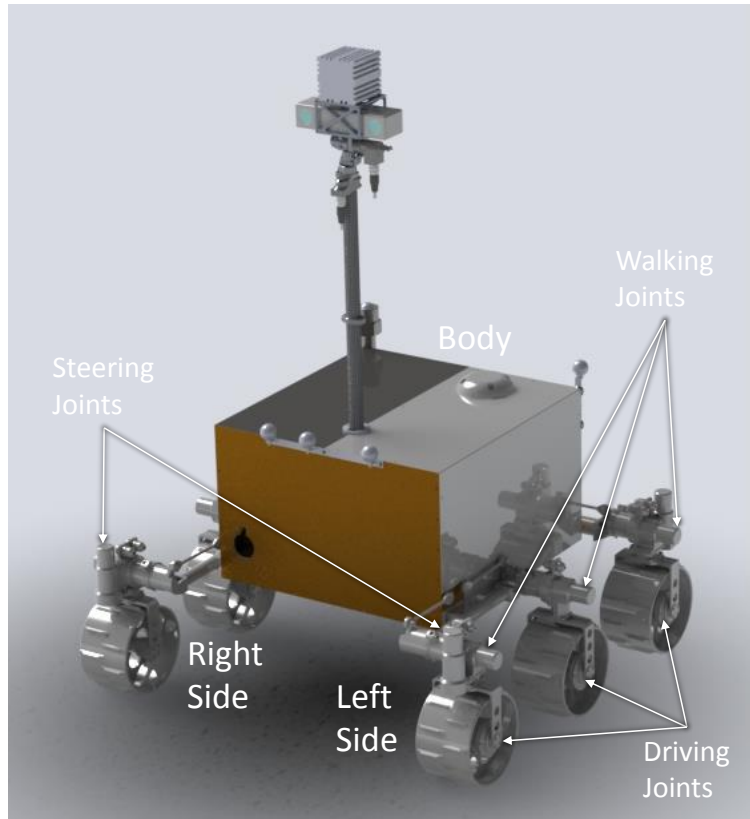
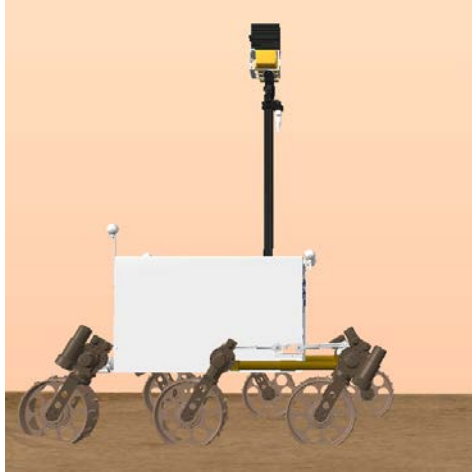


FIGURE 3.2: 3D render image of a rover capable to execute Wheel-walking, with indications of the main parts of its kinematic configuration. Credit: (Pérez-del-Pulgar et al., 2017).

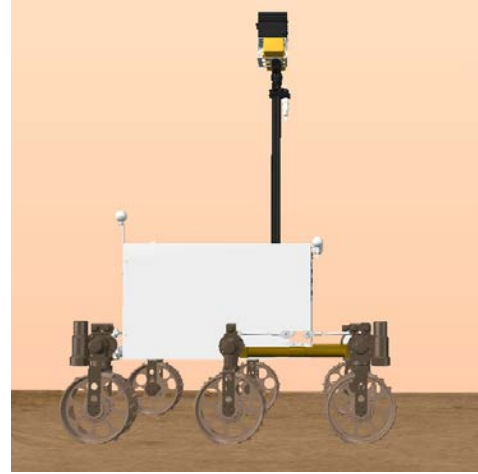
mission (Patel et al., 2010). This rover has a $6 \times 6 \times 6$ drive configuration: 6 driving joints, 6 walking joints and 6 steering joints. Figure 3.2 depicts how these joints can be arranged in a reconfigurable rover. The driving joints make each of them roll one of the robot wheels. The walking joints are located above each wheel and serve to deploy and retrieve each wheeled leg with a motion that resembles a pendulum. The steering joints, as its name suggest, steer the wheels towards a different direction rather than the front. This kinematic configuration allows the execution of two modes: Wheel-walking and Normal-driving. Both of them are later explained in full detail. Hence, $L = \{ww, nd\}$ and $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ returns the least amount of energy consumption between both locomotion modes at the location of \tilde{x}_{ij} . $P_{ww}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})$ and $P_{nd}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})$ correspond to the estimated energy consumption in case of using the Wheel-walking mode and the normal-driving mode respectively. They are built upon the terramechanic parameters ρ_{ij} , σ_{ij} and α_{ij} .

Dynamic Model of Wheel-walking

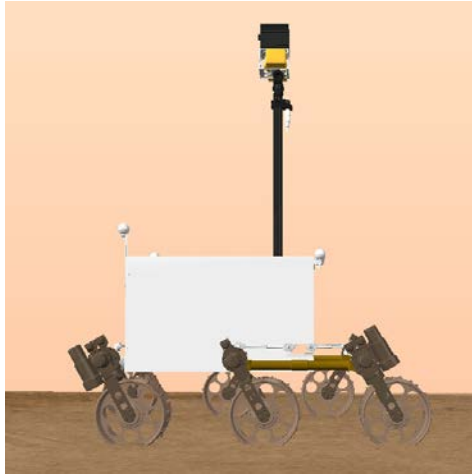
Chapter 2 already introduced the Wheel-walking as a mode of locomotion designed to overcome loose soils (Azkarate, Zwick, et al., 2015; Malenkov et al., 2017). This mode makes use of walking joints (see Figure 3.2). These are revolution joints that are installed, each of them, on top of a bar. This bar connects the robot body to



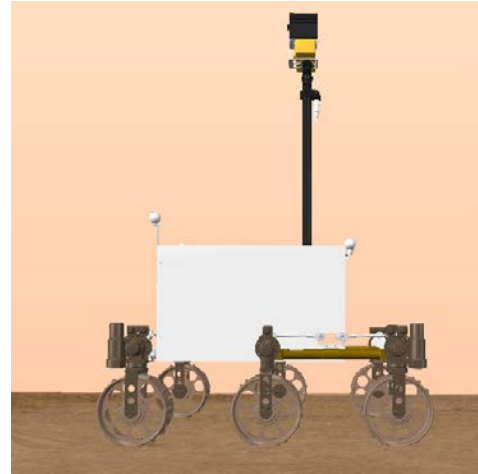
(a) The robot starts the stride of the Wheel-walking mode. Its left side goes forward while its right side goes backwards.



(b) Both sides switch the direction in which they are moving. Left side goes backwards while right side goes forward.



(c) Motion continues following the same direction on both sides. At some point this direction reverses for both sides.



(d) The Wheel-walking mode finishes with all the legs standing perpendicular to the terrain surface.

FIGURE 3.3: Step-by-step depictions of virtual rover performing the Wheel-walking locomotion using the side-by-side gait.

a rolling wheel. Although there exist many possible gaits (Wiese, 2017), only one of them is chosen: the side-by-side gait. Figure 3.3 shows a sequence of images portraying its functioning. The side-by-side gait consists of deploying and retrieving the wheeled legs in two groups, each of them corresponding to one of the sides of the rover. In this way, while the wheeled legs of one side are being deployed (wheel-forward motion), those from the other side are being retrieved (wheel-backward motion). Then, after a deployment it comes the respective retrieval. For simplicity reasons, any reference to the Wheel-walking mode from here on in this thesis refers to this particular gait.

The Wheel-walking locomotion mode is modelled in a simplified way considering only two wheels, referred to with subscripts $w1$ and $w2$. They represent the two sides of the robot, left and right. Figure 3.4 depicts an schematic of this simplified

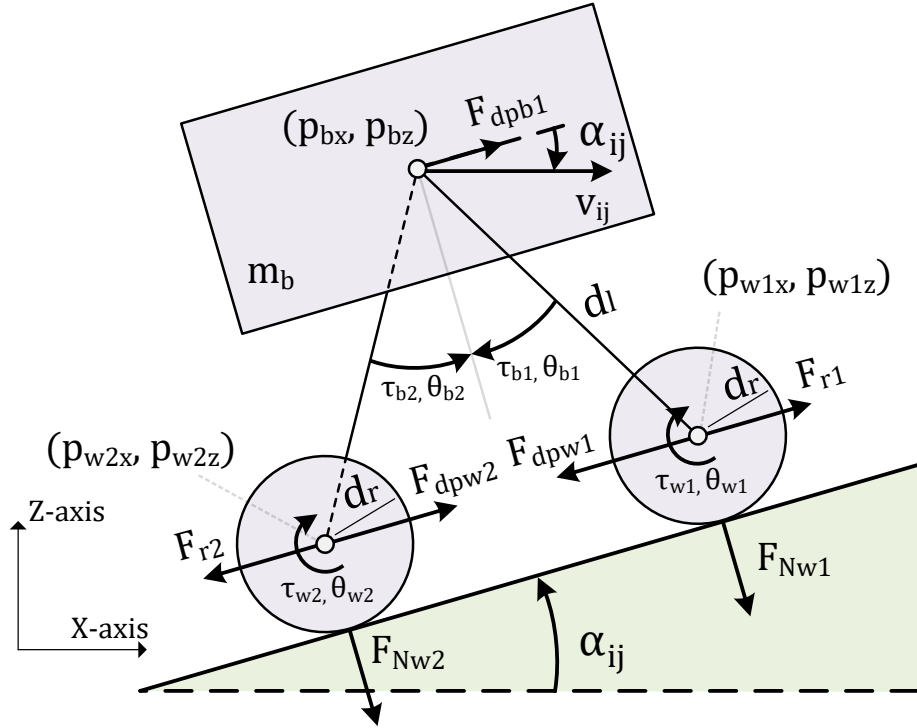


FIGURE 3.4: Schematic of the main forces, torques and position parameters involved in the execution of the Wheel-walking locomotion mode.

version, moving on top of an inclined surface. Here, the model is created considering a local XYZ reference frame. The Z-axis of this local frame points upwards, perpendicular to the XY-plane. The robot moves in the local XZ-plane (or the *sagittal* plane). The first wheel, $w1$, performs the wheel-backward motion, while the second, $w2$, makes a wheel-forward motion. Each of the wheels is connected to the respective robot body joint, indicated with subscripts $b1$ and $b2$, with a bar. Two motors are located at the extremes of each bar: one connecting the body (actuating the walking joint) and another connected to the wheel, making it roll.

The wheel-backward motion, performed by the first wheeled leg with a torque τ_{b1} , serves to pull forward the body of the robot thanks to a lever effect. This makes the corresponding bar rotate a clockwise angle of θ_{b1} . As can be checked in Figure 3.4, the angle θ_{b1} is zero when the bar is perpendicular to the terrain surface. As a result of the bar rotation, the drawbar pull force F_{dpw1} is produced. Another force, the opposing force F_{r1} , opposes to F_{dpw1} as expressed in Equation (3.2). F_{r1} depends on the terrain terramechanics, and if high enough the wheel does not slide backwards and the robot is pushed forward. Equation (3.8) expresses the dynamics underlying this motion modelled with the *d'Alembert* principle (De Canete et al., 2011).

$$J_b \ddot{\theta}_{b1} + B_m \dot{\theta}_{b1} + \frac{m_b}{n} \ddot{p}_{bx} d_l \cos \theta_{b1} + \alpha_{ij} + \frac{m_b}{n} (\ddot{p}_{bz} + g) d_l \sin \theta_{b1} + \alpha_{ij} = \tau_{b1} \quad (3.8)$$

The first two addends of Equation (3.8) represent the motor dynamics, using the moment of inertia J_b and the rotational damper B_m . The rest of addends from the first term correspond to the moment of inertia of the robot body, considering the mass m_b divided by the number of wheels n . The right side of the equation is the torque τ_{b1} , which generates the motion on wheel w_1 by increasing the angle θ_{w1} , i.e. making it roll. The rolling joint of this wheel is assumed to be locked during the operation. Thus, $\tau_{w1} = 0 \text{ N} \cdot \text{m}$ and $\dot{\theta}_{w1} = 0 \text{ rad s}^{-1}$. The resulting drawbar pull force originated on the wheel w_1 from the wheel-backwards movement, F_{dpw1} , is expressed in Equation (3.9), as well as the drawbar pull force on the rover body F_{dpb1} . Here, d_l is the length of this bar as indicated in Figure 3.4. The speed of the robot in the local X-axis, \dot{p}_{bx} , which is in fact its speed v_{ij} within Ω , is expressed in Equation (3.10).

$$F_{dpw1} = \frac{\cos(\theta_{b1})}{d_l} \tau_{b1} \geq F_{r1} = F_{dpb1} \quad (3.9)$$

$$\dot{p}_{bx} = \dot{\theta}_{b1} d_l (\cos_{\theta_{b1}} \cos_{\alpha_{ij}} + \sin_{\theta_{b1}} \sin_{\alpha_{ij}}) \quad (3.10)$$

The other wheel, w_2 performs a wheel-forward movement. It has a contrary direction to the wheel-backwards, but is synchronized with it. This synchronization makes the robot body maintain a linear velocity v_{ij} and be balanced as a consequence. The bar is rotated in a counter-clockwise way, making the wheel w_2 overtake wheel w_1 . The torque τ_{b2} generates this motion while the wheel also rotates thanks to the torque τ_{w2} . The action of both torques produces as a result the drawbar pull force F_{dpw2} on the wheel w_2 , as expressed in Equation (3.11). In this equation the first two addends correspond to the effect of the rotational moment of inertia of the wheel, J_w , and the rotational damper B_m on wheel w_2 . It is assumed that all wheels have the same shape, mass and inertia. The next addend is the linear inertia produced by the wheel mass m_w . The third term is the torque produced by the rolling resistance ρF_{Nw2} . The fourth term models the gravity and the rover body vertical inertia. The motor of the wheel w_2 generates the torque τ_{w2} , affected by the slip ratio σ .

$$J_w \ddot{\theta}_{w2} + B_m \dot{\theta}_{w2} + m_w \dot{v}_{ij} d_r + \rho_{ij} F_{Nw2} d_r + \frac{m_b}{n} d_l \sin_{\theta_{b2} + \alpha_{ij}} (\ddot{p}_{bz} + g \cos_{\alpha_{ij}}) = \tau_{w2} (1 - \sigma_{ij}) + \tau_{b2} \cos_{\theta_{b2} + \alpha_{ij}} \frac{d_r}{d_l} = F_{dpw2} d_r \quad (3.11)$$

The speed of the robot projected on the XY surface is expressed in Equation (3.12). The equality shown in Equation (3.13) is a necessary condition to comply to ensure the robot goes straightforward. the angular velocity of the wheel w_2 , $\dot{\theta}_{w2}$, is calculated as shown in Equation (3.14).

$$\dot{p}_{bx} = v_{ij} = \cos_{\alpha_{ij}} \dot{p}_{w2x} + d_l \dot{\theta}_{b2} \cos_{\theta_{b2} + \alpha_{ij}} \quad (3.12)$$

$$\dot{\theta}_{b1} = -\dot{\theta}_{b2} \quad (3.13)$$

$$\dot{\theta}_{w2} = \frac{2\dot{p}_{bx}}{d_r \cos \alpha_{ij}} \quad (3.14)$$

Finally, current consumption for a step using this locomotion mode can be appreciated in (3.15). In this equation, τ_{wk} and τ_{bk} are the torque values from each wheel joint and walking joint k respectively. K_w is the motor torque constant that relates the current fed to the motor and the wheel joint torque. K_b works in the same way but considering the walking joint torque. V is the voltage fed to the motors. It is assumed V is fixed to the same constant value for all motors. Finally, it is worth mentioning for a multiple wheels rover, wheels 1 to $n/2$ perform the wheel-backward movement and wheels $n/2 + 1$ to n the wheel-forward.

$$P_{ww}(\rho_{ij}, \sigma_{ij}, \alpha_{ij}) = \frac{V}{\Delta t} \left(K_b \sum_{k=1}^n \int_{\Delta t} |\tau_{bk}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})| + K_w \sum_{k=n/2+1}^n \int_{\Delta t} |\tau_{wk}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})| \right) \quad (3.15)$$

Dynamic Model of Normal-driving

Normal-driving is the name used to refer to the locomotion mode usually employed by wheeled mobile robots. It relies on the use of wheels that not only roll but also are steerable. By having steering joints at the frontal and rear wheels (see the virtual model of the robot depicted in Figure 3.2), the vehicle is capable to perform Full-Ackermann manoeuvres as well as turning manoeuvres on the spot or Point Turns. In the case of having steering joints in the middle wheels, the vehicle would be also capable to execute a third mode the Crabbing locomotion. However, this is out of the scope of this thesis. The walking joints are hence here not used in the Normal-driving mode, with all of them locked at $\theta_b = 0$.

The model, as shown in Figure 3.5 is reduced to just one wheel. Under this assumption, the mass of the body m_b is divided by n , the total number of wheels. Figure 3.5 depicts the simplified model of this locomotion mode, where the robot climbs a surface with an inclination angle (slope gradient) α_{ij} . The wheel motor generates a torque τ_w that in turn produces an angular motion θ_w . Taking into consideration the slip ratio (3.3), the drawbar pull force F_{dpwk} that is transmitted from the wheel is calculated using Equation (3.16). The reference velocity v_{ij} in the 2D grid (Global XY axes) is expressed in Equation (3.17). It is the projection on this 2D grid of the linear velocity of the wheel \dot{p}_{wx} , which is also the same projection of the body linear velocity \dot{p}_{bx} since the whole robot moves as a whole. \dot{p}_{bx} is therefore the projection (represented by $\cos \alpha_{ij}$) of the wheel angular speed $\dot{\theta}_w$ multiplied by the wheel radius d_r as well as the slip ratio σ_{ij} .

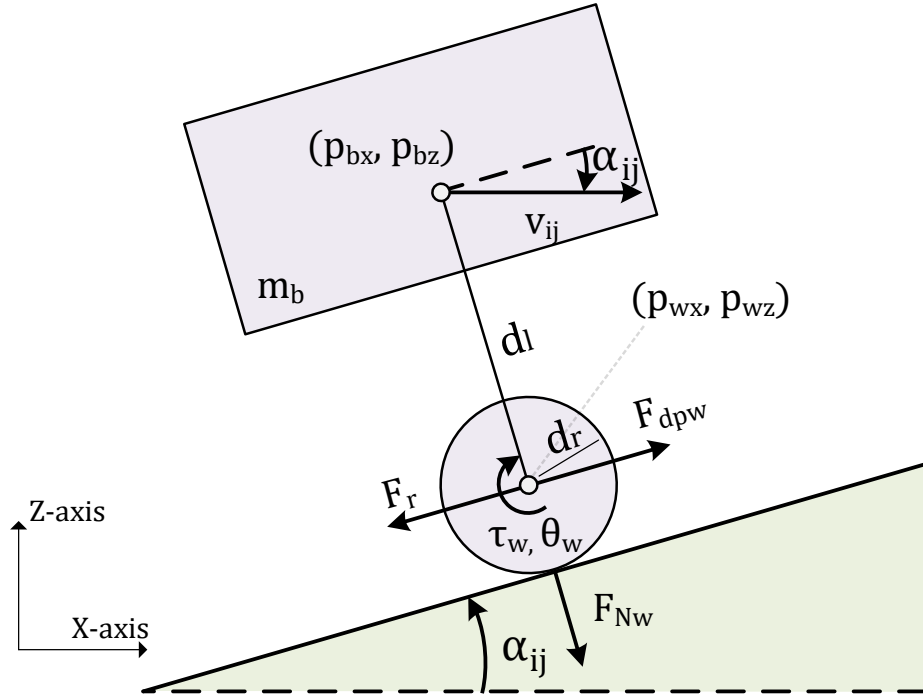


FIGURE 3.5: Schematic of the main forces, torques and position parameters involved in the execution of the Normal-driving locomotion mode.

$$F_{dpw} = (1 - \sigma) \frac{\tau_w}{d_r} \quad (3.16)$$

$$\dot{p}_{wx} = \dot{p}_{bx} = v_{ij} = \cos \alpha_{ij} \dot{\theta}_w d_r (1 - \sigma_{ij}) \quad (3.17)$$

According to D'Alembert principle (De Canete et al., 2011), the linearised Normal-driving dynamics for one wheel is represented as shown in Equation (3.18). The first term is related to the inertia and mass of the robot, affecting its acceleration. The second term corresponds to the effect of the motor rotational damper B_m . The third term is the action of the force that counteracts the drawbar pull force on the wheel. It is defined in Equation (3.19) as the sum of the rolling resistance and the component of the gravity acceleration vector parallel to the surface. The torque τ_w is affected by the slip ratio as well as the term $\cos \alpha_{ij}$, which appears due to projecting the robot velocity onto the two-dimensional XY surface.

$$\left(\frac{J_w}{d_r} + \frac{m_b d_r}{n} \right) \dot{v}_{ij} + \frac{B_m}{d_r} v_{ij} + F_r d_r = \tau_w (1 - \sigma_{ij}) \cos \alpha_{ij} \quad (3.18)$$

$$F_r = \rho F_{Nw} + \frac{m_b}{n} g \sin \alpha_{ij} \quad (3.19)$$

$$F_{Nw} = \frac{m_b}{n} g \cos \alpha_{ij} \quad (3.20)$$

The required instantaneous power consumption is defined in Equation (3.21). Unlike in the Wheel-walking case, the torque of the motor is here relatively constant, omitting any effect due to acceleration or braking. Therefore, the power consumption of the Normal-driving locomotion mode, P_{nd} , is considered to be always constant. P_{nd} results from summing the power of all n wheels, defined by the corresponding values of torque τ_w , previously obtained with Equation (3.18). In (3.21) K_w is the motor constant that translates the supplied current into torque. It is assumed that there is no power consumption from keeping the walking joints to zero, which could be possible using a mechanical brake.

$$P_{nd}(\rho_{ij}, \sigma_{ij}, \alpha_{ij}) = VK_w \sum_{k=1}^n \tau_w(\rho_{ij}, \sigma_{ij}, \alpha_{ij}) \quad (3.21)$$

3.3 Optimal Path Planning using the Fast Marching Method

As mentioned before, it is desired that a path planner acknowledges the different locomotion modes of a robot. In this way, this path planner can produce not only an optimal path but also determine which locomotion mode is better to use at each moment. The optimal path must be the one that minimizes an objective function such as time or energy. This function will depend on the existing terrain features and the available locomotion modes in case of using the power consumption function $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ already introduced in (3.5). This section presents the case of using a cost function that is based exclusively on scalar values. This means that the cost will not depend on any kind of direction, such as the robot heading or the direction of any slope (also called the aspect direction). In this case, the path planning problem is categorized as isotropic.

The Optimal Control based algorithm named FMM can tackle isotropic path planning and return globally optimal paths with ensured completeness, as mentioned in Chapter 2 (see Section 2.7). As a reminder, the latter means that it is guaranteed that the FMM will return the optimal path if such a feasible one exists. Besides, the computational complexity of this method is similar to that of Graph Search algorithms (A^* , D^* , ...), being $O(n_{nodes} \log(n_{nodes}))$ with n_{nodes} as the total number of nodes forming a grid. For these reasons, the FMM is chosen to be adapted in a way it exploits the adaptability of reconfigurable rovers. This algorithm numerically solves a certain PDE: the eikonal equation. The eikonal equation models the rate of propagation and direction of a wave that expands from a certain point within a closed region Ω . Here, this region Ω is two-dimensional, i.e. $\Omega \subset \mathbb{R}^2$, and covers the area of interest that is relevant to the planner. The propagating wave can reach another location and, as a result, the optimal path connecting both positions can be found. The FMM produces a viscous solution of this wave propagation, which means such

solution exists, is unique and is stable. This method uses a discrete grid $\tilde{\Omega}$, which covers Ω and comprises a series of nodes \tilde{x}_{ij} . These nodes are regularly scattered according to the 2D coordinates that are defined by the indexes i and j . Each node is hence placed at a location $\tilde{x}_{ij} \in \tilde{\Omega}$ and is associated with information that is relevant to the planner, such as the cost. This cost, $C(\tilde{x}_{ij})$, affects the wave propagation. The wave accumulates the least possible cost as it expands, making any resulting path the optimal one. The FMM visits each node and calculates its value of accumulated cost, or total cost, using the eikonal equation presented in (3.22). This accumulation of cost is either the amount the wave takes from the origin, $T(\tilde{x}_o, \tilde{x}_{ij})$, or the amount required to reach the goal, $T(\tilde{x}_{ij}, \tilde{x}_g)$, depending on whether the wave starts propagating from the origin or the goal. The FMM calculates the values of the corresponding total cost at each node considering that the rate of propagation of the wave, either $1/||\nabla T(\tilde{x}_o, \tilde{x}_{ij})||$ or $1/||\nabla T(\tilde{x}_{ij}, \tilde{x}_g)||$, is equal to the cost $C(\tilde{x}_{ij})$.

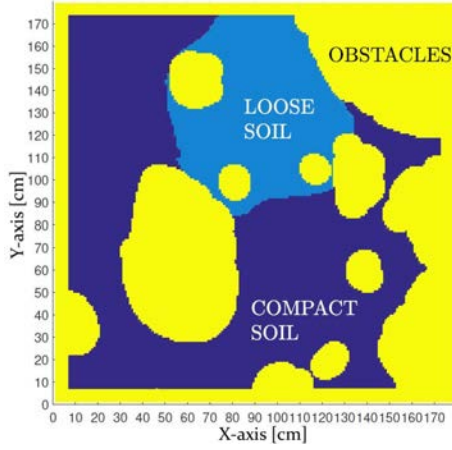
$$||\nabla T(\tilde{x}_o, \tilde{x}_{ij})|| = ||\nabla T(\tilde{x}_{ij}, \tilde{x}_g)|| = C(\tilde{x}_{ij}) \quad \forall \tilde{x}_{ij} \in \tilde{\Omega} \quad (3.22)$$

The rate of propagation increases (producing lower values of total cost as a consequence) if the cost decreases and vice-versa. Figure 3.6 depicts an example case to understand this better. This case partitions Ω into three different kinds of terrain as can be seen in Figure 3.6a. These terrain types are labeled as *Loose Soil*, *Compact Soil* and *Obstacles*. Thus, the grid $\tilde{\Omega}$ contains nodes \tilde{x}_{ij} that fall into one of these three categories. A value of cost is assigned to each node according to not only the type of terrain but also if they are close to obstacles. These nodes that are closer to obstacles are highlighted in Figure 3.6b. The reason for opting to this policy of cost assignation depends on what is the chosen criteria for the path planning problem. A more detailed insight into the criteria for modeling the cost of the path planner is later provided. The key point to understand here is that the cost affects the rate of propagation of the wave, and consequently the resulting path as well. This can be clearly understood by checking Figures 3.6c and 3.6d: by increasing the cost of the *Loose soil* terrain the resulting path is longer. This path circumvents the *Loose soil* area as well as some of the obstacles, as the propagated wave is slowed by them.

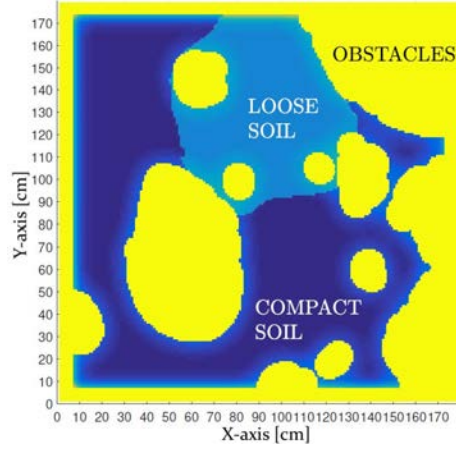
The formulation of the total cost is based on the Dynamic Programming Principle (DPP). The total cost between two points such as the origin \tilde{x}_o and the goal \tilde{x}_g will be the minimum possible. An important condition is that $C(x)$ returns positive non-zero values, i.e. $C : \mathbb{R}^2 \in \Omega \rightarrow \mathbb{R}_+$. Otherwise the FMM would produce undesirable local minimum points and compromise the optimality of the resulting path, making it sub-optimal. For any other point $x \in \Omega$, the total cost from \tilde{x}_o to that point, $T(\tilde{x}_o, x)$, and from that point to \tilde{x}_g , $T(x, \tilde{x}_g)$, is equal to the total cost between \tilde{x}_o and \tilde{x}_g , only if this point x is placed at the optimal path connecting them, $\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$.

$$T(\tilde{x}_o, x) + T(x, \tilde{x}_g) = T(\tilde{x}_o, \tilde{x}_g) \quad \forall x \in \Gamma(\tilde{x}_o, \tilde{x}_g, \cdot) \in \Omega \quad (3.23)$$

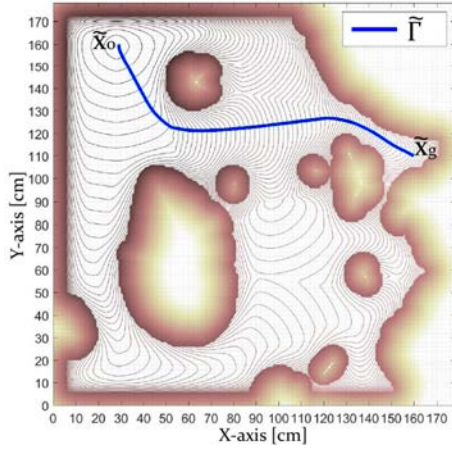
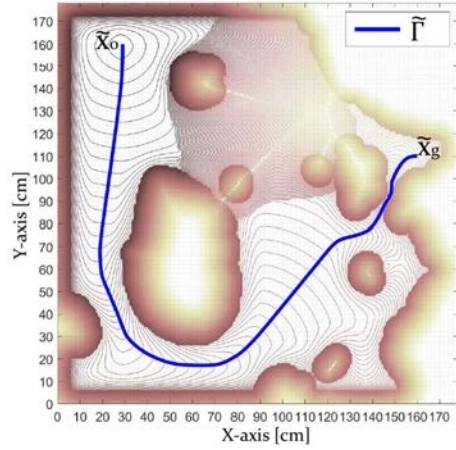
As mentioned, the FMM calculates the values of total cost according to where



(a) Different values of cost are assigned to each kind of terrain.



(b) A penalization cost is added to those nodes that are close to obstacles.

(c) First path planning execution, with the wave starting at \tilde{x}_o .

(d) Second execution with a higher cost in the loose soil.

FIGURE 3.6: An example case in which the FMM is used in a scenario with different kinds of terrain (named 'Compact soil', 'Loose soil' and 'Obstacles' to be representative of a real scenario). The rate at which the wave propagates depends on the assigned values of cost to each terrain. Changing the cost results in changing the wave propagation and, as a consequence, the shape of the optimal path.

the propagating wave starts, either the origin \tilde{x}_o or the goal \tilde{x}_g . This is translated into assigning to one of these nodes a total cost value of zero as the initial condition. From this particular node the values of total cost for the rest of nodes are calculated. Equation (3.24) shows the case in which the propagating wave starts from the origin \tilde{x}_o , hence assigning a value of zero to it ($T(\tilde{x}_o, \tilde{x}_{ij} = \tilde{x}_o) = 0$). This is the approach taken in the case presented in Figure 3.6. Equation (3.25) does the same but for the case in which the wave starts from the goal \tilde{x}_g , having in this way $T(\tilde{x}_{ij} = \tilde{x}_g, \tilde{x}_g) = 0$.

$$T(\tilde{x}_o, \tilde{x}_{ij}) = \min_{\Gamma(\tilde{x}_o, \tilde{x}_g, s) \in \Omega} \left\{ \int_0^{s_{ij}} C(\Gamma(\tilde{x}_o, \tilde{x}_g, s)) ds \right\}, \quad T(\tilde{x}_o, \tilde{x}_{ij} = \tilde{x}_o) = 0 \quad (3.24)$$

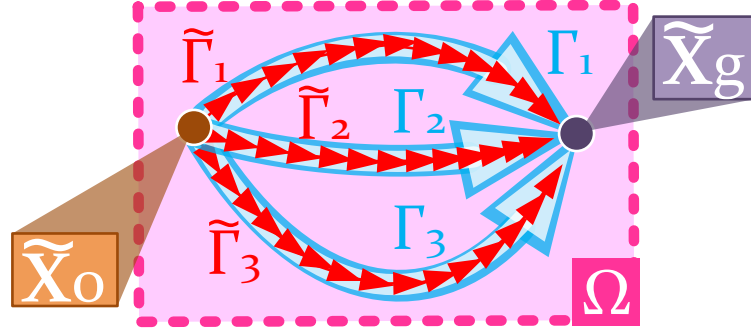


FIGURE 3.7: Different paths $\Gamma(\tilde{x}_o, \tilde{x}_g, s)$ connecting \tilde{x}_o and \tilde{x}_g , all of them located within the 2D closed region Ω . The path planner must find the discretized version $\tilde{\Gamma}$ that approximates one of them: the optimal.

$$T(\tilde{x}_{ij}, \tilde{x}_g) = \min_{\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot) \in \Omega} \left\{ \int_{s_{ij}}^{s_g} C(\Gamma(\tilde{x}_o, \tilde{x}_g, s)) ds \right\}, \quad T(\tilde{x}_{ij} = \tilde{x}_g, \tilde{x}_g) = 0 \quad (3.25)$$

In both equations (3.24) and (3.25) the cost $C(x)$ is integrated along the positions returned by the path function $\Gamma(\tilde{x}_o, \tilde{x}_g, s)$ along the path length s . In the first of these equations, s_{ij} indicates the path length from the origin to \tilde{x}_{ij} . In the second one, s_g is the full length of the path. Figure 3.7 serves to explain how this path is one among infinite possible paths that connect \tilde{x}_o and \tilde{x}_g . The objective of the planner is to find the optimal one that minimizes the total cost in either (3.24) or (3.25), as both cases are equivalent thanks to the definition of the DPP in (3.23). The optimal path $\Gamma(\tilde{x}_o, \tilde{x}_g, s)$ is a continuous function that returns a location $x \in \Omega$ according to the path length s . Here, x is used instead of \tilde{x}_{ij} to remark that the path can be placed anywhere within the region Ω (the 2D region relevant for the planner) and is not restricted to grid nodes $\tilde{x}_{ij} \in \Omega$. In other words, the path does not have to pass through these grid nodes with the exception of \tilde{x}_o and \tilde{x}_g . In this way, $s = 0$ at \tilde{x}_o (where the first waypoint is placed) and $s = s_g$ at \tilde{x}_g (where the last waypoint is placed). To be more precise, the planner must find $\tilde{\Gamma}$, defined in (3.26) as a series of consecutive discrete waypoints that approximates the optimal path $\Gamma(s)$, as also depicted in Figure 3.7. Hence, $\tilde{x}_o = \Gamma(0) \in \tilde{\Gamma}$ and $\tilde{x}_g = \Gamma(s_g) \in \tilde{\Gamma}$. Besides, the waypoints that make up $\tilde{\Gamma}$ can still be located anywhere within Ω as also indicated in (3.26).

$$\tilde{\Gamma} = \{\tilde{\Gamma}_o \in \tilde{\Omega}, \tilde{\Gamma}_1 \in \Omega, \tilde{\Gamma}_2 \in \Omega, \dots, \tilde{\Gamma}_g \in \tilde{\Omega}\} \quad (3.26)$$

The criterion (or criteria) that is chosen to determine in which sense the path planner is optimal settles what the cost and the total cost physically represent. Table 3.1 indicates some possibilities. The first one is the minimization of the traverse duration. In this case, the total cost corresponds to the time at which the propagating wave arrives at the location of any node \tilde{x}_{ij} . This time can be either the minimum

TABLE 3.1: Optimization criteria for optimal isotropic path planning according to how the cost function is defined. Here T is the total cost.

Total cost definition	Cost definition	Cost function	Cost units	T units
Time	Slowness	$C(\tilde{x}_{ij}) = \frac{1}{v_{ij}}$	s m^{-1}	s
Time + Proximity to obstacles	Slowness and collision risk	$C(\tilde{x}_{ij}) = \frac{1}{v_{ij}} + r_{ij}$	s m^{-1}	s
Energy	Energy per meter	$C(\tilde{x}_{ij}) = \frac{P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})}{v_{ij}}$	$\text{W} \cdot \text{s/m}$	$\text{W} \cdot \text{s}$
Energy + Proximity to obstacles	Energy per meter and collision risk	$C(\tilde{x}_{ij}) = \frac{P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})}{v_{ij}} + r_{ij}$	$\text{W} \cdot \text{s/m}$	$\text{W} \cdot \text{s}$
Electric charge	Charge per meter	$C(\tilde{x}_{ij}) = \frac{I(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})}{v_{ij}}$	$\text{A} \cdot \text{s/m}$	$\text{A} \cdot \text{s}$
Electric charge + Proximity to obstacles	Charge per meter and collision risk	$C(\tilde{x}_{ij}) = \frac{I(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})}{v_{ij}} + r_{ij}$	$\text{A} \cdot \text{s/m}$	$\text{A} \cdot \text{s}$

elapsed time that is estimated from the origin position or the estimated minimum time that remains to reach the goal position. The cost $C(\tilde{x}_{ij})$ is represented as the slowness of the robot, i.e. the inverse of its speed v_{ij} . The explanation here is simple: the faster the robot moves the sooner it will arrive at the goal location. Moreover, the second option includes a penalization r_{ij} . This penalization is also called the risk function. It serves to increase the cost based on the proximity of \tilde{x}_{ij} to the closest obstacle. A detailed explanation about how a risk function can be constructed is included in Chapter 4, so it is not addressed here. An example of its use can be seen in Figure 3.6b, where the areas surrounding the obstacles have higher cost. Figure 3.6c shows how the resulting path do not get too close to obstacles, but Figure 3.6d demonstrates how the path can still be close to them (see the portion close to \tilde{x}_g) if necessary. The use of the time minimization approach is commonly used in autonomous navigation applications and there may be situations in which it is prioritized to make the robot reach faster to any location (e.g. in the case of a disaster scenario in which a mobile robot has to reach the location of a possible victim). Another criterion shown in Table 3.1 is the minimization of energy to drive from the origin position to the target destination. Here, the total cost is represented by this amount of energy. The cost $C(\tilde{x}_{ij})$ is defined as the instantaneous power consumption $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ experienced by the robot at the location of each node, divided by the value of its speed v_{ij} . Similarly to the time minimization case, a risk function can be added to prevent the robot from getting too close to obstacles.

The functioning of the FMM to visit the grid nodes \tilde{x}_{ij} and calculate its values of total cost is explained next. Algorithm 1 presents a pseudo-code of how FMM works

for the case where the wave propagates from the goal, i.e. $T(\tilde{x}_{ij}, \tilde{x}_g)$ as defined in (3.25). For the case in which the wave propagates from the origin the process would be the same, but calculating $T(\tilde{x}_o, \tilde{x}_{ij})$, as defined in (3.24), instead of $T(\tilde{x}_{ij}, \tilde{x}_g)$. First of all, it is key to understand that each node has a state that indicates whether it has been visited by the FMM or not and if its value of total cost is definitive. The function S_{ij} , defined in (3.27), indicates this state for any node \tilde{x}_{ij} . In this way, the FMM modifies both the total cost and the state of each node when visiting it.

$$S_{ij} \in \{Far, Considered, Accepted\} \quad (3.27)$$

The explanation about each state S_{ij} is provided next:

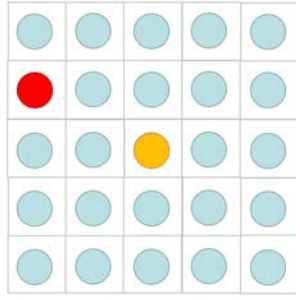
- **Far.** Nodes with this state have not been accessed yet by the calculation loop. This is the state in which all nodes, except the starting one (either the origin or the goal), are initialized. As the real value of total cost associated to each Far node is yet to be calculated, the initial value is irrelevant. Nevertheless, for this particular case, this value is set to ∞ symbolizing a very high value of total cost.
- **Considered.** A tentative value of total cost has been computed by the calculation loop at least once for nodes with this state.
- **Accepted.** During each iteration of the loop, the Considered node with the lowest value of total cost is retrieved. The value of this node is no longer tentative. Thereafter, the state of this node is updated to Accepted. The state and values of total cost of neighbouring Far and Considered nodes are updated. Furthermore, the state of the starting node (either S_o for the origin position or S_g for the goal) is initialized to Accepted, with a zero value of total cost.

Algorithm 1: Fast Marching Method

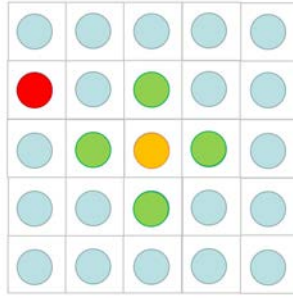
```

1  $T(\tilde{x}_{ij}, \tilde{x}_g), S_{ij} \leftarrow \infty, Far \quad \forall \tilde{x}_{ij} \in \tilde{\Omega}$ 
2  $T(\tilde{x}_g, \tilde{x}_g), S_g \leftarrow 0, Accepted$ 
3  $FWlist \leftarrow \{\}$ 
4  $FWlist \leftarrow FWlist + \tilde{x}_{ij} \quad \forall \tilde{x}_{ij} \in neigh(\tilde{x}_g) \mid C_{ij} \neq \infty$ 
5  $T(\tilde{x}_{ij}, \tilde{x}_g), S_{ij} \leftarrow \min \{T(\tilde{x}_{ij}, \tilde{x}_g), Eq.(3.30)\}, Considered \quad \forall \tilde{x}_{ij} \in neigh(\tilde{x}_g) \mid$ 
    $C_{ij} \neq \infty$ 
6 repeat
7    $\tilde{x}_{next} \leftarrow \tilde{x}_{ij} \mid \min_{\tilde{x}_{ij} \in FWlist} T(\tilde{x}_{ij}, \tilde{x}_g)$ 
8    $FWlist \leftarrow FWlist - \tilde{x}_{next}$ 
9    $S_{next} \leftarrow Accepted$ 
10   $FWlist \leftarrow FWlist + \tilde{x}_{ij} \quad \forall \tilde{x}_{ij} \in neigh(N_{next}) \mid S_{ij} = Far$ 
11   $S_{ij} \leftarrow Considered, \quad \forall N_{ij} \in neigh(\tilde{x}_{next}) \mid S_{ij} = Far$ 
12   $T(\tilde{x}_{ij}, \tilde{x}_g) \leftarrow \min \{T(\tilde{x}_{ij}, \tilde{x}_g), Eq.(3.30)\}, \quad \forall \tilde{x}_{ij} \in neigh(\tilde{x}_{next}) \mid S_{ij} =$ 
    $Considered$ 
13 until  $(S_o = Accepted) \vee (FWlist = \{\});$ 

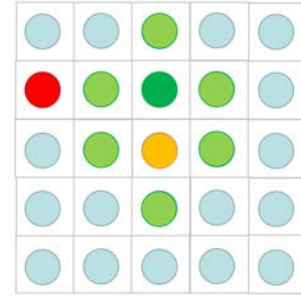
```



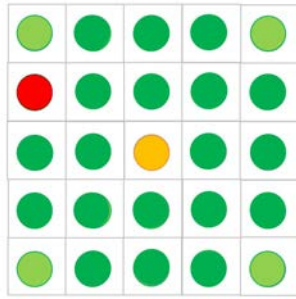
(a) The position of the goal node \tilde{x}_g (in orange) and the initial position \tilde{x}_o (in red) are set. The rest of nodes are labeled as Far (here colored in light blue).



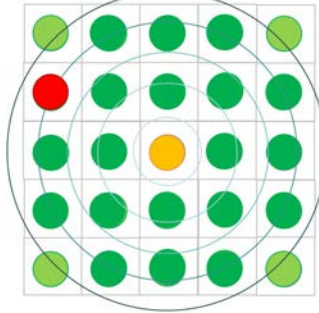
(b) The neighbours of the goal are visited. Their associated values of total cost are computed and they are labeled as Considered (here colored in light green).



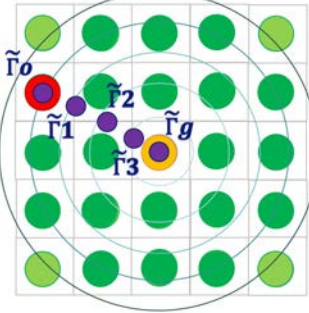
(c) The Considered node with the lowest value of total cost is selected and labeled as Accepted. Its Far neighbours are visited and labeled as Considered.



(d) The computation stops when the red node is labeled as Accepted or there are no more Far nodes.



(e) The calculated values of total cost make up a potential field that grows from the starting node.



(f) The path is retrieved by applying the gradient descent method on the potential field, starting from the red node.

FIGURE 3.8: Graphical step-by-step exposure of the functioning of FMM for path planning.

Figure 3.8 depicts a series of conceptual images that explain the policy the FMM follows to visit the nodes. This is the same policy the Dijkstra algorithm uses, but with the difference of how the value of total cost is updated. First of all, the FMM starts by labeling all nodes as Far and assigning to all of them a value of ∞ , as mentioned before. Thereafter, the FMM sets the initial condition, assigning a total cost of zero, to either the goal or the origin node. This election is up to the user as the resulting path will be the same as previously explained. The state of this starting node is set to Accepted. Its neighbours, as seen in Figure 3.8b, are labeled as Considered. A preliminary value of total cost is computed and assigned to them according to their values of cost $C(\tilde{x}_{ij})$. The *Front Wave* list, or *FWlist*, stores these Considered nodes. The node in *FWlist* with the lowest value of total cost, \tilde{x}_{next} , is extracted and labeled as Accepted ($S_{ij} = Accepted$). All the neighbours of \tilde{x}_{next} are labeled as Considered, as seen in Figure 3.8c. The values of total cost of these newly Considered nodes are calculated and they are later inserted into the *FWlist*. This process is repeated until either the origin node \tilde{x}_o (if the wave starts from the goal) or the goal node \tilde{x}_g (if the

wave starts from the origin) is Accepted or the $FWlist$ is empty. The latter can happen in case the wave does not reach the corresponding node. This could be caused by the existence of obstacles that isolate either the origin or the goal nodes and therefore prevent from finding any feasible path. Otherwise the wave is ensured to reach the node in question at some point, as this algorithm, as previously stated, is complete.

The calculation of the value of total cost to assign to each node is explained next. The eikonal equation presented in (3.22) is discretized with first order finite differences as in the original work of James A Sethian (1999). This results in the quadratic expression shown in (3.28) for the case of using a square grid, being Λ its resolution as indicated before in Figure 3.1a. The value of total cost of a node \tilde{x}_{ij} depends on the value of cost function assigned to it, $C(\tilde{x}_{ij})$, as well as the values of total cost of two of its neighbours. These two last values of total cost, $T(\tilde{x}_x, \tilde{x}_g)$ and $T(\tilde{x}_y, \tilde{x}_g)$ come from Equation (3.29), being the minimum total cost of the horizontal neighbours and the minimum total cost of the vertical neighbours respectively. The explicit form of Equation (3.28) is presented in (3.30). It considers the upwind condition from (3.31), whose compliance prevents the solution from presenting minimum local points. Besides, there are values of $T(\tilde{x}_x, \tilde{x}_g)$ and/or $T(\tilde{x}_y, \tilde{x}_g)$ that may lead to not having a solution complying with this condition. This is usually caused in situations where high differences in cost $C(\tilde{x}_{ij})$ exist in the grid. To overcome this, there is another more direct method to compute the value of $T(\tilde{x}_{ij}, \tilde{x}_g)$, in which it just considers the minimum value between $T(\tilde{x}_x, \tilde{x}_g)$ and $T(\tilde{x}_y, \tilde{x}_g)$ and adds $\Lambda C(\tilde{x}_{ij})$ to it.

$$(T(\tilde{x}_{ij}, \tilde{x}_g) - T(\tilde{x}_x, \tilde{x}_g))^2 + (T(\tilde{x}_{ij}, \tilde{x}_g) - T(\tilde{x}_y, \tilde{x}_g))^2 = \Lambda^2 C(\tilde{x}_{ij})^2 \quad (3.28)$$

$$\begin{bmatrix} T(\tilde{x}_x, \tilde{x}_g) \\ T(\tilde{x}_y, \tilde{x}_g) \end{bmatrix} = \begin{bmatrix} \min \{ T(\tilde{x}_{i-1j}, \tilde{x}_g), T(\tilde{x}_{i+1j}, \tilde{x}_g) \} \\ \min \{ T(\tilde{x}_{ij-1}, \tilde{x}_g), T(\tilde{x}_{ij+1}, \tilde{x}_g) \} \end{bmatrix} \quad (3.29)$$

$$T(\tilde{x}_{ij}, \tilde{x}_g) = \begin{cases} \frac{T(\tilde{x}_x, \tilde{x}_g) + T(\tilde{x}_y, \tilde{x}_g) + \sqrt{2(\Lambda C(\tilde{x}_{ij}))^2 - (T(\tilde{x}_x, \tilde{x}_g) - T(\tilde{x}_y, \tilde{x}_g))^2}}{2}, & |T(\tilde{x}_x, \tilde{x}_g) - T(\tilde{x}_y, \tilde{x}_g)| \leq \Lambda C(\tilde{x}_{ij}) \\ \min \{ T(\tilde{x}_x, \tilde{x}_g), T(\tilde{x}_y, \tilde{x}_g) \} + \Lambda C(\tilde{x}_{ij}), & \text{otherwise} \end{cases} \quad (3.30)$$

$$(T(\tilde{x}_{ij}, \tilde{x}_g) > T(\tilde{x}_x, \tilde{x}_g)) \vee (T(\tilde{x}_{ij}, \tilde{x}_g) > T(\tilde{x}_y, \tilde{x}_g)) \quad (3.31)$$

The explicit form of the eikonal equation in the case in which the grid $\tilde{\Omega}$ is hexagonal can be found in (3.32). Here, the total cost of the two neighbouring nodes are $T(\tilde{x}_{i'j'}, \tilde{x}_g)$ and $T(\tilde{x}_{i''j''}, \tilde{x}_g)$, where $\tilde{x}_{i'j'}$ and $\tilde{x}_{i''j''}$ are also hexagonal neighbours themselves.

$$T(\tilde{x}_{ij}, \tilde{x}_g) = \begin{cases} \min \{T(\tilde{x}_{i'j'}, \tilde{x}_g), T(\tilde{x}_{i''j''})\} + \frac{|T(\tilde{x}_{i'j'}, \tilde{x}_g) - T(\tilde{x}_{i''j''})|}{2} + \\ \quad + \frac{\sqrt{3(\Lambda(\tilde{x}_{ij}))^2 - 3(T(\tilde{x}_{i'j'}, \tilde{x}_g) - T(\tilde{x}_{i''j''}))^2}}{2}, & |T(\tilde{x}_{i'j'}, \tilde{x}_g) - T(\tilde{x}_{i''j''})| \leq \Lambda C(\tilde{x}_{ij}) \\ \min \{T(\tilde{x}_{i'j'}, \tilde{x}_g), T(\tilde{x}_{i''j''})\} + \Lambda C(\tilde{x}_{ij}), & \text{otherwise} \end{cases} \quad (3.32)$$

After the values of total cost have been computed, the next step is to extract the path $\tilde{\Gamma}$ out of them. As mentioned before, this path is a discrete set of waypoints: poses arranged one after another connecting the initial position $\tilde{x}_o = \tilde{\Gamma}_o$ and the goal position $\tilde{x}_g = \tilde{\Gamma}_g$. The waypoints between $\tilde{\Gamma}_o$ and $\tilde{\Gamma}_g$ can be calculated using a gradient descent method. This is possible because the characteristic direction (i.e. the direction of the optimal path passing at that location) in the eikonal equation is coincident with the negative gradient of the total cost. The number of waypoints depends on the value used for the step size d_{step} . For obtaining proper results, this value should be smaller than the length of a node side, i.e. $d_{step} < \Lambda$. Besides, the smaller this step parameter the smoother the path will result. However, a trade-off arises between the smoothness and the number of waypoints. Usually, a value of 0.5 m is reasonable and produces relatively good results. Starting from either the origin or the goal position (being the other position the one from which the wave started), the rest of waypoints are consecutively computed and added to the path. Equation (3.33) shows how new waypoints are created from previous ones according to the approach taken to calculate the total cost. Here, the gradient is normalized to keep the distance between waypoints equal to d_{step} . The path follows the direction indicated by this gradient vector, which in fact corresponds to the one where the value of total cost drops faster. Each waypoint is placed with the previous one as a reference, in the direction denoted by the gradient vector and with the step d_{step} . At a certain moment, one of the waypoints falls relatively close to the goal node, from which the FMM originally started and whose value of total cost is zero.

$$\tilde{\Gamma}_{k-1} = \tilde{\Gamma}_k - d_{step} \frac{\nabla T(\tilde{x}_o, \tilde{\Gamma}_k)}{\|\nabla T(\tilde{x}_o, \tilde{\Gamma}_k)\|}, \quad \tilde{\Gamma}_{k+1} = \tilde{\Gamma}_k - d_{step} \frac{\nabla T(\tilde{\Gamma}_k, \tilde{x}_g)}{\|\nabla T(\tilde{\Gamma}_k, \tilde{x}_g)\|} \quad (3.33)$$

Finally, the locomotion mode the robot should use when arriving at each waypoint depends on the locomotion that is chosen at the closest node \tilde{x}_{ij} . In other words, the locomotion l that is chosen to define $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ at such node according to Equation (3.5). For example, Figure 3.9 provides information about the best locomotion according to the example case presented in Figure 3.6, where two options for the locomotion set are used: $L = \{nd\}$ and $L = \{ww, nd\}$. The nd corresponds to the Normal-driving locomotion mode, while ww is the Wheel-walking. As a side note, the virtual model shown in Figure 3.9 is based on a real terrain:

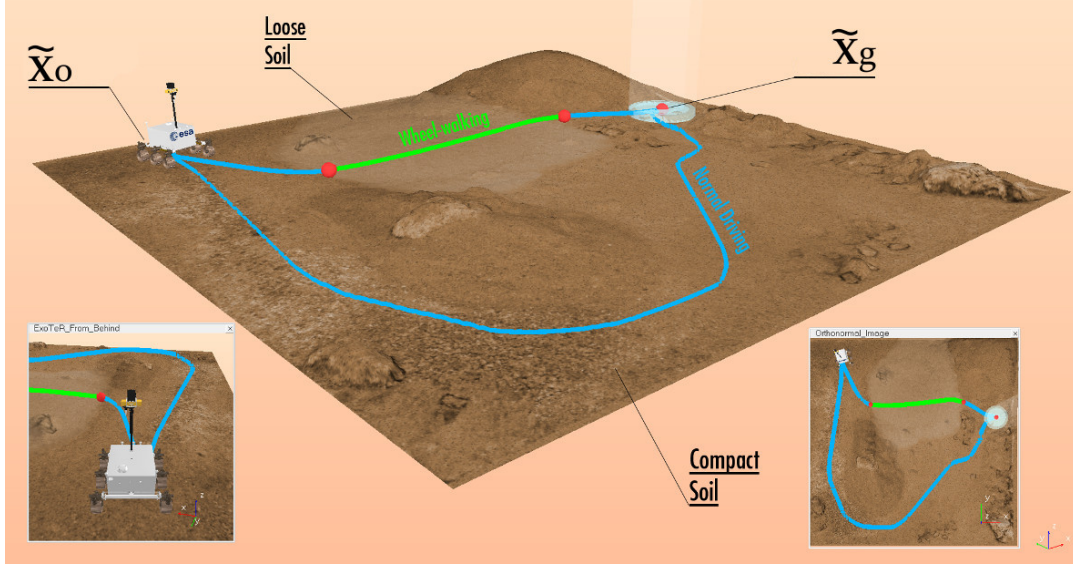


FIGURE 3.9: Virtual scenario with the terrain of the example case.

the Planetary Utilisation Testbed (PUTB), located at ESA-ESTEC¹. The cost map was created including obstacles from this environment. Figure 3.9 shows how the locomotion modes are planned to be used along each path according to L . As this is just a conceptual example, no further details into the numbers are provided here, but the details about a numerical experiment involving the use of the FMM along with the cost function of a reconfigurable rover can be found in Chapter 6.

3.4 Summary and Conclusions

This chapter exposes the ins and outs of the first contribution of this thesis. This contribution consists of the creation of a cost function for energy-minimizing path planning. In particular, this cost function acknowledges the use of multiple modes of locomotion by the same robot. Two locomotion modes are modelled to build up an example case of this cost function: Wheel-walking and Normal-driving. The first exploits the use of joints placed on top of wheeled legs. The second is purely based on the use of rolling and steerable wheels. Each of them compensates for the drawbacks of the other to improve the adaptability of the robot to multiple kinds of terrain. An Optimal Control PDE Solving path planner, FMM, is chosen to make use of the formulated cost function. This planner produces optimal, smooth and continuous paths on terrains with different compositions. Recalling the functioning of FMM comes in handy for explaining the path planning architecture making up the second thesis contribution, detailed in Chapter 4. It is important to mention one of the main omissions in the cost function provided here is the effect on turning manoeuvres for each of the presented locomotion modes. For instance, turning while doing Wheel-walking is still to be studied, and for the moment it is compensated

¹https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Planetary_Robotics_Laboratory

by the use of Point Turn manoeuvres to change the heading. Besides, the effect of switching between locomotion modes could also entail some energy costs. This cost function is more practical for planning long traverses, a scale in which these turning effects are not that relevant. Nevertheless, it is of interest to take them into account for planning at a smaller scale, like the next few meters from the rover position. However, this also entails substituting FMM by another PDE Solving path planning algorithm that is compatible with turning-dependant cost. FSM seems a suitable algorithm for this given the results provided by Takei et al. (2013). Moreover, it is assumed in this chapter scenarios without slopes. It would be interesting to extend this work by also considering the effect of slopes, tackled in Chapter 5, but with reconfigurable robots.

Chapter 4

Dynamic Multilayered Path Planning

"Intelligence is the ability of a system to adjust appropriately to a changing world, and the more capable of adjusting - the more versatile its adjusting power - the more intelligent it is."

Christopher Evans
The Mighty Micro, 1979

4.1 Introduction

The ability to drive long distances autonomously is advantageous for an explorer robot to speed up its mission. It can, thanks to the onboard guidance system, move without the intervention of human operators for a long time. For instance, in planetary exploration scenarios, communications with the ground stations have limitations. These limitations were already mentioned in Chapter 1. Yet, the plan followed by the robot may miss important information. This is because there is always an inherent level of uncertainty in a scenario that was not previously explored by a mobile robot in situ. External perception systems may provide a priori information describing this scenario. However, these systems have limitations to provide the preliminary map: the shape and the properties of the terrain can only be estimated up to a certain point. The limited resolution of the satellite images entails missing terrain accidents such as rocks, grooves or rifts. As a result, the long-traverse path the robot is following does not tackle these elements and may even pass through them. This situation jeopardizes the integrity of the robot and forces it to stop. Thereafter, two options arise: either the robot waits for new commands/plans from the ground station, which entails wasting valuable mission time, or it autonomously decides the next course of action to overcome the newfound terrain elements. The latter is desirable towards keeping the objective of driving long distances without human

intervention. This entails the need of updating the path according to the newly acquired information. In other words, a solution to come out of this situation is the use of a replanning process that dynamically updates the path. This replanning is made according to what the rover perceives, i.e. according to the new information coming from the rover onboard sensors.

The data obtained from the onboard sensors is generally more accurate than that coming from external sources, such as satellites. In this way, the shape of the terrain accidents can be better perceived and the system can consider them dangerous or not according to the robot mechanical structure. For example, most Mars rovers are equipped with stereo cameras that allow them to detect terrain features with a resolution of centimetres, while a rocker-bogie system allows them to negotiate rocks that fall under a certain height threshold. However, some caution is still desirable as these rocks may still damage the wheels (e.g. because of being pointy or having sharp edges). Besides, passing on top of them with the wheels may entail more energy to spend. The main drawback of using onboard sensors for mapping in contrast with external sources is the fact that their range is limited. Stereo cameras in the front can reach only from one to just a few meters ahead of the vehicle. This means the update of the original path is limited by this distance. A common strategy shared by most applications in autonomous navigation is to categorize the maps obtained by external sources as global, while those obtained from the robot itself are local. For instance, this global-local approach has been used in past applications such as the exploration of the Martian surface (Carsten et al., 2007). After checking how one of the rovers, only equipped with a local planner, got stuck in a situation where it was partially surrounded by rocks, NASA engineers decided to remotely install a global planner (Maimone et al., 2007). In this way, the rover acquired the capability to autonomously deliberate longer paths using global information and the Field-D* algorithm. Field-D* is an Any-angle algorithm mentioned in the state of the art analysis of this thesis (see Chapter 2). It performed successfully on the rovers and demonstrated the effectiveness of the global-local approach. As a Graph Search planner, it constraints the shape of the path to the nodes of a grid. Besides, it was demonstrated how other Any-angle methods performed better in obstacle avoidance cases (Nash and Koenig, 2013), as these methods do not ensure the resulting path is optimal by using heuristic searches and functions. As a local planner the Mars rovers use GESTALT. This method evaluates a discrete set of immediate short trajectories, as depicted in Figure 4.1, and selects the most suitable one. This evaluation is based on the terrain condition, determining as a result the safest path to navigate.

The main motivation behind the thesis contribution presented in this chapter is the use of Optimal Control PDE solving path planners, whose advantages were mentioned in previous chapters, to exploit this global-local strategy. In other words, the contribution in question provides this kind of algorithms with the capability to dynamically update the resulting paths while the rover is driving. This feature

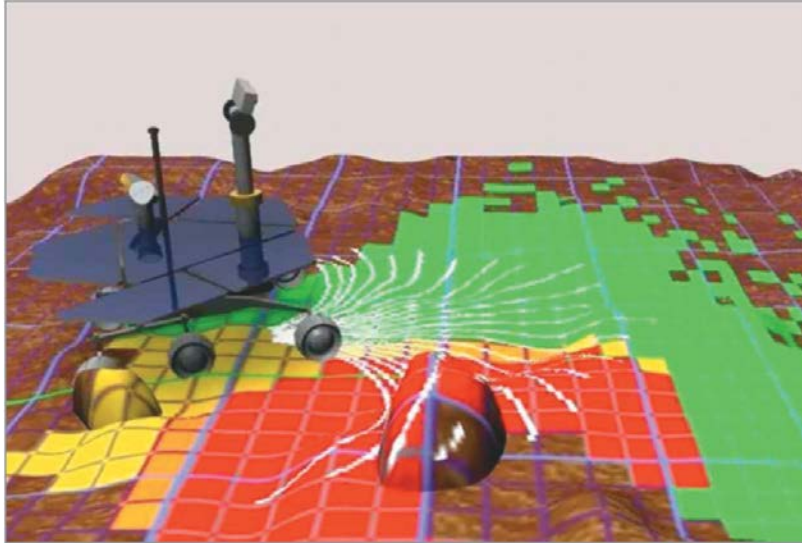


FIGURE 4.1: Representation of the functioning of GESTALT: a set of predefined paths is evaluated and thereafter one of them is selected according to safety conditions. Credits: Bajracharya et al. (2008).

was missing in PDE solving path planners, making it one of their main drawbacks. The main reason for choosing this kind of algorithm is that, as seen in Chapter 2, they stand out mainly due to their capability to obtain optimal paths that are not restricted to the grid structure. They invest similar or equal computational complexity as Graph Search algorithms, which are not capable to avoid the grid restrictions when returning a path. Hence, PDE solving path planners are suitable to be used in combination with cost functions aimed at minimizing the energy spent in any traverse, as explained in Chapter 3. In this way, saving up energy may result in achieving longer and more numerous traverses, which in turn increases the number of places to explore. Moreover, the combination of algorithms of this kind working at two scales, global and local, is a novelty introduced here. This is possible thanks to the use of a multi-resolution grid that combines information coming from both external sources and the robot perception systems, integrating path planning at different scales in a novel way. With regards to the planning algorithms, it is assumed the global plan is produced by an isotropic planner such as the Fast Marching Method (FMM). At a local scale, the chosen algorithm to execute the replanning operation is the heuristic version of this method, called FM*.

The next sections are arranged as follows. First, this chapter introduces the architecture of the aforementioned path planning solution in Section 4.2. Second, this chapter details the multi-resolution grid used to combine the data from initial global maps and the maps provided by the rover onboard sensors in Section 4.3. Section 4.4 introduces the novel PDE solving based approach that serves to update the path locally: the Local Path Repairing (LPR). Finally, Section 4.5 provides a summary of the chapter and some of the conclusions extracted from the presented content.

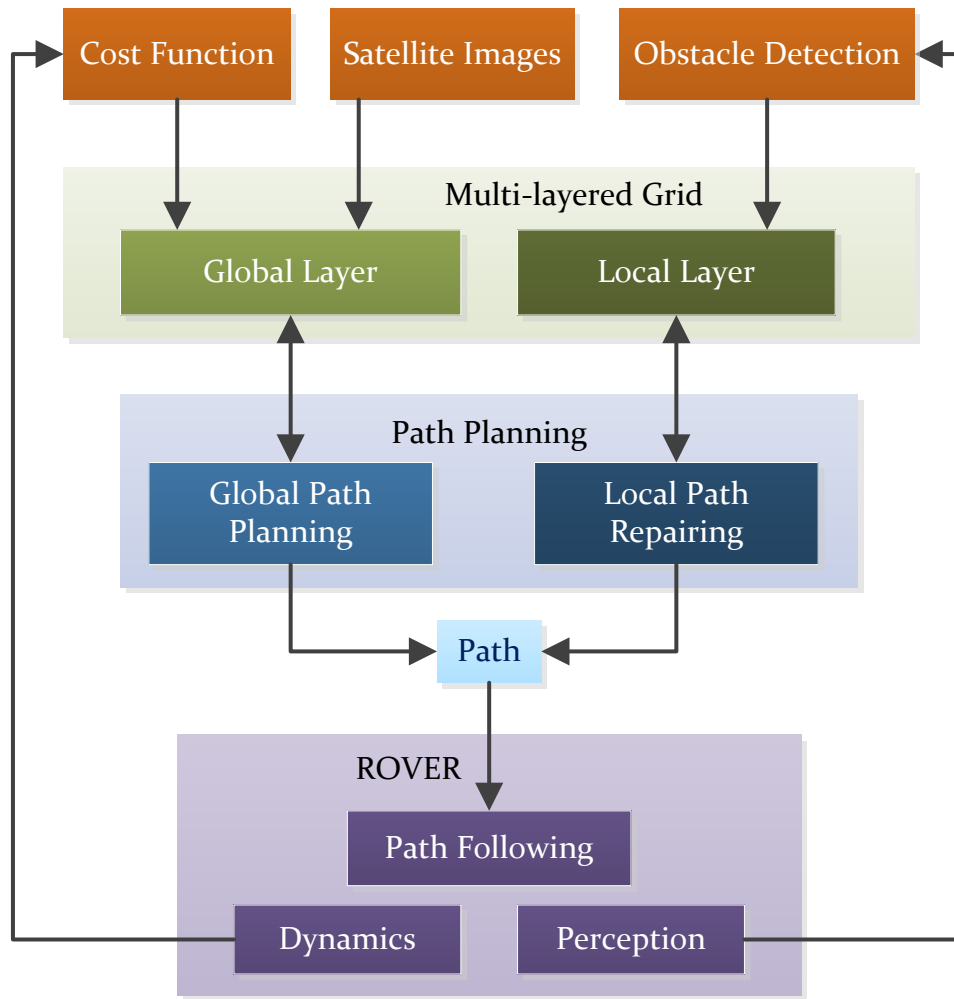


FIGURE 4.2: Schematic showing the flow of information in DyMu, the proposed dynamic path planning solution. This architecture rests on the multi-layered grid to represent information at different scales. The path planner makes use of this grid as well as a cost function to generate and update the optimal path. This path serves to guide the rover in its autonomous drive.

4.2 Multi-scale Planning Architecture

The global-local path planning solution presented in this chapter is named Dynamic Multilayered path planner (DyMu). This solution can plan and dynamically update a smooth and optimal path using information about the environment coming from different sources. Figure 4.2 depicts the architecture behind it, with the information that flows between the different parts. There are three main conceptual blocks in this schematic going from top to bottom: the multi-layered grid, the path planning and the one representing the rover itself. This architecture addresses information formatted in two different ways, one coming from out of the robot (satellite imagery in this case) and another from the robot perception system. These two ways are in different sizes of covered areas and with different values of resolution. They correspond to the global and local scales of planning and are explained next.

First of all, global maps must cover large outdoors and off-road areas from which the planner produces an initial long path. For example, these maps can be built from the imagery provided by satellites. For the particular case of Mars exploration, these images can cover huge areas in terms of square kilometres, while using a resolution that can achieve one meter per pixel or even 20 centimetres in some cases, as mentioned in Chapter 3. The satellite imagery allows modelling the terrain with a certain degree of detail before sending the rover to navigate. Besides, the type and characteristics of the soil may be estimated by using terrain classifiers (Brooks et al., 2012; Rothrock et al., 2016). This information can give an idea of how the locomotion system of the robot will perform on it, in combination of a cost model that serves as a path planner cost function. For instance, Chapter 3 introduced a model in (3.5) to estimate the energy consumption according to the terrain characteristics. In this way, the overall course of action for a long time can be planned on Earth without needing the rover to be physically there.

The second block shown in Figure 4.2 represents the multi-layered grid, which is explained in detail in the next section. It is a regular multi-resolution grid that tackles the global information and later interconnects it with information coming from the internal means, i.e. the local maps provided by the robot itself. This special grid is capable of storing and combining the input map information with different sizes and values of resolution using two overlapping square grid layers. The first of the two layers forming the multi-layered grid is called the global layer, while the second is referred to as the local layer. On the one hand, the global layer handles information coming from external sources, such as the mentioned satellite imagery or even imagery from drones in the case of autonomous navigation applications on Earth. As mentioned, this information comes in the form of large but not very detailed maps. On the other hand, the local layer addresses information that comes from the robot perception system. Usually, this information, obtained through the robot sensors, has limited reach and covers just a few meters around the robotic system or even just ahead of it. In other words, the map that feeds the local layer only covers very small areas, in comparison with the whole area considered for the whole mission. This can be compensated by having high resolution, which helps to capture with detail the shape of the terrain located nearby and look for any possible obstacle, i.e. any terrain shape that could harm the robot in case they contact. To summarize, it is assumed the robot takes as input a large map with low resolution for planning the initial path, and it provides information to the global layer. This information is processed using a cost function that represents the robot dynamic model, in order to be later used by the global path planner. Smaller maps with high resolution, that serve to update the local layer, can be used to represent those elements found that are placed in the way of the rover, triggering the update of the path. The global layer and the local layer, as will be seen later, are connected, so the different information stored in the multi-layered grid is integrated.

The path planning component generates and updates the path that guides the

robot to the desired destination. The performance of this component rests on the information saved in the multi-layered grid. In an analogous way to this special grid, the path planning component comprises two different processes: the Global Path Planning (GPP) and the Local Path Repairing (LPR). The GPP is in charge of generating an initial path at the beginning of the mission. It relies on the information provided by the global layer to create a rough preliminary long-traverse path. This path connects an origin position \tilde{x}_o , which may be coincident with the robot position, with the desired position to reach, i.e. the goal position \tilde{x}_g . Both the origin and the goal positions, as well as the resulting path, have to be located within the global layer, in a similar fashion as how the region of interest Ω and its discretized version $\tilde{\Omega}$ are defined in Chapter 3. In this way, all the nodes making up the global layer have a value of cost $C(\tilde{x}_{ij})$ that is assigned to them, modelled after a path planning optimization criterion (see Table 3.1 from Chapter 3 as well). Moreover, an isotropic PDE solving algorithm, the FMM, is chosen to act as the global path planner. Its functioning was introduced back in Chapter 3.

The LPR compensates for the lack of detail in the initial planning by repairing the path based on the information coming from the local layer. Its main advantage is that it not only repairs the path based on the information describing nearby obstacles but it also takes into consideration the computation done during the GPP. The algorithm used here is the heuristic version of the FMM, the FM*. This method propagates a wave that is guided by a custom heuristic function. This function can be based on the values of total cost computed on the global layer or on reaching any of the waypoints from the original path. The first case is called the Sweeping approach, as the LPR searches a certain local node from the local layer that complies with certain requirements. The second case is called the Conservative approach, as it focuses on repairing only certain sections of the original path, trying to preserve existing waypoints as much as possible. More details into the functioning of the LPR and these two approaches are provided later.

The last part of the schematic shown in Figure 4.2 is the robot itself. It must have the necessary software to follow the provided path. To do this, algorithms like the Conservative Pursuit, or C-Pursuit (Gerdes et al., 2020), can be used. As also depicted in the schematic of Figure 4.2, there is a cost function based on the dynamics of the robot locomotion. It serves as input to the GPP through the global layer, representing the power consumption of the robot according to terrain parameters (and hence setting the planning criterion to energy minimization). This function shall correspond to the one presented in the previous chapter, aimed at both reconfigurable and non-reconfigurable rovers. Moreover, it is assumed the rover has installed the required hardware and software to detect and map the obstacles that may appear on its way. The resulting information from them serves as input to the local layer and, if necessary, triggers the LPR.

4.3 Multi-layered Grid

An important trade-off between the extension of the area Ω that is covered and the value of the resolution Λ arises when creating a grid that fully or partially models an environment. The result of this trade-off determines the number of nodes contained in such a grid. This number will influence the computational effort taken by the path planner, as it will need to visit more or fewer nodes to generate a path. As a reminder, for the case of the FMM, as well as many Graph-search algorithms based on the Dijkstra policy to visit nodes, the computational complexity is $O(n_{nodes} \log(n_{nodes}))$, being n_{nodes} the total number of nodes in the grid $\tilde{\Omega}$.

On the one hand, the euclidean distance between the destination and the robot depends on the area that the map (i.e. the region Ω) covers. Pushing the map boundaries far away means that the destination can be placed further from the robot. This is translated into increasing the distance the robot can traverse with a single plan. Besides, covering more areas can result in finding a better path to reach this destination. However, bigger areas demand in turn more nodes to process, i.e. the computational load increases. On the other hand, the resolution influences the maximum level of detail with which the map elements will be represented. In a similar way to making the covered area bigger, the resolution affects the computational load of the planner. The increase in the resolution, i.e. using smaller spacing between nodes, entails a higher number of nodes to process. Nevertheless, reducing the area covered results in losing possible destinations. Lowering the resolution has a loss of definition as consequence and, inherently, a loss in path quality. The latter can be clearly understood as the loss of level of detail entails missing elements such as rocks of a certain size. Having all this into account, a problem arises in the form of making a trade-off between the number and arrangement of the nodes (spacing) and the quality of the path. In other words, how to make longer paths while taking into account the presence of relevant small elements and reducing the computational load of the planner.

Chapter 2 already introduced different approaches to map the environment using a grid (see Figure 2.1 located in such chapter). Regular grids include hexagonal and square (see Figure 3.1 in Chapter 3). The regular tessellation using square cells goes in consonance with the pixels of those images taken from orbital satellites. The centre of each cell is a node and each node is connected to others using the von Neumann neighbourhood, as already presented in Equation (3.1) from Chapter 3. In other words, each node is connected to two vertical and two horizontal neighbours using the constant spacing Λ , for the case of the global layer, or λ , for the case of the local layer. This is more clear in Figure 4.3, where the two overlapping layers are represented. The local layer is constructed upon the subdivision of nodes from the global layer. A number of $(\Lambda/\lambda)^2$ local nodes results from each subdivision, given that Λ/λ returns an integer number. For the case presented in Figure 4.3, this number is sixteen. As mentioned, the neighbourhood for the global layer is already

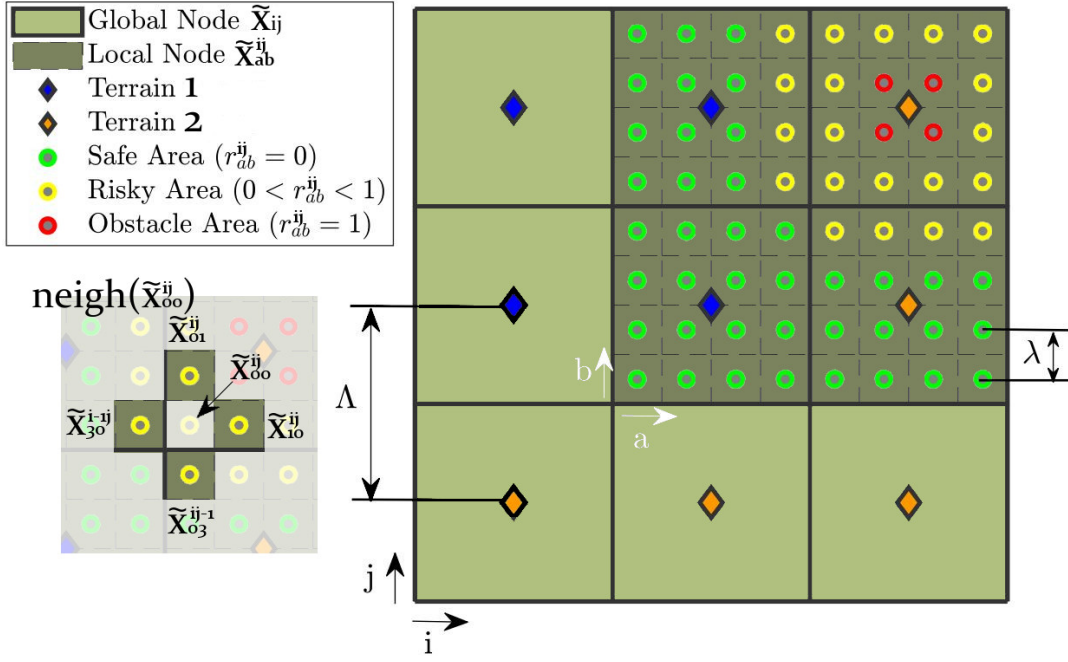


FIGURE 4.3: Illustrative image of the multi-layered grid. Each global node indicates the type of terrain that is contained in its area and occupies the same area as a finite number of local nodes, each of them providing an indication of its state relative to nearby obstacles.

expressed in (3.1). The expression for the neighbourhood in the local layer is presented in Equation (4.2). As can be denoted, the latter neighbourhood is different and more complex than that of the global layer. This is because the neighbours of a local node \tilde{x}_{ab}^{ij} that is located at the edge or the corner of a global node \tilde{x}_{ij} are the local nodes resulting from another global node. Indexes a and b indicate the exact position of the local node inside the parent global node. They are bounded by the values shown in (4.1). Figure 4.3 portrays the neighbourhood of a local node \tilde{x}_{00}^{ij} . The indexes $a = 0$ and $b = 0$ mean that this local node is placed at the bottom left corner of a global node \tilde{x}_{ij} . Two of its neighbours, \tilde{x}_{30}^{ij} and \tilde{x}_{03}^{ij-1} , are originated from different global nodes, \tilde{x}_{i-1j} and \tilde{x}_{ij-1} respectively. As in this particular case each global node can comprise sixteen nodes, then $\frac{\Lambda}{\lambda} - 1 = 3$. Having this in mind, the nodes \tilde{x}_{30}^{ij} and \tilde{x}_{03}^{ij-1} are located at the top left corner and the bottom right corner of their respective parent global nodes.

$$a \in [0, \frac{\Lambda}{\lambda} - 1], b \in [0, \frac{\Lambda}{\lambda} - 1] \quad (4.1)$$

$$\text{neigh}(\tilde{x}_{ab}^{ij}) = \begin{cases} \left\{ \tilde{x}_{\Lambda/\lambda-1,b}^{i-1,j}, \tilde{x}_{1,b}^{i,j}, \tilde{x}_{ab-1}^{i,j}, \tilde{x}_{ab+1}^{i,j} \right\}, & a = 0 \wedge b \in (1, \frac{\Lambda}{\lambda} - 2) \\ \left\{ \tilde{x}_{\Lambda/\lambda-2,b}^{i,j}, \tilde{x}_{0,b}^{i+1,j}, \tilde{x}_{ab-1}^{i,j}, \tilde{x}_{ab+1}^{i,j} \right\}, & a = \frac{\Lambda}{\lambda} - 1 \wedge b \in (1, \frac{\Lambda}{\lambda} - 2) \\ \left\{ \tilde{x}_{a-1,b}^{i,j}, \tilde{x}_{a+1,b}^{i,j}, \tilde{x}_{a,\Lambda/\lambda-1}^{i,j-1}, \tilde{x}_{a,1}^{i,j} \right\}, & a \in (1, \frac{\Lambda}{\lambda} - 2) \wedge b = 0 \\ \left\{ \tilde{x}_{a-1,b}^{i,j}, \tilde{x}_{a+1,b}^{i,j}, \tilde{x}_{a,\Lambda/\lambda-2}^{i,j}, \tilde{x}_{a,0}^{i,j+1} \right\}, & a \in (1, \frac{\Lambda}{\lambda} - 2) \wedge b = \frac{\Lambda}{\lambda} - 1 \\ \left\{ \tilde{x}_{\Lambda/\lambda-1,b}^{i-1,j}, \tilde{x}_{1,b}^{i,j}, \tilde{x}_{a,\Lambda/\lambda-1}^{i,j-1}, \tilde{x}_{a,1}^{i,j} \right\}, & a = 0 \wedge b = 0 \\ \left\{ \tilde{x}_{\Lambda/\lambda-1,b}^{i-1,j}, \tilde{x}_{1,b}^{i,j}, \tilde{x}_{a,\Lambda/\lambda-2}^{i,j}, \tilde{x}_{a,0}^{i,j+1} \right\}, & a = 0 \wedge b = \frac{\Lambda}{\lambda} - 1 \\ \left\{ \tilde{x}_{\Lambda/\lambda-2,b}^{i,j}, \tilde{x}_{0,b}^{i+1,j}, \tilde{x}_{a,\Lambda/\lambda-1}^{i,j-1}, \tilde{x}_{a,1}^{i,j} \right\}, & a = \frac{\Lambda}{\lambda} - 1 \wedge b = 0 \\ \left\{ \tilde{x}_{\Lambda/\lambda-2,b}^{i,j}, \tilde{x}_{0,b}^{i+1,j}, \tilde{x}_{a,\Lambda/\lambda-2}^{i,j}, \tilde{x}_{a,0}^{i,j+1} \right\}, & a = \frac{\Lambda}{\lambda} - 1 \wedge b = \frac{\Lambda}{\lambda} - 1 \\ \left\{ \tilde{x}_{a-1,b}^{i,j}, \tilde{x}_{a+1,b}^{i,j}, \tilde{x}_{a,b-1}^{i,j}, \tilde{x}_{a,b+1}^{i,j} \right\}, & \text{otherwise} \end{cases} \quad (4.2)$$

As stated before, the global nodes \tilde{x}_{ij} that make up the global layer store information about the cost function $C(\tilde{x}_{ij})$. This cost function is used by the GPP to produce the initial optimal path at a global scale. On the other hand, the local nodes that form the local layer store information about their proximity to the nearest obstacles. This information comes in the form of a risk function r_{ab}^{ij} that normalizes this distance and returns values between 0 and 1, i.e. $r : \tilde{x}_{ab}^{ij} \in \text{local layer} \rightarrow \mathbb{R} \in [0, 1]$. In this way, this function gives a hint about the proximity to the closest obstacle to prevent the robot from colliding with it. If it takes a value of zero the local node in question is placed within the safe area. This means the distance from this local node to the nearest obstacle is higher than a certain distance threshold. On the contrary, a value of one means that the local node is placed at the border of an obstacle. Therefore, the robot should not be located at such a local node to avoid any harm to its integrity. Values of risk between both limits point out that the local node is located within the so-called risky area. This area surrounds the obstacles and is calculated using a process named Risk Expansion that will be explained later.

To summarize, the multi-layered grid comprises two layers: the global layer and the local layer. On the one hand, the global layer is defined in (4.3) as the set of all global nodes that form the grid $\tilde{\Omega}$. A value of cost $C(\tilde{x}_{ij})$ is associated to each of them. On the other hand, the local layer encompasses all local nodes that result from subdividing global nodes, as it is defined in (4.4). A value of risk function r_{ab}^{ij} is assigned to each local node according to its proximity to obstacles.

$$\text{Global Layer} = \{ \tilde{x}_{ij} \mid \forall (i, j) | \tilde{x}_{ij} \in \tilde{\Omega} \} \quad (4.3)$$

$$\text{Local Layer} = \{ \tilde{x}_{ab}^{ij} \mid \forall (a, b, i, j) | \tilde{x}_{ij} \text{ is subdivided} \} \quad (4.4)$$

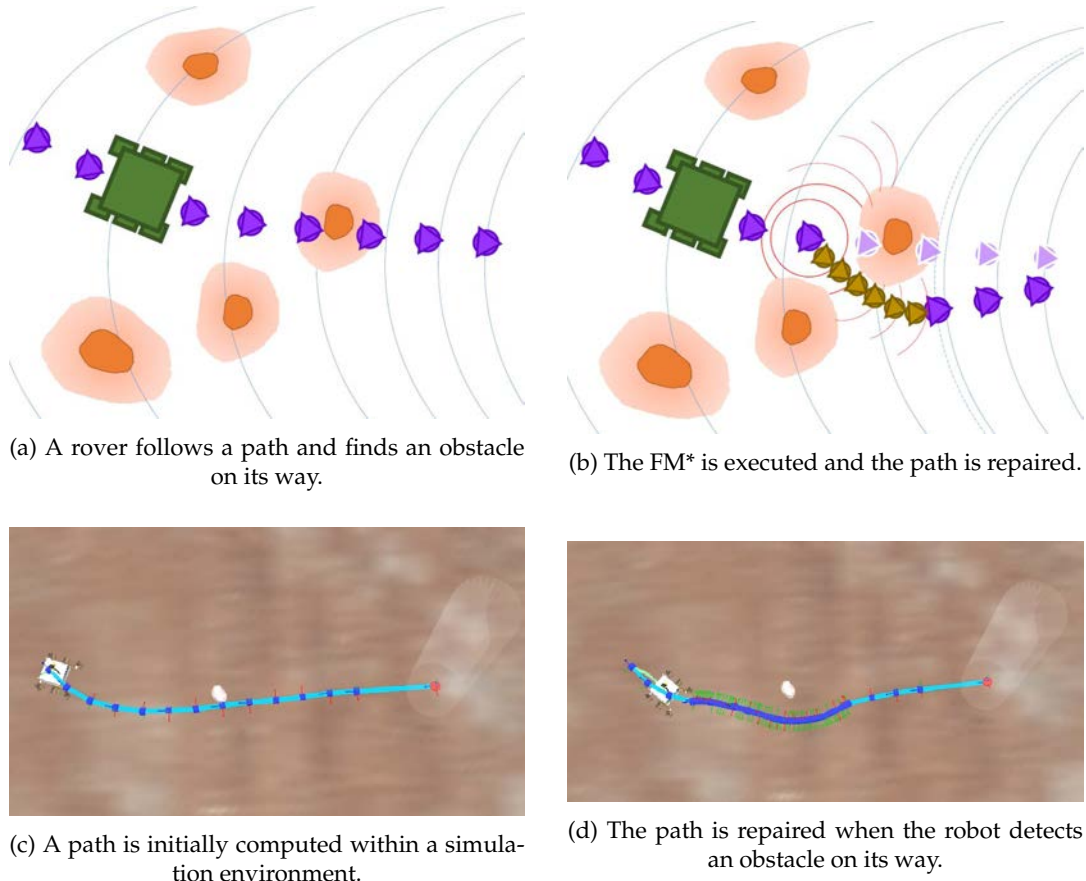


FIGURE 4.4: The GPP generates a global path (a)(c). Then, whenever an obstacle is detected in the way of the rover, it computes an alternate path, either completely new (b) or just partially, reconnecting with the original one at some point (d).

4.4 Local Path Repairing

The main premise of the Local Path Repairing (LPR) process is to repair the path generated by the Global Path Planning (GPP) process using the local layer. An example of this functioning is shown in Figure 4.4. Here, a robot finds an obstacle on its way while following a path. It is worth mentioning that the robot may still detect obstacles that are placed far from the path. Although these obstacles do not trigger the path update, they must still be considered in the local layer. After detecting the obstacle that blocks the path, the LPR is triggered and the update process searches for a feasible solution. This process relies on using the FM*. This is a heuristic version of FMM, which means the wave that is propagated expands in a direction determined by a heuristic function. This is done to penalize the wave when propagating backwards for two reasons. First, to save computation time by reducing the number of visited nodes. Second, to prioritize solutions that prevent the robot from returning to already visited positions. Two approaches are further discussed in detail later. One of them is the Sweeping approach (see Figure 4.4b), in which the local waypoints reach a location from which new global waypoints are created using the

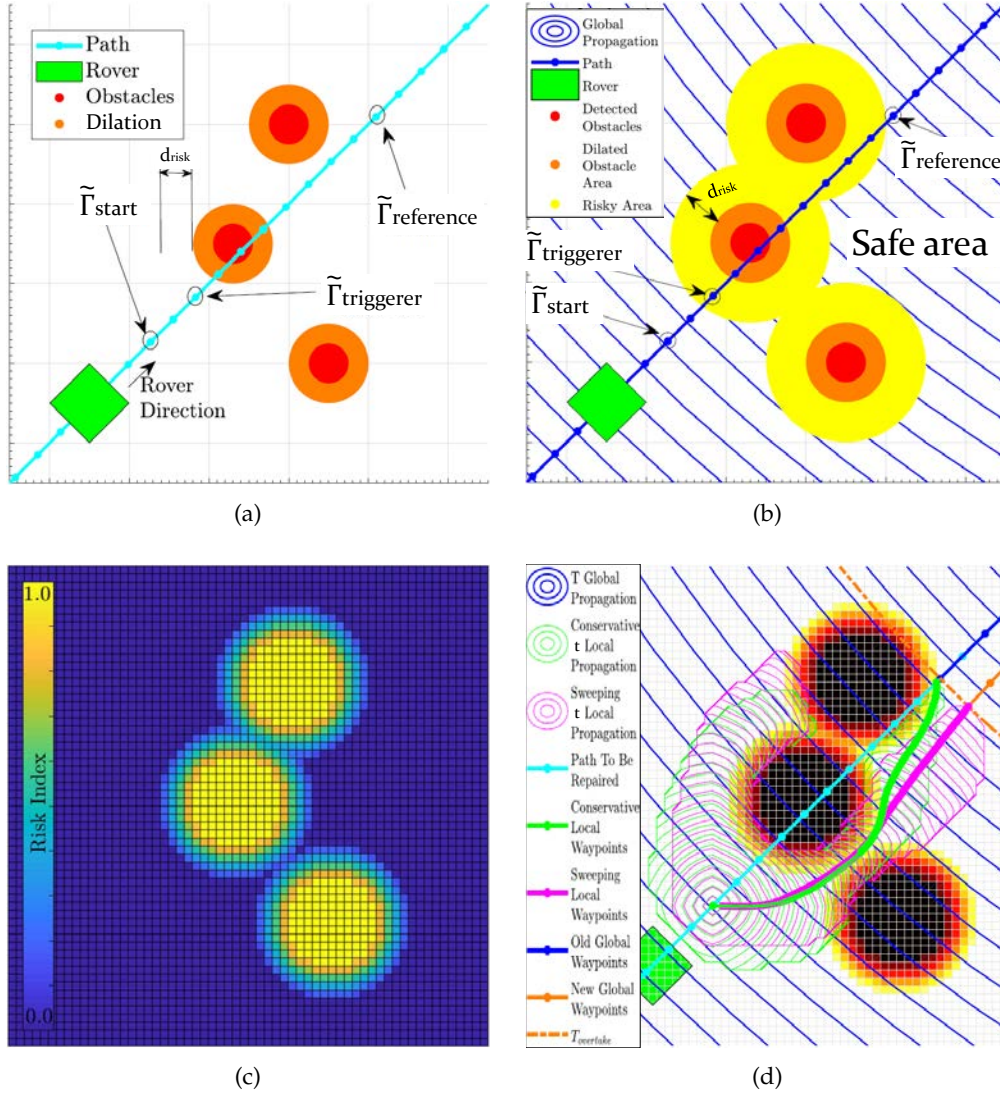


FIGURE 4.5: The LPR process is composed by several operations. First, when obstacles are in the way of the rover the process is triggered (a). Local nodes are created (b) and the values of r_{ab}^{ij} are calculated (c). Finally, the FM* uses these values to get a new path (d).

global layer. The other approach is the Conservative (see Figure 4.4d). In this case, the local waypoints connects the robot position with one of the global waypoints from the previous path.

Figure 4.5 depicts the steps followed by the LPR to perform the path update in the two approaches. Here is presented another example case, this time in more detail, where the robot encounters some obstacles on its way (Figure 4.5a). As the path goes through one of these obstacles, the repairing is triggered. First of all, the LPR not only maps the obstacles onto the local layer but also dilates them according to the shape of the robot. Usually, this dilation is equivalent to the radius of the circumference that circumscribes the robot 2d projection. Besides, it can even be more dilated considering errors in the tracking of the path. Figure 4.5b shows how one of the global waypoints from the original path, $\tilde{\Gamma}_{triggerer}$, is located in the risky area, under

a distance d_{risk} from the closest obstacle. This is the condition that triggers the LPR. Moreover, another global waypoint, $\tilde{\Gamma}_{reference}$, will be taken into account in a later step. This global waypoint can be the first one that is located in the safe area after $\tilde{\Gamma}_{triggerer}$. $\tilde{\Gamma}_{reference}$ can be also another global waypoint placed even further, but still in the safe area. Thereafter, the LPR creates in the local layer a potential field of cost around each of the obstacles. The key behind this is to make the local cost increase with the proximity to obstacles, representing the risk of colliding with them. For this reason, it is defined a function called the risk function r_{ab}^{ij} , which return values that are assigned to each local node \tilde{x}_{ab}^{ij} . The risk can take a value from zero to one, where zero is completely safe and one is next to the obstacle. The values ranging from zero to one are set to those nodes under a certain distance d_{risk} to the closest obstacle, as can be checked in the example shown in Figure 4.5c. Finally, the LPR employs the FM* to compute local waypoints as part of the final path. This means the new local waypoints are placed between global waypoints. The previous global waypoints are not accounted for since they are assumed to be already tracked by the robot. The global waypoints placed after the local section will depend on the chosen approach as mentioned before. In the Sweeping approach these global waypoints will be all-new, while in the Conservative approach they will be part of the previous path (in particular, those global waypoints from $\tilde{\Gamma}_{reference}$ until the last one). Figure 4.5d showcases both repairing approaches. Next, it is explained in detail how the distance d_{risk} , the values of the risk function and the local cost values are defined, together with the heuristic function used in both approaches.

The value of d_{risk} must be chosen so it complies with the condition set in (4.5). The reason behind this condition is to avoid that any point in the segment that connects consecutive global waypoints falls in the obstacle area. As a consequence, there exists an admissible error in which the segment connecting two consecutive global waypoints can fall in the risky area (see Figure 4.6). This area of committed error is reduced by increasing d_{risk} , but increasing this value will trigger a higher number of calls to the LPR process and increase the size of the risky area as a consequence.

$$d_{risk} \geq d_{step}\Lambda \quad (4.5)$$

The values given by the risk function are assigned to the local nodes following an operation named Risk Expansion. This way of creating repulsive potential fields of cost resembles the approaches that are introduced in previous works (Petres et al., 2005; Valero-Gomez et al., 2013). The main difference in the approach taken here with respect to those works is its use in the local layer to later repair a path. The Risk Expansion is based on the FMM, but it has some differences with respect to its application on path planning. Instead of calculating the value of total cost, the function to calculate here is the risk r_{ab}^{ij} for all nodes \tilde{x}_{ab}^{ij} in the local layer. To do this, the eikonal equation shown in (4.6) is used by the Risk Expansion. The cost comes in the form of $-1/d_{risk}$ because the risk decreases as the wave gets further from the obstacles, until reaching the distance of d_{risk} . In other words, the risk decreases as the

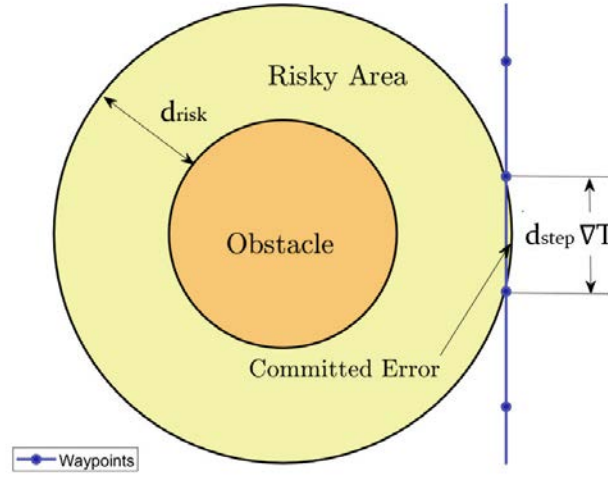


FIGURE 4.6: Admissible error due to path discretization after the GPP. In this extreme case the path does not trigger the LPR, although the segment connecting two waypoints traverses the risky area.

wave propagates. The discretized explicit version of this equation is shown in (4.7), which complies with the *Upwind condition* (4.8). Here, $r_{ab}^{ij,x}$ and $r_{ab}^{ij,y}$ are the horizontal and vertical neighbours with higher value of risk. Values between zero and one are assigned to each of the nodes, decreasing the value as the wave advances. The wave stops at those Accepted nodes that already have a value of risk higher than that of the wave. As a result, the risk function is calculated as in the example that is depicted in Figure 4.5c. All those nodes at a distance d_{risk} to the closest obstacle are assigned a value of risk higher than zero.

$$\|\nabla r_{ab}^{ij}\| = -\frac{1}{d_{risk}} \quad (4.6)$$

$$r_{ab}^{ij} = \begin{cases} \frac{1}{2} \left(r_{ab}^{ij,x} + r_{ab}^{ij,y} - \sqrt{2(\lambda/d_{risk})^2 - (r_{ab}^{ij,x} - r_{ab}^{ij,y})^2} \right) & |r_{ab}^{ij,x} - r_{ab}^{ij,y}| \leq \lambda/d_{risk} \\ \max \{ r_{ab}^{ij,x}, r_{ab}^{ij,y} \} - \lambda/d_{risk}, & otherwise \end{cases} \quad (4.7)$$

$$(r_{ab}^{ij} < r_{ab}^{ij,x}) \vee (r_{ab}^{ij} < r_{ab}^{ij,y}) \quad (4.8)$$

The Algorithm 2 contains the pseudo-code of the Risk Expansion. As a first step, a value of risk $r_{ab}^{ij} = 1$ is set on those local nodes that are not an obstacle but are placed next to obstacle local nodes. In other words, these nodes have at least one neighbour that is located in the obstacle area. Their respective state S_{ab}^{ij} is set to Considered and they are stored in the Front Wave list *FWlist*. This list is similar to the one introduced in Chapter 3, storing all Considered nodes while the FMM is functioning. With respect to the rest of nodes, those located in obstacle area have a

Algorithm 2: Risk Expansion using the Fast Marching Method

```

1  $r_{ab}^{ij}, S_{ab}^{ij} \leftarrow 1, Accepted \quad \forall \tilde{x}_{ab}^{ij} \in \text{Obstacle Area}$ 
2  $r_{ab}^{ij}, S_{ab}^{ij} \leftarrow 0, Far \quad \forall \tilde{x}_{ab}^{ij} \notin \text{Obstacle Area}$ 
3  $FWlist \leftarrow \{\}$ 
4  $FWlist \leftarrow FWlist + \tilde{x}_{ab}^{ij}, \quad \forall \tilde{x}_{ab}^{ij} \in \text{Local Layer}$ 
    $\quad | (\exists \tilde{x} \in \text{neigh}(\tilde{x}_{ab}^{ij}) \mid r(\tilde{x}) = 0) \wedge (r_{ab}^{ij} = 1)$ 
5  $S_{ab}^{ij} \leftarrow Considered, \quad \forall \tilde{x}_{ab}^{ij} \in FWlist$ 
6 repeat
7    $\tilde{x}_{next} \leftarrow \tilde{x}_{ab}^{ij} \mid \max_{\tilde{x}_{ab}^{ij} \in FWlist} r_{ab}^{ij}$ 
8    $FWlist \leftarrow FWlist - \tilde{x}_{next}$ 
9    $S_{next} = Accepted$ 
10  for  $\tilde{x}_{ab}^{ij} \in \text{neigh}(\tilde{x}_{next}) \mid S_{ab}^{ij} \neq Accepted$  do
11     $r' \leftarrow Eq.(4.7)$ 
12    if  $r' > r_{ab}^{ij}$  then
13       $r_{ab}^{ij} \leftarrow r'$ 
14      if  $S_{ab}^{ij} = Far$  then
15         $S_{ab}^{ij} \leftarrow Considered$ 
16         $FWlist \leftarrow FWlist + \tilde{x}_{ab}^{ij}$ 
17 until  $FWlist = \{\}$ ;

```

state S_{ab}^{ij} set to Accepted and an initial risk value r_{ab}^{ij} of one, while the remaining ones have a state S_{ab}^{ij} set to Far and an initial risk value r_{ab}^{ij} of zero. As can be denoted, these states are also similar to those introduced in Chapter 3 to explain the Fast Marching Method (FMM). The main difference is that the state of local nodes is referred to as S_{ab}^{ij} , for any local node \tilde{x}_{ab}^{ij} , instead of S_{ij} . Thereafter, the same process as in GPP is done: a wave propagates according to the calculation of an eikonal equation. For each iteration, the node with the least value of risk from $FWlist$ is extracted. The process finishes when the list $FWlist$ becomes empty, as the wave either visits all nodes or reaches the distance d_{risk} from the obstacles.

Once the values of risk are properly set, the next step is to execute FM* to calculate the wave propagation from one of the waypoints of the original path. One of them is referred to as $\tilde{\Gamma}_{triggerer}$ (see Figure 4.5b). It is the first waypoint (with respect to the order the robot follows the waypoints) that is placed either in the risky area or in the obstacle area. Having this waypoint as reference, another one placed between it and the closest to the robot position serves as the origin of the propagating wave, including the latter one. Ideally, the starting waypoint, or $\tilde{\Gamma}_{start}$, would be located at a distance from $\tilde{\Gamma}_{triggerer}$ higher than d_{risk} . In this way, the repaired section of the path will connect to $\tilde{\Gamma}_{start}$ and the waypoints located prior to it will remain the same. The other point at which this local section will connect to will depend on the approach chosen to do the repairing. For both of them, another waypoint of interest is taken into account. This waypoint is $\tilde{\Gamma}_{reference}$. This is the first waypoint

that is located in the safe area after the path passes through the risky or the obstacle areas. According to the chosen approach, either its value of total cost (interpolated from the nearest global nodes from the global layer) or its location is used. The functioning of each approach and how do they make use of $\tilde{\Gamma}_{reference}$ is later explained. The heuristic FMM, or FM*, starts from $\tilde{\Gamma}_{start}$ computing values of the local total cost function $t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij})$. This function indicates what is the traversed distance from the starting position \tilde{x}_{start}^{ij} , which is the local node that is closest to $\tilde{\Gamma}_{start}$, to any local node \tilde{x}_{ab}^{ij} . The main idea is to find the shortest local section of path given the risk and the heuristic function, which depends on the chosen approach.

Algorithm 3 presents the pseudo-code of the FM* process using the local layer. It is similar to the functioning of the FMM, but with two major differences. First, the condition to stop the loop evaluates whether the last Accepted local node, \tilde{x}_{next}^{ij} , complies with (4.12) or (4.15) according to the repairing approach chosen (either Sweeping or Conservative). These two equation defining the conditions of both approaches are later explained. Second, the condition to decide which Considered local node is selected from *FWlist* to change its state to Accepted is different. Here it is not selected the local node with the lowest value of local total cost $t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij})$. Instead, the local node that is selected is the one with the lowest value of heuristic function h_{ab}^{ij} . The expression to define this heuristic function will depend on the chosen approach and is also later explained.

Algorithm 3: Local FM* Propagation

```

1  $t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}), S_{ab}^{ij} \leftarrow \infty, Far \ \forall \tilde{x}_{ab}^{ij} \in \text{Local Layer}$ 
2  $\tilde{x}_{start}^{ij} \leftarrow \tilde{x}_{ab}^{ij} \text{ closest to } \tilde{\Gamma}_{start}$ 
3  $t(\tilde{x}_{start}^{ij}, \tilde{x}_{start}^{ij}), S_{start}^{ij} \leftarrow 0, Accepted$ 
4  $FWlist \leftarrow \{\}$ 
5  $FWlist \leftarrow FWlist + \tilde{x}_{ab}^{ij} \ \forall \tilde{x}_{ab}^{ij} \in neigh(\tilde{x}_{start}^{ij}) \mid S_{ab}^{ij} = Far$ 
6  $t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}), S_{ab}^{ij} \leftarrow \min \left\{ t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}), Eq.(4.10) \right\}, Considered \ \forall \tilde{x}_{ab}^{ij} \in$ 
    $neigh(\tilde{x}_{start}^{ij}) \mid S_{ab}^{ij} = Far$ 
7 repeat
8    $\tilde{x}_{next}^{ij} \leftarrow \tilde{x}_{ab}^{ij} \mid \min_{\tilde{x}_{ab}^{ij} \in FWlist} h_{ab}^{ij}$ 
9    $FWlist \leftarrow FWlist - \tilde{x}_{next}^{ij}$ 
10   $S_{next}^{ij} \leftarrow Accepted$ 
11   $FWlist \leftarrow FWlist + \tilde{x}_{ab}^{ij} \ \forall \tilde{x}_{ab}^{ij} \in neigh(\tilde{x}_{next}^{ij}) \mid S_{ab}^{ij} = Far$ 
12   $S_{ab}^{ij} \leftarrow Considered, \ \forall \tilde{x}_{ab}^{ij} \in neigh(\tilde{x}_{next}^{ij}) \mid S_{ab}^{ij} = Far$ 
13   $t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}) \leftarrow \min \left\{ t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}), Eq.(4.10) \right\}, \ \forall \tilde{x}_{ab}^{ij} \in neigh(\tilde{x}_{next}^{ij}) \mid S_{ab}^{ij} =$ 
    $Considered$ 
14 until  $\tilde{x}_{next}^{ij}$  satisfies either (4.12) or (4.15);

```

The local cost function $c(\tilde{x}_{ab}^{ij})$ assigned to each node \tilde{x}_{ab}^{ij} of the local layer is hence determined by Equation (4.9). It depends on the value of risk r_{ab}^{ij} calculated before in the Risk Expansion process. The eikonal equation that determines the value of

local total cost $t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij})$ according to the local cost $c(\tilde{x}_{ab}^{ij})$ is shown in (4.10). In a similar way to the Risk Expansion and the GPP, the discretized explicit form of this equation takes the expression shown in (4.11).

$$c(\tilde{x}_{ab}^{ij}) = 1 + r_{ab}^{ij} \quad (4.9)$$

$$||\nabla t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij})|| = c(\tilde{x}_{ab}^{ij}) \quad (4.10)$$

$$t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}) = \begin{cases} \frac{t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij,x}) + t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij,y})}{2} + \frac{\sqrt{2(\lambda c(\tilde{x}_{ab}^{ij}))^2 - (t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij,x}) - t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij,y}))^2}}{2}, \\ \quad |t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij,x}) - t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij,y})| \leq \lambda c(\tilde{x}_{ab}^{ij}) \\ \min \{t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij,x}), t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij,y})\} + \lambda c(\tilde{x}_{ab}^{ij}), \quad \text{otherwise} \end{cases} \quad (4.11)$$

Sweeping approach

The first of the two approaches not only focuses on producing a path that surrounds the obstacle and allows the robot to avoid it. It also considers the values of global total cost that were previously computed in the global layer with the GPP process. With this in mind, the main premise of the Sweeping approach is to take advantage of the global computation to decide when it is better to stop the propagation. In other words, the heuristic here is not oriented to make the local path reach a certain position, but any position that complies with (4.12). Here, the position in question corresponds to that of the local node \tilde{x}_{next}^{ij} . Its associated value of global total cost is $T(\tilde{x}_{next}^{ij}, \tilde{x}_g)$, and results from calculating the interpolation with the values of total cost $T(\tilde{x}_{ij}, \tilde{x}_g)$ of the closest global nodes. It is key that this total cost is computed from a propagating wave that expanded from the goal \tilde{x}_g . The lower bound that \tilde{x}_{next}^{ij} has to reach is $T(\tilde{\Gamma}_{reference}, \tilde{x}_g)$. This is the total cost at the location of the global waypoint $\tilde{\Gamma}_{reference}$. This value also results from interpolating with the values of total cost of the nearby global nodes. Since the waypoint $\tilde{\Gamma}_{reference}$ was located on the other side of the obstacle, it serves as a reference to look for any other position ahead of the obstacle, with a similar or even lower value of total cost interpolated from the global nodes of the global layer.

$$T(\tilde{x}_{next}^{ij}, \tilde{x}_g) \leq T(\tilde{\Gamma}_{reference}, \tilde{x}_g) \quad (4.12)$$

Moreover, it is also checked that the next global waypoints that result from \tilde{x}_{next} do not fall into the obstacle area when making the gradient descent method in the global layer. Due to how the gradient descent method works, these new waypoints will follow values of total cost in decreasing order as indicated in Equation (3.33)

from Chapter 3. Therefore, this approach focuses on finding a local node from which a new path on the global layer is calculated towards the goal as well as on the local layer towards $\tilde{\Gamma}_{start}$.

The heuristic function h_{ab}^{ij} used by the Sweeping approach prioritizes the search towards local nodes whose values of total cost are closer to $T(\tilde{\Gamma}_{reference}, \tilde{x}_g)$ or even are lower than it. This is translated into using Equation (4.13) as such heuristic function. Here, the variable χ is the length in the portion of path going from $\tilde{\Gamma}_{start}$ to $\tilde{\Gamma}_{reference}$, and is defined in (4.14) as the sum of the distances between the waypoints that are placed between those two. In order to avoid negative values, the heuristic function returns a minimum value of $t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij})$ if the found local nodes have already a value of local total cost lower than that of $\tilde{\Gamma}_{reference}$. This could happen as there may be local nodes that, although satisfying the condition shown in (4.12), the global waypoints that would be calculated from their respective location would re-enter into risky or even obstacle areas. This fact consequently encourages the algorithm to keep searching.

$$h_{ab}^{ij} = t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}) + \max \left\{ 0, \frac{t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}) - T(\tilde{x}_{reference}^{ij}, \tilde{x}_g)}{T(\tilde{x}_{start}^{ij}, \tilde{x}_g) - T(\tilde{x}_{reference}^{ij}, \tilde{x}_g)} \chi \right\} \quad (4.13)$$

$$\chi = \sum_{k \leftarrow \tilde{\Gamma}_k = \tilde{\Gamma}_{start}}^{k-1 \leftarrow \tilde{\Gamma}_k = \tilde{\Gamma}_{reference}} ||\tilde{\Gamma}_{k+1} - \tilde{\Gamma}_k|| \quad (4.14)$$

Conservative approach

The objective of the Conservative approach is to find the local path that connects the position of the robot with one of the global waypoints of the original path, placed on the other side of the detected obstacle. The main difference of this approach with respect to the previous one is that the local propagation of the FM* stops at a certain local node instead of at any local node that complies with condition (4.12). The local node at which this propagation stops is $\tilde{x}_{reference}^{ij}$, which corresponds to the one that is placed the closest to the position of $\tilde{\Gamma}_{reference}$. This means the propagation loop stops when it reaches the local node $\tilde{x}_{reference}^{ij}$ as expressed in (4.15), after setting its state as Accepted.

$$\tilde{x}_{next}^{ij} = \tilde{x}_{reference}^{ij} \quad (4.15)$$

Figure 4.5d shows how the local path generated using the Conservative approach does not create a smooth transition towards the global waypoints. This is mainly because the local propagation does not expand towards lower values of global total cost, i.e. towards the direction of the gradient descent. This causes that the created local waypoints are not necessarily parallel to this gradient descent direction. Therefore, opting for this approach prioritizes finding a local section of path that reaches

the original one, entailing longer local paths than those created with the Sweeping approach. The Conservative approach can be desired in cases where the robot must stick to the original plan, when the initial path was not produced by the GPP process (and hence not having available values of global total cost) or when using the global layer locally is either not possible or not a preferable option. The latter can be justified if the size of the global layer is too big, or if it is desired not to spend computation on interpolating the global total cost associated to the local nodes, i.e. $T(\tilde{x}_{ij}^{ij}, \tilde{x}_g)$. Another case would be the one in which the global path serves to cover an area, obtained thanks to a coverage path planning method. In this case, the Conservative approach would serve to make the rover account for obstacles while covering an extensive area. Moreover, this approach of LPR would serve simply as a local planner, just by considering a local destination instead of the location of a global waypoint to rejoin another path.

The heuristic function used by the Conservative approach is simply a distance heuristic. In other words, this heuristic function prioritizes the propagation towards local nodes that are closer to $\tilde{x}_{reference}^{ij}$. This function is expressed in equation (4.16).

$$h_{ab}^{ij} = t(\tilde{x}_{start}^{ij}, \tilde{x}_{ab}^{ij}) + ||\tilde{x}_{ab}^{ij} - \tilde{x}_{reference}|| \quad (4.16)$$

4.5 Summary and Conclusions

This chapter presents and details the second contribution of this thesis, a path planning solution named DyMu. It is based on the use of PDE planners, in particular the FMM and its heuristic version FM*. The presented solution provides this kind of method with the capability to dynamically update the path in scenarios with some grade of uncertainty. This is a feature that was missing in this kind of algorithms, but their use is justified by the fact that the path they generate is smooth, continuous and globally optimal.

DyMu works with a multi-resolution grid called the multi-layered grid. This grid contains two layers of different size and resolution. Each grid layer correspond to a different scale of planning, either global or local. The global layer serves to generate a plan for a long traverse between the position of a robot and a desired destination. This initial plan, in the form of a path, depends on the information about the terrain and the locomotion modes available in the robot. Thereafter, this path is updated each time the robot detects an obstacle on its way. This update process is computed on the local layer and is referred to as the Local Path Repairing (LPR). This chapter presents two approaches to LPR: the Sweeping and the Conservative. The Sweeping approach resembles a bi-directional version of the FMM, where two waves are propagated from the robot location and the destination respectively, but with two differences. First, one of the waves is already calculated. It is the solution calculated by the Global Path Planning (GPP). Second, the other wave is calculated using a different layer, the local layer. The second wave starts from the robot position (or a

position close to it), reaching a local node that complies with certain requirements. The resulting path is formed by sections calculated in both the local and the global layer using the gradient descent method. The other approach, the Conservative, only relies on the computation done in the local layer to produce a path connecting the robot position (or close to it) and another location, such as one of the global waypoints from a pre-planned path.

The solution presented in this chapter has still a margin to improve in both the GPP and the LPR processes. In the case of the GPP, an anisotropic algorithm such as the one presented in the next chapter (Chapter 5), the Ordered Upwind Method (OUM), could be integrated. In this way, the planner could account for the slopes existing in the scenario and produce an optimal path that takes gravity into account. In its current state, the LPR only repairs the path to avoid obstacles. However, it seems interesting to extend this process to account for restrictions in turning manoeuvres as well as to optimize the local traverse given the local information about the terrain, similarly as it is done in the GPP process.

Chapter 5

Anisotropic Cost Model for Traversing Inclined Terrains

"Gravity is a habit that is hard to shake off."

Terry Pratchett
Small Gods, 1992

5.1 Introduction

The locomotion of a mobile robot determines its capability to adapt to different terrains, i.e. its adaptability. Rough terrain may pose a challenge to the locomotion subsystem due to the presence of irregularities on the surface. For instance, these irregularities can exist in the form of inclined surfaces or slopes. The inclination affects the robot as it is pulled by gravity in the direction of the slope. This situation makes the robot experience an anisotropic phenomenon: the power consumption varies with the heading direction of the robot. This means that the robot invests different amounts of energy into driving depending on whether it is ascending, descending or following a direction in-between. This can also be understood by analyzing the situation in terms of potential energy: when the robot ascends, it must reach a state with higher potential energy and therefore the consumption increases. On the contrary, when the robot descends it liberates part of this potential energy, resulting in reducing the energetic cost of the drive. Furthermore, the inclination of the terrain can also make it dangerous to traverse. This is because the pose of the robot body may be affected by this inclination, compromising the stability of the whole robot as a consequence. In an extreme case, the robot could overturn and, in the absence of any recovery system capable to handle this situation, put an end to the mission.

In the case of planetary exploration, rovers drive not only on horizontal but also on inclined terrains. For instance, the Spirit rover was commanded over weeks to climb the Husband summit on Mars and later descend it (Arvidson, Bell, et al., [2010](#)). On the other hand, Search And Rescue (SAR) operations may be carried out on inclined terrains, like the side of a mountain.

The majority of existing path planning approaches consider the cost function as isotropic for simplification purposes. For example, the FMM can be used along with a cost function that only depends on the gradient of any slope (Miró et al., 2010). The main drawback of the isotropic approach is that the resulting path does not acknowledge the differences in the cost due to the robot heading. For this reason, the generated path can be either too conservative (missing better paths) or too permissive (assuming undesirable risks or energetic costs). An Optimal Control PDE Solving algorithm, the Ordered Upwind Method (OUM), was previously used to address the risk of tipping over in the lateral and the longitudinal axes of the robot in the work presented by Shum et al. (2015). They also introduced in this work the bi-OUM, a bi-directional version of OUM. This version keeps the compatibility with anisotropic cost functions while it speeds up the computation of OUM.

Gravity not only affects the safety of the robot but also terramechanic processes such as the slippage. Previous work addressed the slippage when finding the optimal ascent of slopes, using the Sampling Based algorithm RRT* (H. Inotsume et al., 2020). This slip changes according to whether the robot drives straightly (in the direction of maximum ascent) or diagonally (Hiroaki Inotsume et al., 2016). A simplification to this approach consists of only considering elevation changes as proposed by Gruning et al. (2020), together with the Sampling Based SBMPO algorithm. Other approaches consider the effect of gravity on the friction along with Graph Search algorithms, including Dijkstra (Z. Sun et al., 2005) and some heuristic variants (Rowe et al., 1990; Choi et al., 2012; Ganganath, Cheng, and Chi, 2015; Ganganath, Cheng, Fernando, et al., 2018). These approaches using Graph Search algorithms do not ensure that the global optimal is found, since the optimality is bounded by the grid topology as explained in Chapter 2. Nevertheless, these approaches can address discontinuities in the cost function. Examples of these discontinuities include the definition of forbidden directions (e.g. when ascending a too pronounced slope) and the consideration of using zig-zag manoeuvres (Ganganath, Cheng, and Chi, 2015). Local Optimization algorithms based on the use of nonlinear programming consider dynamic constraints while using a Graph-Search as a global planner (Howard et al., 2007).

Other approaches combine slip, friction and risk. Sakayori et al. (2017) proposed a planner that generates bezier curves from a neural network to climb inclined surfaces after tuning some terramechanic parameters. However, a planner combining these parameters in a continuous anisotropic fashion was not present in the reviewed literature. For this reason, the contribution presented in this chapter is based on the use of a cost function usable by an Optimal Control path planning algorithm, a PDE Solving one, that addresses slip, friction and risk, and produces the optimal path. In particular, this cost function is used along with the OUM, being anisotropic, continuous and smooth. In other words, this cost function depends on the direction and is fully continuous and differentiable. The name of this cost function is the Continuous Anisotropic Model for Inclined Surfaces (CAMIS), and

it tackles the effect of gravity, friction, slip and the risk of overturning. The anisotropic planner, using CAMIS, can produce a path that is optimal in the sense that it minimizes energy consumption and/or preserves the robot stability.

This chapter is structured as follows. Section 5.2 presents the fundamentals about the PDE Solving path planning and how it is formulated as an optimization problem using the anisotropic cost function. Later on, Section 5.3 introduces the details about how the OUM works, in particular its bi-directional version. Section 5.4 presents the mathematical background behind the modelling of the proposed anisotropic cost function, as well as how it is adapted to the use of terramechanic parameters by using the locomotion models introduced in Chapter 3. Finally, Section 5.5 presents a summary of this chapter and shows the conclusions extracted from it, along with suggestions for possible future work.

5.2 Problem Formulation

This section details how to produce the optimal path by formulating the path planning problem using an anisotropic PDE. This is done in the scope of a robot that must autonomously navigate through rough irregular terrain. In a similar way to how Chapter 3 introduces the isotropic approach, here the region of interest for the path planning problem is named $\Omega \subset \mathbb{R}^2$. This is the portion of the environment enclosed by the user, who selects what area is relevant for the path planning problem. This area is discretized into the grid $\tilde{\Omega}$. This grid is considered here a regular one, and it can be either square or hexagonal (see Figure 3.1 in Chapter 3). The target of the path planner is to find the optimal path $\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$ connecting the goal node \tilde{x}_g and the origin node \tilde{x}_o . The optimality of this path comes from the minimization of the total cost function. This function, as in Chapter 3, is either the minimal amount of cost from the origin position or the remaining amount to reach the goal. Each definition is expressed through Equation (5.1) and Equation (5.2) respectively. The planner has to calculate one of these expressions of total cost when visiting each node, according to which initial condition is used. As a reminder, the total cost from the origin node \tilde{x}_o takes a value of zero in it, $T(\tilde{x}_o, \tilde{x}_{ij} = \tilde{x}_o) = 0$, while the remaining total cost has the initial condition set in the goal node \tilde{x}_g , having $T(\tilde{x}_{ij} = \tilde{x}_g, \tilde{x}_g) = 0$.

$$T(\tilde{x}_o, \tilde{x}_{ij}) = \min_{\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot) \in \Omega} \left\{ \int_0^{s_{ij}} Q(\Gamma(\tilde{x}_o, \tilde{x}_g, s), \vec{u}(\Gamma(\tilde{x}_o, \tilde{x}_g, s))) ds \right\}, \quad T(\tilde{x}_o, \tilde{x}_{ij} = \tilde{x}_o) = 0 \quad (5.1)$$

$$T(\tilde{x}_{ij}, \tilde{x}_g) = \min_{\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot) \in \Omega} \left\{ \int_{s_{ij}}^{s_g} Q(\Gamma(\tilde{x}_o, \tilde{x}_g, s), \vec{u}(\Gamma(\tilde{x}_o, \tilde{x}_g, s))) ds \right\}, \quad T(\tilde{x}_{ij} = \tilde{x}_g, \tilde{x}_g) = 0 \quad (5.2)$$

The main difference between the anisotropic expressions (5.1) and (5.2) with respect to the isotropic ones presented in (3.24) and (3.25) is the definition of the cost

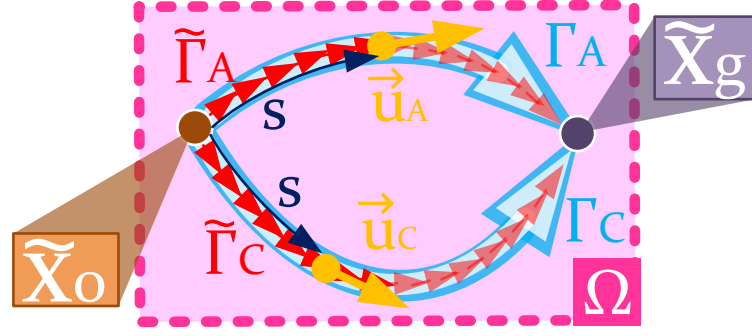


FIGURE 5.1: Different paths $\Gamma = \Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$ connecting \tilde{x}_o and \tilde{x}_g can be defined upon different tangent directions $\vec{u}(\Gamma)$. The role of the anisotropic path planner is to find the optimal path among them.

function. While in the isotropic case the cost function $C(x)$ only depends on position (in this case returned by the path function $x = \Gamma(\tilde{x}_o, \tilde{x}_g, s)$), in the anisotropic case the cost function $Q(x, \vec{u}(x))$ also depends on a direction $\vec{u}(x)$. This function, defined according to a position $x \in \Omega$, is the direction tangent to the optimal path that passes through $x = \Gamma(\tilde{x}_o, \tilde{x}_g, s)$. This is expressed in (5.3), where s_g is the total length of the path between \tilde{x}_o and \tilde{x}_g . Function $\vec{u}(x)$ is also called the characteristic direction of that location x . In this way, given how the anisotropic cost function is defined, all the optimal paths passing through the location in question will have this same characteristic direction on it. Therefore, the anisotropic cost function not only takes into account where the path is located to consider if it is optimal among all infinite possibilities, but also its direction. Figure 5.1 graphically depicts this situation in which the origin and goal nodes can be connected by many different paths and only one of them, given its shape $\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot)$ and direction $\vec{u}(\Gamma(\tilde{x}_o, \tilde{x}_g, \cdot))$ is the optimal one. The planner must hence not only calculate the value of total cost for each node \tilde{x}_{ij} but also the characteristic direction $\vec{u}(\tilde{x}_{ij})$.

$$\vec{u}(\Gamma(\tilde{x}_o, \tilde{x}_g, s)) = \frac{d\Gamma(\tilde{x}_o, \tilde{x}_g, s)}{ds} \quad \forall s \in (0, s_g) \quad (5.3)$$

To solve the expressions presented in (5.1) and (5.2), they are reformulated as the Hamilton-Jacobi-Bellman (HJB) equation, following the reasoning of previous works (James A Sethian and Vladimírsky, 2003; Shum et al., 2015; Shum et al., 2016). This equation, shown in (5.4) for both equivalent approaches of the total cost, establishes the correspondence between the anisotropic cost $Q(\tilde{x}_{ij}, \vec{\psi})$ and the spatial gradient of the total cost. Here, the anisotropic cost function is defined not only according to the location of any grid \tilde{x}_{ij} but also to the heading function $\vec{\psi}$. The latter function is the direction of the robot in the XY-plane, i.e. its heading. The direction $\vec{\psi}$ that complies with (5.4) in a grid node \tilde{x}_{ij} corresponds hence to the characteristic direction $\vec{u}(\tilde{x}_{ij})$ passing through it. Finally, it is worth mentioning that the eikonal equation presented in Chapter 3 is a particular case of the HJB in which the characteristic direction is equal to the negative gradient of the total cost.

$$\begin{aligned}
\vec{u}(\tilde{x}_{ij}) &= \vec{\psi} \mid \min_{\vec{\psi}} \{ \nabla T(\tilde{x}_o, \tilde{x}_{ij}) \cdot \vec{\psi} + Q(\tilde{x}_{ij}, \vec{\psi}) \} = 0, \quad \forall \tilde{x}_{ij} \in \tilde{\Omega} \\
&= \vec{\psi} \mid \min_{\vec{\psi}} \{ \nabla T(\tilde{x}_{ij}, \tilde{x}_g) \cdot \vec{\psi} + Q(\tilde{x}_{ij}, \vec{\psi}) \} = 0, \quad \forall \tilde{x}_{ij} \in \tilde{\Omega}
\end{aligned} \tag{5.4}$$

5.3 Optimal Path Planning using the Ordered Upwind Method

In an analogous way to the FMM in the resolution of the eikonal equation, the Ordered Upwind Method (OUM) is the PDE Solving algorithm chosen to tackle (5.4) and find the optimal path. According to the work of James A Sethian and Vladimirsky (2003), this algorithm has a computational complexity that depends on the anisotropy Y of the cost function: $O(Y n_{nodes} \log(n_{nodes}))$, where n_{nodes} is the number of grid nodes. This anisotropy comes as the ratio of the highest and the lowest values of cost. However, for the case presented in this chapter the anisotropy varies from node to node, i.e. the anisotropy here is $Y(\tilde{x}_{ij})$. Therefore, the computational complexity is expected to be variable as well, bounded by the maximum existing anisotropy in the grid $\tilde{\Omega}$. According to equation (5.5), this anisotropy $Y(\tilde{x}_{ij})$ of a node \tilde{x}_{ij} comes as the ratio between the highest possible cost at its location and the lowest according to the heading direction $\vec{\psi}$.

$$Y(\tilde{x}_{ij}) = \frac{\max_{\vec{\psi}} \{ Q(\tilde{x}_{ij}, \vec{\psi}) \}}{\min_{\vec{\psi}} \{ Q(\tilde{x}_{ij}, \vec{\psi}) \}} \tag{5.5}$$

The computational complexity of the OUM justifies the use of a bi-directional version that speeds up the calculation of the total cost. This bi-directional feature exploits the fact that the total cost complies with the Dynamic Programming Principle (DPP) as was shown in Equation (3.23) presented back in Chapter 3. As a reminder, this equation is presented again here in (5.6), but using \tilde{x}_{ij} instead of x to indicate this principle is exploited with the bi-OUM (as the bi-directional OUM but shortened) using the grid $\tilde{\Omega}$.

$$T(\tilde{x}_o, \tilde{x}_{ij}) + T(\tilde{x}_{ij}, \tilde{x}_g) = T(\tilde{x}_o, \tilde{x}_g) \quad \forall \tilde{x}_{ij} \in \Gamma(\tilde{x}_o, \tilde{x}_g, \cdot) \in \tilde{\Omega} \tag{5.6}$$

Bi-OUM calculates both $T(\tilde{x}_o, \tilde{x}_{ij})$ and $T(\tilde{x}_{ij}, \tilde{x}_g)$ using two propagating waves in two parallel loops. Figure 5.2 shows this functioning with a hexagonal grid (although a square grid would be valid too). Each parallel loop is coloured with a different colour, either purple (the wave starting from the goal node \tilde{x}_g , setting a total cost of zero on it) or orange (the wave starting from the origin node \tilde{x}_o , setting a total cost of zero on it). As can be checked in this figure, is not necessary that the two waves propagate until reaching the starting position of the other wave. When both waves reach an intermediate linking node \tilde{x}_l the bi-OUM process stops. This is because, as highlighted in Equation 5.6, this node complies with the DPP and there

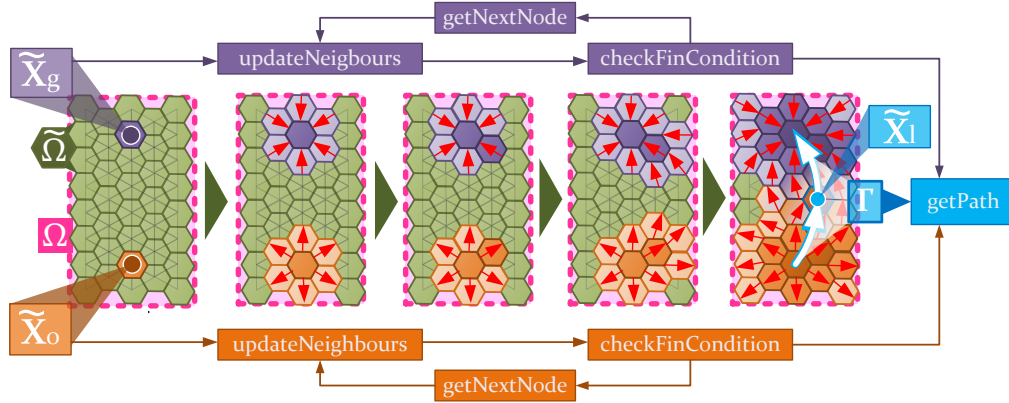


FIGURE 5.2: Schematic of the functioning of biOUM. Two expanding waves start from the goal and the origin nodes (\tilde{x}_g and \tilde{x}_o) respectively. The loop controlling each wave is colored in purple and orange. The red arrows indicate the characteristic direction $\vec{u}(\tilde{x}_{ij})$ that passes through each node.

is only a single path passing through it: the optimal path. This path is calculated by following the characteristic direction that is also calculated in both waves, one from \tilde{x}_l to \tilde{x}_o and another from \tilde{x}_l to \tilde{x}_g . In this way, bi-OUM achieves the same optimal solution as the OUM but investing less time than the latter, as the bi-directional version visits less nodes (Shum et al., 2015).

The process followed by bi-OUM to visit the grid nodes is explained with the help of the conceptual scheme presented in Figure 5.2 together with the pseudo-code contained in Algorithm 4. First of all, the origin node \tilde{x}_o and the goal node \tilde{x}_g are selected out of $\tilde{\Omega}$. From each of these nodes a parallel process is initiated, concurrently visiting neighbouring nodes and updating their states. In the bi-directional version of OUM each node has two states: S_{ij}^o and S_{ij}^g . Each of these states points out the status of the node \tilde{x}_{ij} according to the execution of the respective loop associated with the respective set. There are four options that each of the states can take:

- **Far.** Nodes with this state have not been accessed yet by the corresponding loop. This is the state in which all nodes, except the starting ones (the origin and the goal), are initialized at the beginning of the process for both S_{ij}^o and S_{ij}^g . As the real value of total cost associated to each Far node is yet to be calculated, the initial value is irrelevant. Nevertheless, for this particular case, this value is set to ∞ symbolizing a very high total cost. Besides, the characteristic direction for Far nodes is initialized to a Null value or NaN (Not a Number).
- **Considered.** A tentative value of total cost has been computed by the corresponding loop at least once for nodes with this state. This tentative value is calculated by means of the `updateNeighbours` function. This function is called to take the neighbouring Far nodes from an `AcceptedFront` node and compute their respective values of total cost and characteristic direction. Thereafter, the state of these nodes is updated and set as **Considered**.

- **AcceptedFront.** During each iteration of each loop, the function getNextNode is executed to retrieve the Considered node with the lowest value of total cost. The value of this node is no longer tentative. Thereafter, the state of this node is updated to AcceptedFront if it has Far or Considered nodes as neighbours, otherwise it is updated to AcceptedInner. In case of being an AcceptedFront node, the updateNeighbours function is called to update the state and values of total cost and characteristic direction of neighbouring Far nodes. Furthermore, the starting nodes are initialized to the AcceptedFront state, with initial values of total cost and characteristic direction of zero and NaN respectively. It is assumed that the heading directions at the origin and the destination are not considered, and hence the value of NaN is assigned to their respective characteristic directions.
- **AcceptedInner.** Nodes with this state do not only have associated definitive values of total cost and characteristic direction, but also all of their neighbours. In other words, the neighbours of AcceptedInner nodes are either AcceptedFront or also AcceptedInner nodes. This is a condition that is checked for all AcceptedFront during each iteration to update their state to AcceptedInner in the affirmative case.

Algorithm 4: The bi-directional Ordered Upwind Method.

```

1 Input:  $\tilde{x}_0, \tilde{x}_g, \tilde{\Omega}$ 
2 Output:  $\tilde{\Gamma}$ 
3  $T(\tilde{x}_{ij}, \tilde{x}_g), \tilde{u}(\tilde{x}_{ij}), S_{ij}^g \leftarrow \infty, NaN, Far \quad \forall \tilde{x}_{ij} \in \tilde{\Omega}$ 
4  $T(\tilde{x}_g, \tilde{x}_g), \tilde{u}(\tilde{x}_g), S_g^g \leftarrow 0, NaN, AcceptedFront$ 
5  $T(\tilde{x}_0, \tilde{x}_{ij}), \tilde{u}(\tilde{x}_{ij}), S_{ij}^o \leftarrow \infty, NaN, Far \quad \forall \tilde{x}_{ij} \in \tilde{\Omega}$ 
6  $T(\tilde{x}_0, \tilde{x}_0), \tilde{u}(\tilde{x}_0), S_o^o \leftarrow 0, NaN, AcceptedFront$ 
7  $\tilde{x}_{tg} \leftarrow \tilde{x}_g$ 
8  $\tilde{x}_{to} \leftarrow \tilde{x}_0$ 
9 while  $\neg checkFinCondition(S_{tg}^o, S_{tg}^g) \wedge \neg checkFinCondition(S_{to}^o, S_{to}^g)$  do
10      $\triangleright$  Update both loops
11      $updateNeighbours(\tilde{x}_{tg})$ 
12      $updateNeighbours(\tilde{x}_{to})$ 
13      $\triangleright$  Get next nodes
14      $\tilde{x}_{tg} \leftarrow getNextNode(T(\cdot, \tilde{x}_g))$ 
15      $\tilde{x}_{to} \leftarrow getNextNode(T(\tilde{x}_0, \cdot))$ 
16 return  $getPath(S_g, S_0)$ 

```

During a visit to each newly AcceptedFront node, it is evaluated by the corresponding loop using the updateNeighbours function. This function evaluates the values of total cost and the characteristic direction of nodes with a state S_{ij}^o or S_{ij}^g equal to Considered (with respect to the calculation loop that executes the function). These nodes must be close enough to \tilde{x}_{ij} , located at a distance lower than $\zeta(\tilde{x}_{ij})$ from it. This distance is defined in Equation (5.7) and comes in function of the anisotropy

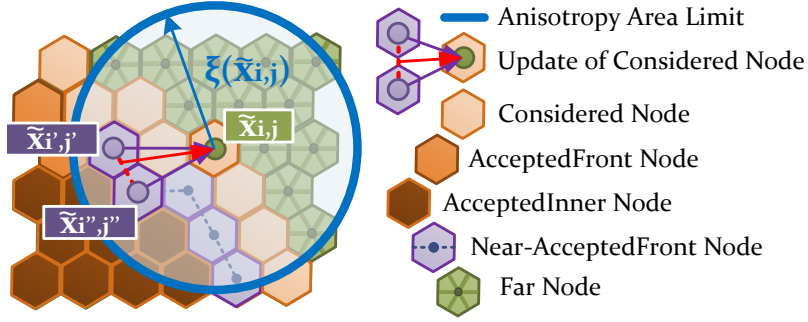


FIGURE 5.3: Update process of a Considered node \tilde{x}_{ij} . Its values of total cost and characteristic direction are computed taking into account AcceptedFront nodes within the distance $\zeta(\tilde{x}_{ij})$ expressed in Equation (5.7).

that is present at such node and the grid resolution Λ . Figure 5.3 shows how this distance works. Here, Near-AcceptedFront nodes are those AcceptedFront nodes that comply with the mentioned distance condition between them and the Considered node.

$$\zeta(\tilde{x}_{ij}) = \Lambda Y(\tilde{x}_{ij}) \quad (5.7)$$

The update of the total cost and the characteristic direction of a Considered node, when the updateNeighbours function is called, can be done using a semi-lagrangian discretization of the HJB presented in Equation (5.4). This method was already used by James A Sethian and Vladimirovsky (2003) when they introduced the OUM for the first time. Equations (5.8) and (5.9) present the resulting expressions to calculate the value of total cost, based on the AcceptedFront nodes $\tilde{x}_{i',j'}$ and $\tilde{x}_{i'',j''}$ and according to the starting node of the propagating wave. The main difference between them is that the total cost from the goal node has to consider that the anisotropic cost function multiplies the input value of characteristic direction by -1 . This is because the propagating wave advances in the contrary direction to the heading of the robot. Equation (5.10) expresses how the characteristic direction is calculated. The iterative process calculates either (5.8) or (5.9) together with (5.10), using a parameter called ϵ that takes values between zero and one and is updated in each iteration. A drawback of this method is that it can take more time than solving an explicit quadratic expression such as the discretized eikonal equations presented in Chapter 3.

$$T(\tilde{x}_o, \tilde{x}_{ij}) = \min_{\epsilon \in [0,1]} \{Q(\tilde{x}_{ij}, \vec{u}(\tilde{x}_{ij}))|\epsilon \tilde{x}_{i',j'} + (1 - \epsilon) \tilde{x}_{i'',j''} - \tilde{x}_{ij}| + \epsilon T_{i',j'} + (1 - \epsilon) T_{i'',j''}\} \quad (5.8)$$

$$T(\tilde{x}_{ij}, \tilde{x}_g) = \min_{\epsilon \in [0,1]} \{Q(\tilde{x}_{ij}, -\vec{u}(\tilde{x}_{ij}))|\epsilon \tilde{x}_{i',j'} + (1 - \epsilon) \tilde{x}_{i'',j''} - \tilde{x}_{ij}| + \epsilon T_{i',j'} + (1 - \epsilon) T_{i'',j''}\} \quad (5.9)$$

$$\vec{u}(\tilde{x}_{ij}) = \frac{\epsilon \tilde{x}_{i'j'} + (1 - \epsilon) \tilde{x}_{i''j''} - \tilde{x}_{ij}}{\| \epsilon \tilde{x}_{i'j'} + (1 - \epsilon) \tilde{x}_{i''j''} - \tilde{x}_{ij} \|} \quad (5.10)$$

As mentioned, for the isotropic case, where $Y(\tilde{x}_{ij}) = 1$, the total cost can be directly calculated using an explicit expression. It could be either Equation (3.30) or Equation (3.32) for a square grid or a hexagonal grid respectively. These expressions are referred to as the eulerian discretization of the eikonal equation, and they are used by the FMM as explained in Chapter 3. Nevertheless, as the path can be extracted using the gradient descent method the characteristic direction does not need to be calculated in FMM. For the case of the OUM, isotropic nodes can be present together with anisotropic ones. For this reason, in case of using the eulerian discretization, there must be also an expression to directly calculate $\vec{u}(\tilde{x}_{ij})$ in the absence of the ϵ parameter used in (5.11). This expression is shown in (5.11) for the loop that calculates the total cost from the origin and for the loop that calculates the total cost remaining to the goal.

$$\vec{u}(\tilde{x}_{ij}) = \begin{cases} \frac{\begin{bmatrix} \tilde{x}_{ij} - \tilde{x}_{i'j'} \\ \tilde{x}_{ij} - \tilde{x}_{i''j''} \end{bmatrix}^{-1} \begin{bmatrix} T_{ij} - T_{i'j'} \\ T_{ij} - T_{i''j''} \end{bmatrix}}{\left\| \begin{bmatrix} \tilde{x}_{ij} - \tilde{x}_{i'j'} \\ \tilde{x}_{ij} - \tilde{x}_{i''j''} \end{bmatrix}^{-1} \begin{bmatrix} T_{ij} - T_{i'j'} \\ T_{ij} - T_{i''j''} \end{bmatrix} \right\|}, & \text{using } T_{ij} = T(\tilde{x}_o, \tilde{x}_{ij}) \wedge Y(\tilde{x}_{ij}) = 1 \\ \frac{\begin{bmatrix} \tilde{x}_{ij} - \tilde{x}_{i'j'} \\ \tilde{x}_{ij} - \tilde{x}_{i''j''} \end{bmatrix}^{-1} \begin{bmatrix} T_{i'j'} - T_{ij} \\ T_{i''j''} - T_{ij} \end{bmatrix}}{\left\| \begin{bmatrix} \tilde{x}_{ij} - \tilde{x}_{i'j'} \\ \tilde{x}_{ij} - \tilde{x}_{i''j''} \end{bmatrix}^{-1} \begin{bmatrix} T_{i'j'} - T_{ij} \\ T_{i''j''} - T_{ij} \end{bmatrix} \right\|}, & \text{using } T_{ij} = T(\tilde{x}_{ij}, \tilde{x}_g) \wedge Y(\tilde{x}_{ij}) = 1 \end{cases} \quad (5.11)$$

Finally, the checkFinCondition function checks during each iteration if the last node returned by getNextNode is either AcceptedInner or AcceptedFront for both loops. In the affirmative case, this node becomes \tilde{x}_l and the iteration process stops for both loops. Indicated in Figure 5.2 as a blue dot, this node is used to extract two sub-paths with the getPath function. Each of these portions starts from the location of \tilde{x}_l and reaches either the origin or the goal nodes, \tilde{x}_o or \tilde{x}_g . The waypoints making up both sub-paths are calculated one after another from $\tilde{\Gamma}_l = \tilde{x}_l$ following the two ways indicated in (5.12). The step distance is d_{step} , and the characteristic direction for each waypoint is interpolated from the values calculated on the nodes. The final path $\tilde{\Gamma}$ is obtained by joining them (see Figure 5.2).

$$\tilde{\Gamma}_{k-1} = \tilde{\Gamma}_k - d_{step} \vec{u}(\tilde{\Gamma}_k), \quad \tilde{\Gamma}_{k+1} = \tilde{\Gamma}_k + d_{step} \vec{u}(\tilde{\Gamma}_k) \quad (5.12)$$

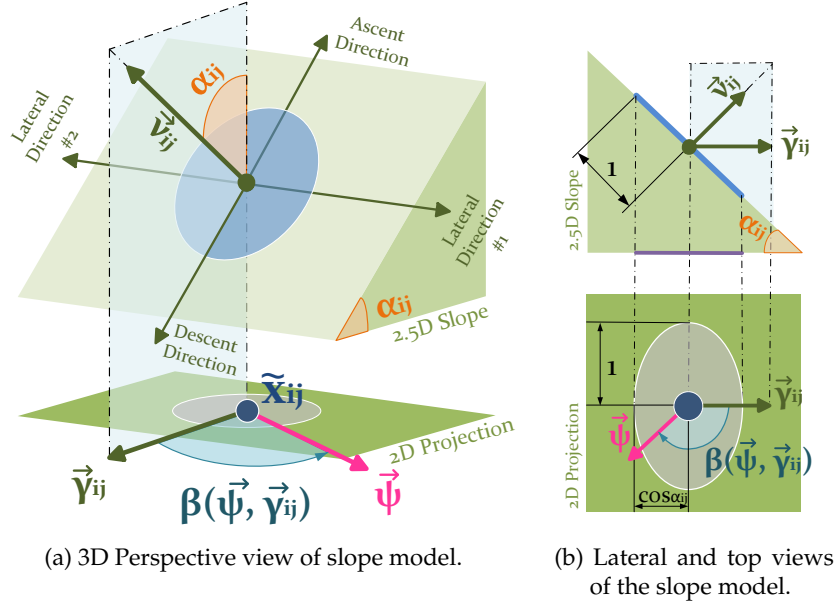


FIGURE 5.4: Graphical representation of the slope model used to build up the anisotropic cost function. This model encompasses multiple variables such as the slope gradient α_{ij} , the aspect direction $\vec{\gamma}_{ij}$, the normal vector \vec{v}_{ij} , the heading direction $\vec{\psi}$ and the relative angle $\beta(\vec{\psi}, \vec{\gamma}_{ij})$. All of them are marked in this conceptual depiction.

5.4 Anisotropic Cost Function for Planning on Inclined Surfaces

This section presents the steps to make an anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$ based on slope parameters and compatible with the OUM. Figure 5.4 shows the slope model used to represent the interaction between the robot and a inclined terrain. In the absence of any kinematic configuration capable to reconfigure itself (Brunner et al., 2015), the pose of the robot body will change according to the shape of the terrain surface. This change will be determined by the contact points between the robot and the surface. Nevertheless, to avoid making the formulation more complex a simplification is made: the robot-terrain interaction is modelled after a single contact point. This simplification assumes the robot body is always parallel to an imaginary inclined plane. The normal vector of this plane, named \vec{v}_{ij} in Figure 5.4, will be hence coincident with the Z-axis of the robot local reference frame. This imaginary plane is inclined a certain magnitude from the horizontal XY-plane (the plane perpendicular to the gravity vector). The value of this magnitude corresponds to the slope gradient α_{ij} , and is equivalent to the angle between the normal vector \vec{v}_{ij} and the global Z-axis as showcased in Figure 5.4a. To clarify, this global Z-axis corresponds to the normal vector of the 2D projection. The direction the slope faces, i.e. the direction in which the steepest descent occurs, is the slope ascent or $\vec{\gamma}_{ij}$. It can be obtained from projecting the normal vector of the slope, \vec{v}_{ij} , onto the 2D XY-plane and normalizing it.

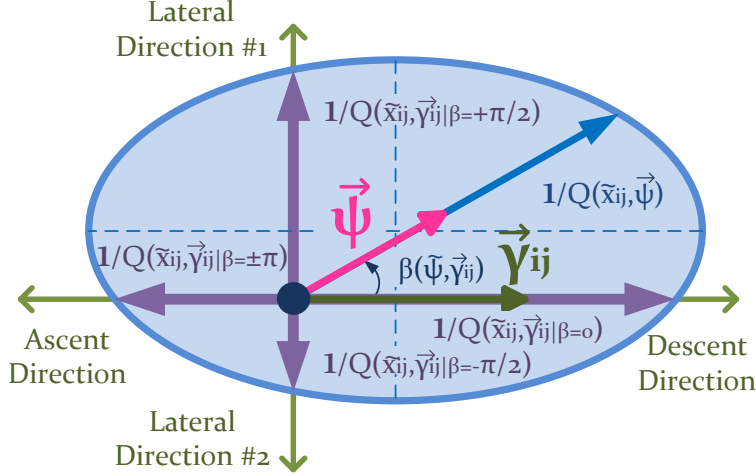


FIGURE 5.5: Elliptical inverse of the anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$. The descent direction is the one coincident with the slope aspect $\vec{\gamma}_{ij}$. The robot has a heading direction $\vec{\psi}$ that determines which value of this anisotropic cost is returned.

To make the anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$ compatible with the (bi-)OUM, it is necessary to understand its requirements. The solution that bi-OUM produces is viscous, as the FMM does when solving a path planning problem formulated with the eikonal equation. As a reminder, this means that the solution exists, is unique and is stable. This is thanks to the fact that the computed solution omits any discontinuity present in the real solution of the total cost (Shum et al., 2016). However, the computed solution is guaranteed to be unique if, and only if, it is ensured that the inverse of the cost function, $1/Q(\tilde{x}_{ij}, \vec{\psi})$, is fully differentiable and convex (James A Sethian and Vladimírsky, 2003; Shum et al., 2016). A closed conic curve complies with this condition, while at the same time it can be formulated using a mathematical expression. For this reason, $1/Q(\tilde{x}_{ij}, \vec{\psi})$ is made equivalent to the polar form of a displaced ellipse, as depicted in Figure 5.5. The radius of this polar form varies with the angle $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ between the robot heading $\vec{\psi}$ and the slope aspect $\vec{\gamma}_{ij}$. This is expressed in Equation (5.13), where $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ is also defined as the angle of the counter-clockwise rotation from $\vec{\gamma}_{ij}$ to $\vec{\psi}$, as Equation (5.14) indicates. When $\vec{\psi}$ coincides with $\vec{\gamma}_{ij}$ then $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ returns a value of zero. This is the direction in which the elevation decreases the most. For this reason, the direction of the slope aspect $\vec{\gamma}_{ij}$ is also referred to as the Descent direction. When $\vec{\psi}$ points towards the inverse direction, i.e. $\beta = \pm\pi$, this direction is known as the Ascent direction. Finally, the Lateral directions are those in which the value of $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ is either $\pi/2$ or $-\pi/2$.

$$\begin{aligned}
 0 = & [\cos\beta \sin\beta]^T \begin{bmatrix} p_1(\tilde{x}_{ij}) & p_2(\tilde{x}_{ij})/2 \\ p_2(\tilde{x}_{ij})/2 & p_3(\tilde{x}_{ij}) \end{bmatrix} \begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix} 1/Q(\tilde{x}_{ij}, \vec{\psi})^2 + \\
 & + \begin{bmatrix} p_4(\tilde{x}_{ij}) & p_5(\tilde{x}_{ij}) \end{bmatrix} \begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix} 1/Q(\tilde{x}_{ij}, \vec{\psi}) + p_6(\tilde{x}_{ij}) \quad (5.13)
 \end{aligned}$$

$$\beta(\vec{\psi}, \vec{\gamma}_{ij}) = \text{atan2}([0, 0, 1] \cdot (\vec{\gamma}_{ij} \wedge \vec{\psi}), \vec{\gamma}_{ij} \cdot \vec{\psi}) \quad (5.14)$$

Equation (5.13) uses six functions that are defined over the grid node \tilde{x}_{ij} : $p_1(\tilde{x}_{ij})$, $p_2(\tilde{x}_{ij})$, $p_3(\tilde{x}_{ij})$, $p_4(\tilde{x}_{ij})$, $p_5(\tilde{x}_{ij})$ and $p_6(\tilde{x}_{ij})$. In this way, these functions serve to build up the shape and the location of the displaced ellipse according to the information about the terrain that is placed at \tilde{x}_{ij} . In other words, terramechanic functions such as those presented in Chapter 3 determine the length of the ellipse axes together with its displacement, as they in turn affect the power consumption function introduced in (3.5). Nevertheless, these six functions are not intuitive at all, and for this reason the next step consists of substituting them. Here enters a novel way to model the anisotropic cost: the Continuous Anisotropic Model for Inclined Surfaces (CAMIS). It comprises three isotropic cost functions named the ascent cost $C^a(\tilde{x}_{ij})$, the lateral cost $C^l(\tilde{x}_{ij})$ and the descent cost $C^d(\tilde{x}_{ij})$. Each of them correspond to what $Q(\tilde{x}_{ij}, \vec{\psi})$ returns when the value of $\vec{\psi}$ is fixed at a certain direction relative to the slope aspect $\vec{\gamma}_{ij}$. In this way, $C^a(\tilde{x}_{ij})$ is defined in (5.15) as the cost function in the Ascent direction (see Figure 5.5), $C^l(\tilde{x}_{ij})$ in (5.16) as the cost function in any of the two Lateral directions and $C^d(\tilde{x}_{ij})$ in (5.17) as the cost function in the Descent direction.

$$C^a(\tilde{x}_{ij}) = Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \pm\pi) \quad (5.15)$$

$$C^l(\tilde{x}_{ij}) = Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \pm\frac{\pi}{2}) \quad (5.16)$$

$$C^d(\tilde{x}_{ij}) = Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = 0) \quad (5.17)$$

Going from the previous six ellipse functions to the new three CAMIS functions is done by using the substitutions presented in Equation (5.18). Another simplification is taken here to avoid complexity and is the main reason the number of functions is lower after the substitution. This simplification consists of setting $p_2(\alpha_{ij}) = 0$, which makes the axes of the ellipse parallel to the Descent - Ascent and Lateral directions respectively. Moreover, this ellipse is symmetrical in the axis parallel to the slope aspect $\vec{\gamma}_{ij}$. The cost returned by the anisotropic function is the same for both $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ and $-\beta(\vec{\psi}, \vec{\gamma}_{ij})$. In other words, the cost at both Lateral directions is the same, having hence $Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \frac{\pi}{2}) = Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = -\frac{\pi}{2})$. With this in mind, the \cos_β and \sin_β can be transformed into the expressions shown in (5.19).

$$\begin{aligned}
\begin{bmatrix} \frac{p_1(\tilde{x}_{ij})}{p_6(\tilde{x}_{ij})} \\ \frac{p_2(\tilde{x}_{ij})}{p_6(\tilde{x}_{ij})} \\ \frac{p_3(\tilde{x}_{ij})}{p_6(\tilde{x}_{ij})} \\ \frac{p_4(\tilde{x}_{ij})}{p_6(\tilde{x}_{ij})} \\ \frac{p_5(\tilde{x}_{ij})}{p_6(\tilde{x}_{ij})} \end{bmatrix} &= \begin{bmatrix} -Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \pm\pi) \ Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = 0) \\ 0 \\ -Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = -\frac{\pi}{2}) \ Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = +\frac{\pi}{2}) \\ Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \pm\pi) - Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = 0) \\ Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = -\frac{\pi}{2}) - Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = +\frac{\pi}{2}) \end{bmatrix} \quad (5.18) \\
&= \begin{bmatrix} -C_{ij}^a C_{ij}^d \\ 0 \\ C_{ij}^l{}^2 \\ C_{ij}^a - C_{ij}^d \\ 0 \end{bmatrix}
\end{aligned}$$

$$|\cos \beta| = \vec{\psi} \cdot \vec{\gamma}_{ij} \ , \ |\sin \beta| = \|\vec{\psi} \times \vec{\gamma}_{ij}\| \quad (5.19)$$

The anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$ is fully defined in (5.20). This expression results from getting the explicit form of the ellipse general form in (5.13) and applying the substitutions defined in (5.18) and (5.19). The CAMIS functions $C^a(\tilde{x}_{ij})$, $C^l(\tilde{x}_{ij})$ and $C^d(\tilde{x}_{ij})$ are defined in Equations (5.21), (5.22) and (5.23) respectively, considering the path planning criterion of electric charge minimization. In this way, the CAMIS functions are based on the use of the current consumption function $I(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ defined in Equation (3.7) presented in Chapter 3. As a side note, the instantaneous power function $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ from (3.5) could be used instead by simply multiplying the current consumption by the voltage V that is supplied to the motors.

$$Q(\tilde{x}_{ij}, \vec{\psi}) = \sqrt{\left(\frac{C_{ij}^a + C_{ij}^d}{2}\right)^2 (\vec{\psi} \cdot \vec{\gamma}_{ij})^2 + \left(C_{ij}^l \|\vec{\psi} \times \vec{\gamma}_{ij}\|\right)^2} - \frac{C_{ij}^a - C_{ij}^d}{2} \vec{\psi} \cdot \vec{\gamma}_{ij} \quad (5.20)$$

$$C^a(\tilde{x}_{ij}) = Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \pm\pi) = \frac{I(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})}{v_{ij}} \quad (5.21)$$

$$C^l(\tilde{x}_{ij}) = Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = \pm\frac{\pi}{2}) = \frac{I(L, \rho_{ij}, \sigma_{ij}, 0)}{v_{ij}} w_{ij}^\Phi \quad (5.22)$$

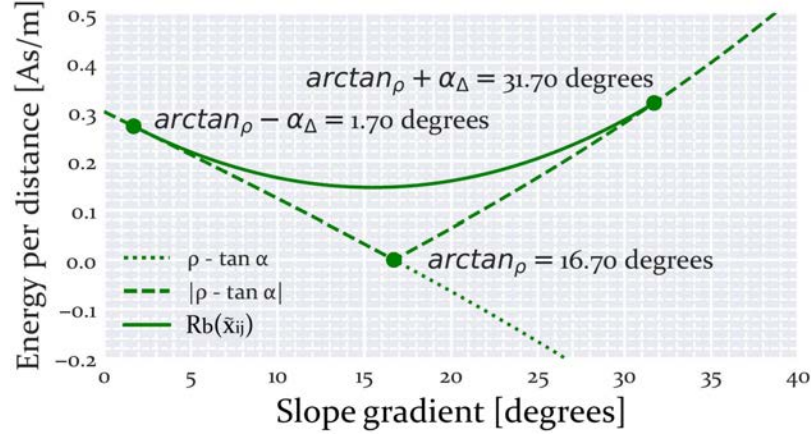


FIGURE 5.6: Use of Bezier curve function $R_b(\tilde{x}_{ij})$ to comply with the non-zero positive cost specification in a smooth way. $\rho_{ij} = 0.3$, $\alpha_{\Delta} = 15^\circ$ and $\alpha_{ij}^{zero} = \arctan_{\rho_{ij}}$.

$$C^d(\tilde{x}_{ij}) = Q(\tilde{x}_{ij}, \vec{\psi} \mid \beta(\vec{\psi}, \vec{\gamma}_{ij}) = 0) = \begin{cases} R_b(\tilde{x}_{ij}), & \alpha_{ij} \in (\alpha_{ij}^{zero} - \alpha_{\Delta}, \alpha_{ij}^{zero} + \alpha_{\Delta}) \\ \left| \frac{I(L, \rho_{ij}, \sigma_{ij}, -\alpha_{ij})}{v_{ij}} \right|, & \text{otherwise} \end{cases} \quad (5.23)$$

As can be denoted in the Lateral and Descent CAMIS functions, (5.22) and (5.23) respectively, there are some extra functions introduced. In the Lateral cost $C^l(\tilde{x}_{ij})$ there is a penalization function named w_{ij}^Φ . This function serves to minimize the value of the experienced roll angle when ascending or descending through slopes. In other words, it determines whether it is admissible for the robot to drive diagonally through the slope or not. The main justification to use this function is to avoid situations in which the robot may overturn because of losing stability in the lateral direction. Besides, it can make the robot avoid suffering from drift in such direction when traversing a slope.

A special treatment is given to the Descent CAMIS cost $C^d(\tilde{x}_{ij})$ in (5.23). The effect of gravity pulls the robot and reduces its energy consumption when it descends. Here, it is assumed the vehicle cannot recharge itself, so the energetic cost should always be present in the form of a positive value. Besides, it may be also desirable to prevent the robot from braking when descending through slopes, so the loss of energy in the form of heat is avoided (Rowe et al., 1990). When the robot descends, having a value of $-\alpha_{ij}$ as input that acknowledges this robot pose, the current function $I(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ and the instantaneous power function $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ could return zero or even negative values. This is incompatible with the OUM and bi-OUM requirements. Therefore, this situation is dealt with by using the Bezier function $R_b(\tilde{x}_{ij})$ that is present in (5.23). This function acknowledges the angle of slope gradient in which the robot would start gaining energy, α_{ij}^{zero} . From this slope gradient, the robot starts braking to keep its velocity, avoiding any acceleration. $R_b(\tilde{x}_{ij})$

preserves continuity and smoothness while making the Descent CAMIS cost $C^d(\tilde{x}_{ij})$ return always positive values that increase with the slope gradient α_{ij} . This is shown in Figure 5.6 for the case of the Normal-driving locomotion. Having in mind that Equations (3.18), (3.19) and (3.20) model the torque of the motors that actuate the wheels, and with the assumption of not having friction in the motor ($B_m = 0$) and the velocity of the robot is constant ($\dot{v}_{ij} = 0$), then the angle α_{ij}^{zero} would be equal to $\arctan \rho_{ij}$. The absolute value is used to define $C^d(\tilde{x}_{ij})$ in (5.23) so this function increases the cost with values of slope gradient higher than $\alpha_{ij}^{zero} + \alpha_\Delta$. The use of an absolute value was used by Rowe et al. (1990) to penalize paths that required the robot to brake and not accelerate. Here, α_Δ is a custom configurable angle margin. $R_b(\tilde{x}_{ij})$ marks three points using this margin: at $\alpha_{ij}^{zero} - \alpha_\Delta$, at α_{ij}^{zero} and at $\alpha_{ij}^{zero} + \alpha_\Delta$. A Bezier curve is the basis of $R_b(\tilde{x}_{ij})$ to not only preserve continuity but also smoothness while penalizing braking.

5.5 Summary and Conclusions

This chapter introduced the ins and outs of an anisotropic cost function for finding energy-optimal paths in scenarios with inclined surfaces. The path planning problem using this cost function is modelled after a PDE named the Hamilton-Jacobi-Bellman (HJB) equation. The resolution of this equation allows obtaining the optimal path between two points. To do this, the PDE Solving planner OUM, as well as its upgraded version bi-OUM, is considered. The use of this algorithm produces a viscous solution to the HJB equation, in the sense that it is a unique discretized approximation to the real solution without discontinuities. Nevertheless, to guarantee the uniqueness of the solution the anisotropic cost function is built in a way it complies with the convexity restriction demanded by the OUM. This compliance consists of modelling the inverse of this cost function as a closed conic function. Three CAMIS functions can be used to define the anisotropic cost function in four specific directions: Descent, Ascent and two Laterals. Moreover, these CAMIS functions are modelled after the locomotion models created in Chapter 3, so they can address terramechanic parameters. Beyond energy minimization, the Lateral CAMIS functions allow preserving the lateral stability of the robot by minimizing the angle of Roll along the trajectory. Besides, the Descent CAMIS function prevents the robot from braking on slopes when descending.

The novelty in the contents presented in this chapter rests on the way the novel cost function links the locomotion performance of a vehicle on slopes with a PDE Solving planner. The locomotion model introduced in Chapter 3 is here addressed again with the perspective of taking the direction of the robot into account. The presented cost function serves as a starting point that can be refined in the future as more terramechanic effects are considered (e.g. sinkage or lateral slippage). An important advantage is that the anisotropic cost function of any vehicle could be modelled using experimental data. For example, a vehicle could drive in a series of

tests where it moves in the Descent, Ascent and Lateral directions of many inclined surfaces. In this way, the CAMIS functions could be created and the compatibility with OUM be preserved, avoiding any concavities in the inverse of the cost function. In fact, this is done through some simulation and field experiments whose results are presented later in Chapter 6.

Chapter 6

Experiments

"Extraordinary claims require
extraordinary evidence."

Carl Sagan
1980

6.1 Introduction

The results from several tests support the contributions presented in previous chapters. These tests include simulation and field experiments. Table 6.1 presents all of them and relates them to the respective thesis contribution. Moreover, this table indicates the target platform for each experiment. This target can be either a simulation environment or a mobile testbed robot. The ins and outs of each experiment are introduced together with a discussion about the results.

The first of the experiments is a simulation comparative using isotropic cost functions presented in Table 3.1 included in Chapter 3. In particular, the chosen cost functions are those that minimize energy consumption. One of them is used in a scenario without obstacles, so it does not need to use the risk function r_{ij} to make paths get further from them. On the contrary, the second of them does take this risk function into account. Both functions are built upon the characteristics of a real prototype reconfigurable rover that is later presented. Two cases are compared twice, using a different scenario each time. These scenarios contain areas with different values of terramechanic parameters. The first of the two cases corresponds to a rover that only uses one locomotion mode. The second case corresponds to the same rover but using two locomotion modes instead. The second simulation comparative serves to analyze the performance of the Local Path Repairing (LPR) using the Sweeping approach. This process is executed in several cases where the rover must repair using the local layer one of the paths computed on the global layer in the previous comparative. Each repairing in this test emulates an encounter with a different arrangement of previously non-considered obstacles. The third experiment is carried out using another prototype rover named Heavy Duty Planetary Rover (HDPR).

TABLE 6.1: Summary of experiments presented in this thesis.

Thesis Contribution	Test Objective	Platform
Contribution 1: Optimal path planning considering multiple locomotion modes.	Reconfigurable Locomotion Cost Function	Simulation
Contribution 2: Multi-scale path planning combining initial long-traverse planning and dynamic local path replanning.	Local Path Repairing (Sweeping approach) Local Path Repairing (Conservative approach)	Simulation Real (HDPR)
Contribution 3: Creation of a direction-dependant cost function for planning the optimal traverse on slopes.	CAMIS - Energy Minimization CAMIS - Lateral Stability CAMIS - Detection of strengths and flaws	Simulation Simulation Real (Cuadriga)

This rover served to execute and validate the LPR employing the Conservative approach in a field test. The last three experiments, being two of them in simulation and the last one using a real platform, serve to validate the use of CAMIS functions to build an anisotropic cost function. The two simulation tests evaluate CAMIS on a certain criterion out of two: either the exclusive minimization of energy along the traverse or prioritizing the minimization of the experienced roll angle to preserve lateral stability. Finally, the last experiment consists of using a real mobile testbed named Cuádriga to evaluate landing CAMIS into reality, using for this a real terrain containing a slope.

This chapter is organized as follows. Section 6.2 details the first simulation test that validates the first contribution of this thesis. Thereafter, Section 6.3 presents the results and discussion of the next two tests: a simulation and a field experiment. Both of them validate the second contribution of this thesis by checking the two approaches of the LPR process. Later on, Section 6.4 presents the details about two simulation tests and one field test carried out to validate CAMIS. Finally, at the end of this chapter, Section 6.5 presents the conclusions extracted after analyzing the results.

6.2 Global Path Planning with Reconfigurable Rovers

As mentioned in the previous section, Chapter 3 presented an isotropic cost function that acknowledges robots capable to execute multiple locomotion modes, i.e., reconfigurable. This cost function is based on the dynamic models of many locomotion modes, selecting the best one given the terrain conditions. To do this, it uses the power consumption function $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ defined in Equation (3.5). In this way, the chosen criterion for the path planner is the minimization of energy. Equation (6.1) shows the expression of the cost function in question from Table (3.1).

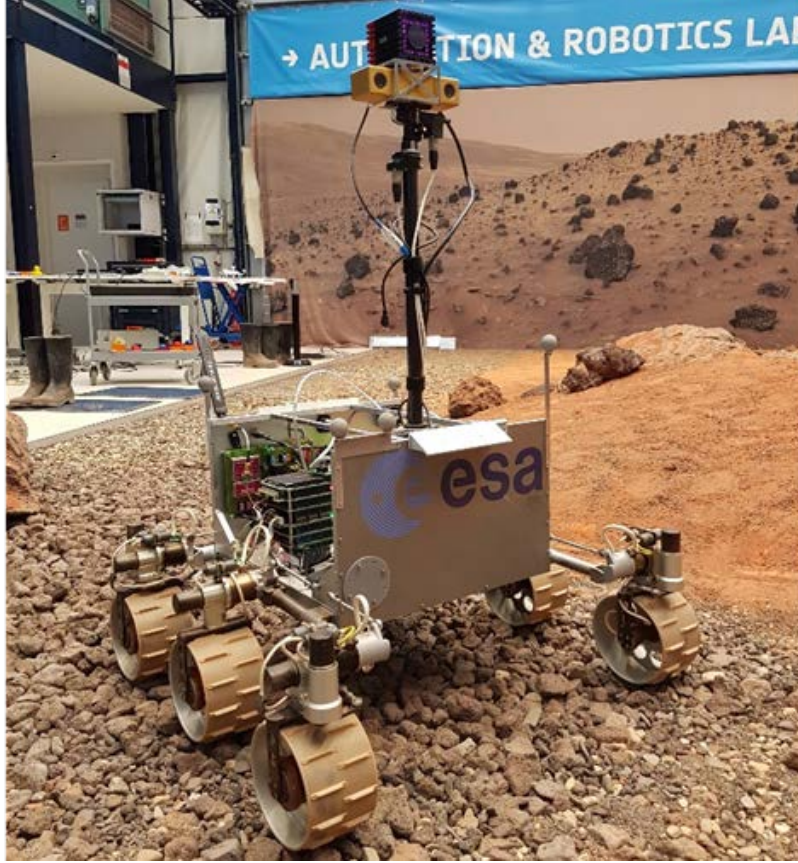


FIGURE 6.1: ExoMars Testing Rover (ExoTeR), a prototype that mimics the locomotion subsystem of ExoMars *Rosalind Franklin* rover.

$$C(\tilde{x}_{ij}) = \frac{P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})}{v_{ij}}, \quad L = \{nd, ww\} \quad (6.1)$$

The cost function in (6.1) serves to estimate the power consumption of a rover capable to perform Wheel-walking (see Equation (3.15)) together with Normal-driving (see Equation (3.21)). In other words, the set L is formed by locomotion modes nd and ww as expressed in (6.1). The power consumption function $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ is designed to comply with the requirements of an isotropic PDE planner such as FMM, capable to solve the eikonal equation in (3.22). Here, two simulation scenarios are presented to validate the use of this power consumption function. The modelling of the power consumption function and the path planner were carried out with the MATLAB-Simulink software. The main purpose of the simulation is to define, for each scenario, two distributions of different kinds of terrain and thereafter execute FMM to analyze how are the paths that are produced as a result. The execution of FMM is done twice, one considering just one locomotion mode (Normal-driving) and another considering two (Normal-driving and Wheel-walking).

Here, real numbers taken from the technical specifications of an experimental rover platform were considered to produce the instantaneous power consumption

TABLE 6.2: Simulation values of isotropic tests

Parameter	Value	Unit	Parameter	Value	Unit
B_m	$11 \cdot 10^{-5}$	Nms/rad	g	3.711	m/s^2
J_m	$4.36 \cdot 10^{-7}$	$kg\ m^2$	d_r	0.07	m
J_w	$12 \cdot 10^{-4}$	$kg\ m^2$	$K_w = K_b$	10.9/19	mNm/A
d_l	0.125	m	n	6	—
m_b	15	kg	m_w	0.484	kg

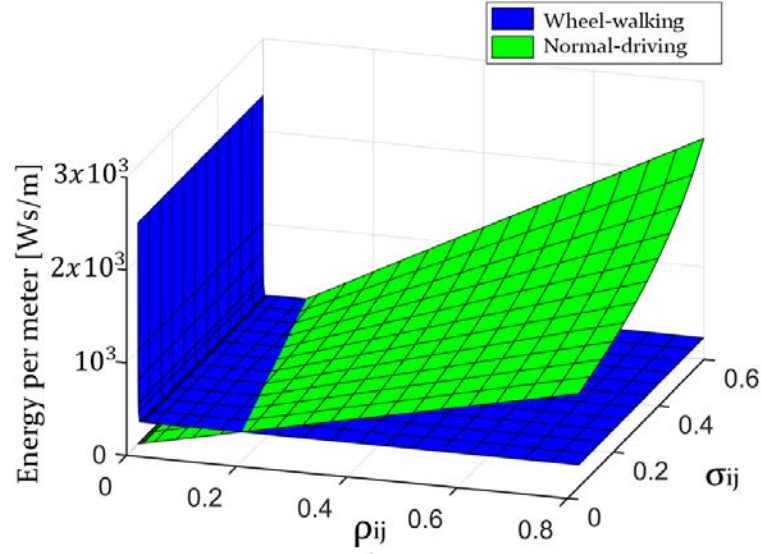


FIGURE 6.2: Estimation of the energy per meter cost consumed by the ExoTeR rover for each locomotion mode according to the values of slip ratio σ_{ij} and the specific resistance ρ_{ij} . This model does not address the effect of the slip ratio on the wheel-walking locomotion. Here it is considered the velocity v_{ij} as $0.02\ m\ s^{-1}$.

function $P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$ introduced in (3.5). As a reminder, this function is constructed upon the set L of available locomotion modes, the specific resistance ρ_{ij} , the slip ratio σ_{ij} and the slope gradient α_{ij} . The platform in question is the Exomars Testing Rover (ExoTeR) (Azkarate, Gerdes, et al., 2022). It is a reconfigurable rover capable to execute the two locomotion modes introduced in Chapter 3: the normal driving and the wheel walking. As mentioned earlier, the set L is defined as $\{nd, ww\}$. The corresponding numbers for this rover are shown in Table 6.2. The same kind of motors are used on top of the rover legs and on the wheels, and hence they share the same motor parameter values (in particular, the motor constants $K_w = K_b$). The values of the masses and moments of inertia are also included in the table. The gravity constant was set to $3.711\ m/s^2$ as it is the value that corresponds to the Martian surface. It is assumed the robot drives with a constant speed v_{ij} of $0.02\ m/s$, which is similar to the speed of real rovers.

Figure 6.2 shows the shape of the $P_{ww}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})$ and $P_{nd}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})$ models, estimated by simulation means using the *MATLAB - Simulink* software. There are some considerations taken. First, the effect of the slope gradient α_{ij} is here omitted, and

TABLE 6.3: Terrain parameters used in the first two simulation tests, where $P_{ij} = P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})$.

Terrain model	ρ_{ij}	σ_{ij}	α_{ij}	$P_{ij}[\text{W}]$	$v_{ij}[\text{m s}^{-1}]$	r_{ij}	$C(\tilde{x}_{ij})[\text{W} \cdot \text{s/m}]$	l
Rough	0.07	0.05	0.0	1.76	0.02	0.0	88.0	<i>nd</i>
Soft	0.45	0.5	0.0	4.72	0.02	0.0	236.0	<i>ww</i>
Near Obstacle	—	—	0.0	—	0.02	2148.0	2148.0	<i>nd</i>
Obstacle	—	—	—	∞	0.0	∞	∞	\emptyset

therefore it is set to zero. Both models were built according to the values of ρ_{ij} and σ_{ij} comprehended between 0 and 1. The *Side-by-side* gait of Wheel-walking (see Figure 3.3 in Chapter 3) was implemented with maximum and minimum limits for both θ_{b1} and θ_{b2} angles as 15 and -15 degrees. These are the same values that were used in previous experiments done in the past with ExoTeR (Azkarate, Zwick, et al., 2015). A second consideration that was made while creating the Wheel-walking mode is that this locomotion mode was only affected by ρ_{ij} , being independent of σ_{ij} . It was assumed that the wheel that remains static while the rover makes a stride does not slide. The same assumption went for the other wheel, as it is dragged by the motion of the robot body. More elaboration on this has been done, considering also slip that may happen, but the model here presented does not consider that. It is worth mentioning that the main focus of this thesis is the path planning application rather than deepening into the modelling of the terramechanics.

As can be checked in Figure 6.2 and in Table 6.3, $P_{ww}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})$ returns an energy consumption per meter of $236 \text{ W} \cdot \text{s/m}$. This remains more or less constant for all values of ρ_{ij} and σ_{ij} . On the contrary, the function of Normal-driving has the shape of an inclined surface. It intersects with the function that models Wheel-walking. In this way, for values of ρ_{ij} less than 0.15-0.2 (depending on the value of σ_{ij}), Normal-driving seems the cheapest locomotion, while for the rest of the values the better option is the Wheel-walking mode. The cost of Normal-driving increases with the slip ratio σ_{ij} , which makes sense in soft terrains such as sand. For very low values of ρ_{ij} , the cost of Wheel-walking escalates immensely. This is because if no resistance exists this mode cannot keep the wheel locked in the surface and slides, like on top of an icy surface. Moreover, in case the motors reach the maximum power they can handle in both locomotion modes, the corresponding node should be considered as an obstacle. In this situation, especially when the cost of Normal-driving increases, the rover could get stuck or perform much worse. As a reminder, these models are preliminary and rest on estimations of these parameters rather than being refined and more detailed functions. The basis (and focus) of the first contribution in this thesis is to link this kind of model with isotropic path planning.

Figure 6.3 shows the scenario with the location of the origin \tilde{x}_o and goal \tilde{x}_g positions, as well as the resulting paths Γ_A and Γ_B . With regards to the types of terrain used in this scenario, they are coloured in Figure 6.3c: the lighter of them is labelled Soft Terrain while the darker one is Rough Terrain. The simulation tests that were

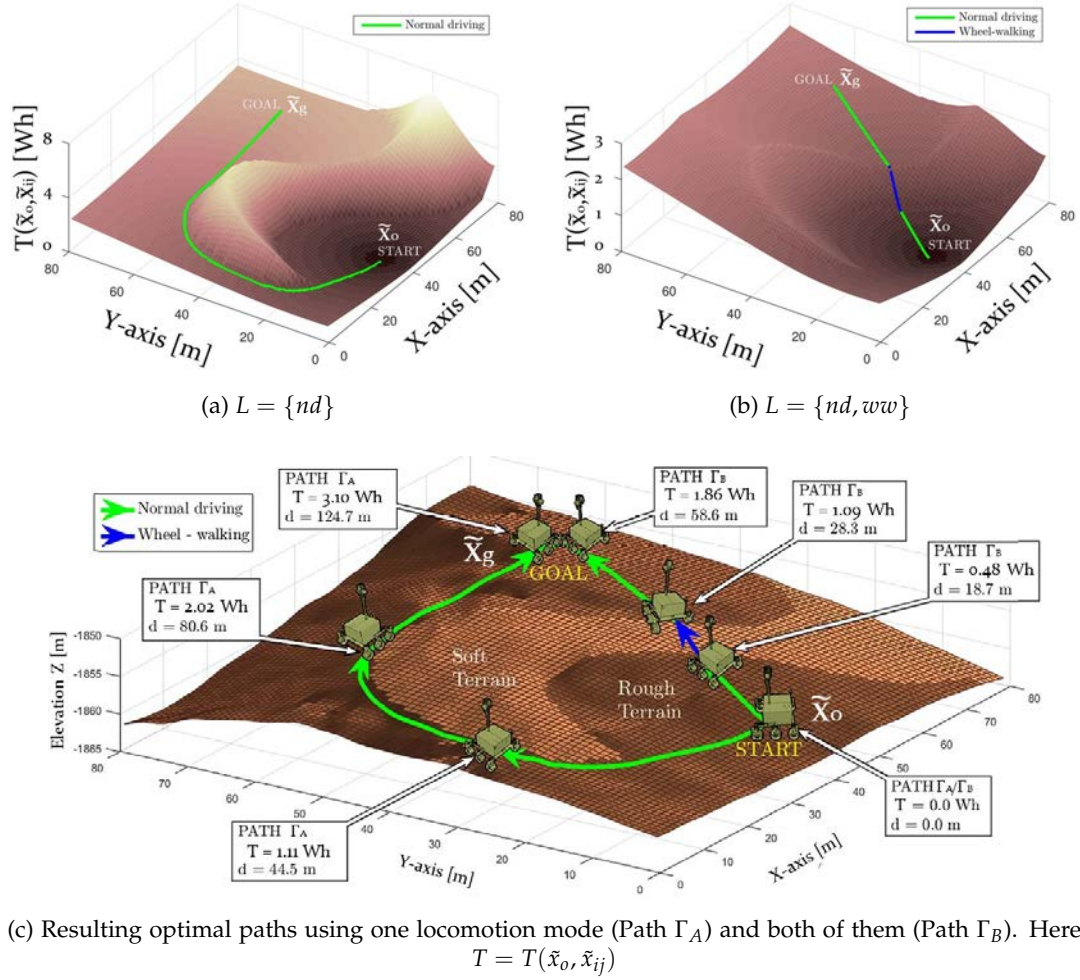


FIGURE 6.3: Calculated values of total cost $T(\tilde{x}_o, \tilde{x}_{ij})$ and obtained paths when using only Normal-driving or $L = \{nd\}$ (a) and both Normal-driving and Wheel-walking or $L = \{nd, ww\}$ (b).

performed in this first scenario emphasize the fact that the use of an alternate locomotion mode can make a rover save up energy and drive lower distances. The custom distribution of the areas covered by each type of terrain (with one of them resembling a hook) is made to highlight an extreme case supporting this statement. The terramechanic parameters assigned to each terrain are shown in Table 6.3.

There are two distinct cases as mentioned before: one in which a robot with only Normal-driving is considered and another that also is capable to execute Wheel-walking. Path Γ_A is the one resulting from the first case, while path Γ_B does the same but with respect to the second case. The total cost $T(\tilde{x}_o, \tilde{x}_{ij})$ is represented as a potential function in Figures 6.3a and 6.3b. The first of them shows the total cost for the first case with path Γ_A , while the second does this with path Γ_B . As can be seen in both cases, the only minimum point is global and is located at the point where the wave originates, \tilde{x}_o in this case. Therefore, here the total cost here indicates the amount of energy required to arrive at each node from the initial position. The planner estimated that Path Γ_A takes up to 3.1 Wh of energy and entails a distance of

124.7 m, while Path Γ_B requires 1.86 Wh and 58.6 m. In other words, given the constant speed command v_{ij} with a value of 0.02 m s^{-1} , path Γ_A takes 6235 s (equivalent to 1 h 43 min 55 s) and path Γ_B takes 2930 s (equivalent to 48 min 50 s). By using the second locomotion mode the rover could save in this scenario, given the initial and goal positions, 40.17 % of energy and 53.0 % of time. This is the same conclusion extracted from a similar example given at the end of Chapter 3, where there are as well two kinds of terrain and a reconfigurable rover. In that chapter, Figures 3.6 and 3.9 detail such example but without making an insight into the numerical results.

The next scenario is similar but is based on a different distribution of terrain areas, uses more origin points and contains obstacles. A 3d representation of this terrain can be seen in Figure 6.4a. Being close to the facilities of ESA-ESTEC¹, this terrain resembles a Martian landscape, including craters. Through drone imagery and photogrammetry techniques, an elevation map is constructed. The simulation tests using this terrain were implemented using *MATLAB* and the code can be found in an online repository²³. The planning done here serves as the Global Path Planning (GPP) for the DyMu algorithm, producing long traverses as a result. One of them is later used to test the Local Path Repairing (LPR). For this reason, the total cost in this case was calculated with respect to the goal, i.e. $T(\tilde{x}_{ij}, \tilde{x}_g)$. The resolution Λ of the Digital Elevation Map (DEM) shown in Figure 6.4b is 1 meter. To determine which nodes are obstacles, the slope gradient α_{ij} is calculated using (3.4) and the elevation Z_{ij} shown in Figure 6.4b. All those nodes with a value of slope gradient higher than a threshold of 15° are considered obstacles.

Similarly to the previous simulation test, here two custom different terrain areas were defined: Rough and Soft. They share the same properties as the terrains used in the previous simulation and detailed in Table 6.3. Figure 6.5a shows how these terrains are distributed. Around those nodes labelled as obstacles, there is a third type of terrain. It has a higher cost than the other two types as it uses the risk function r_{ij} , which serves to make paths go further from obstacles. This risk function is integrated into the cost function as expressed in (6.2). The value it returns is indicated in Table 6.3. The risk function r_{ij} serves to compensate for the discontinuity between the traversable and the non-traversable costs. This is useful to avoid doing the gradient descent method near obstacle nodes, where the calculated values of total cost degenerate because of the discontinuity.

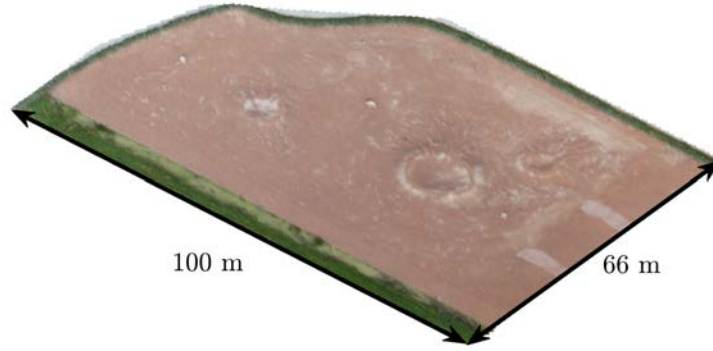
$$C(\tilde{x}_{ij}) = \frac{P(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})}{v_{ij}} + r_{ij}, \quad L = \{nd, ww\} \quad (6.2)$$

As images like those provided by THEMIS are not as precise as the ones from HiRISE (around 100 meters per pixel instead of 1) the custom distribution is not too much detailed. Similarly to the simulation carried out using the previous scenario,

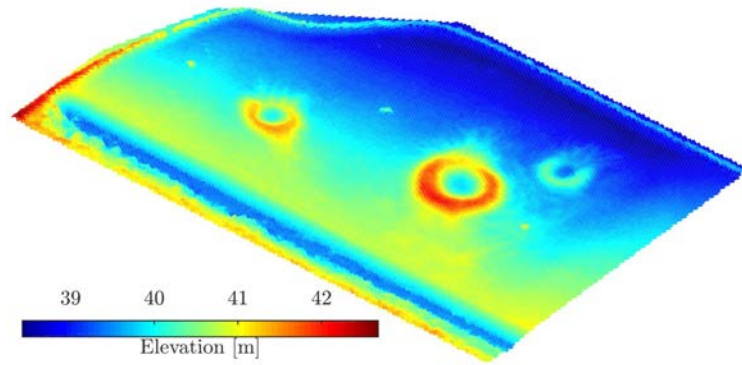
¹DMS Coordinates: 52°12'55.0"N 4°25'39.1"E

²https://github.com/spaceuma/ARES-DyMu_matlab

³https://github.com/ESA-PRL/planning-path_planning



(a) 3d model using an orthonormal image as texture.



(b) Digital Elevation Map.

FIGURE 6.4: Overview of virtual models describing the shape of the experimental terrain located near ESA facilities.

two cases were considered: one with only Normal-driving ($L = \{nd\}$) and another with both Normal-driving and Wheel-walking ($L = \{nd, ww\}$). The total cost gained by having Wheel-walking against not having it is represented in Figure 6.5b as a percentage with respect to having only one locomotion mode. In other words, this figure depicts the reduction in the total cost calculated for the reconfigurable rover in comparison with the total cost for the common rover that only uses Normal-driving. It was expected to obtain this gain as the parameters chosen for the Soft terrain make the power consumption function of Wheel-walking, $P_{ww}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})$, return a lower value than that of the Normal-driving $P_{nd}(\rho_{ij}, \sigma_{ij}, \alpha_{ij})$. Besides, this reduction in the total cost also affects certain places of the Rough terrain. This is because of how the Soft terrain is placed between the initial and goal positions. The location of these positions and the resulting paths can be found in Figures 6.5c and 6.5d. Figure 6.5c shows the paths generated from the planner that only considers Normal-driving. As can be seen, most of the resulting paths must circumvent the Soft terrain as it is very costly for the robot. This means the robot would need to drive longer distances than in the other case with Wheel-walking. For the case of the reconfigurable robot, some of the paths, those starting from the top side, traverse the Soft terrain. This is because Wheel-walking reduces the associated cost to this kind of terrain. As a result, these paths take less total cost and save up energy compared to the paths generated for a

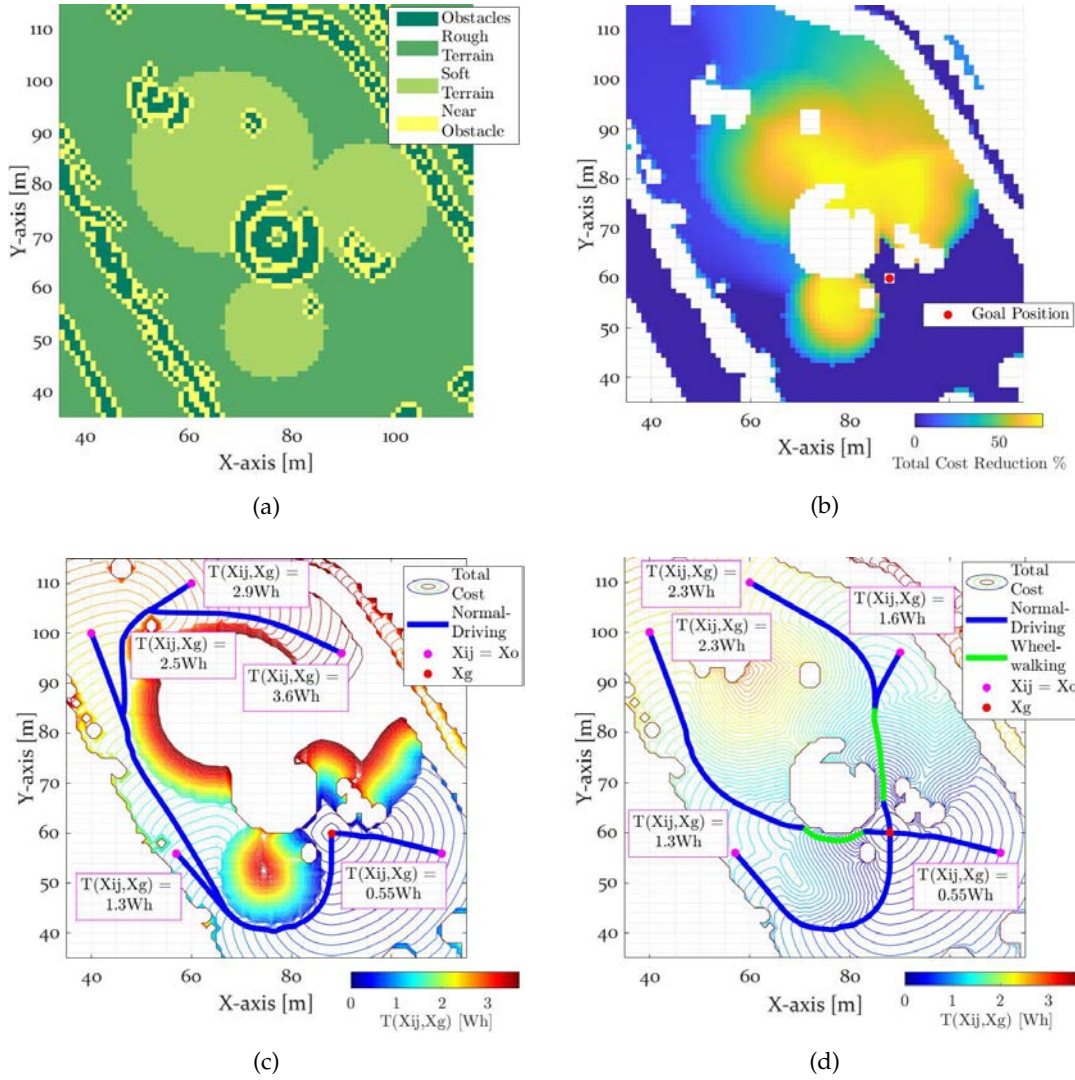


FIGURE 6.5: The distribution of the different terrains is shown in (a). The FMM is executed twice: one considering just the Normal-driving mode (c), $L = \{nd\}$, and other also taking the Wheel-walking mode into account (d), $L = \{nd, ww\}$. In this way, the total cost to arrive at the destination is reduced for some areas (b).

robot only capable to execute Normal-driving. This experiment demonstrates how the proper modelling of the cost function to address multiple locomotion modes allows finding better paths in scenarios with multiple kinds of terrain.

6.3 Local Path Repairing Tests

This section presents the details about the tests carried out to check the functioning of the Local Path Repairing (LPR), introduced in Chapter 4. In particular, the results from a simulation and field experiments are introduced. The simulation test was carried out to analyze the Sweeping approach by executing the LPR in one of the paths calculated in the previous simulation test. The field test consisted of a rover autonomously navigating through a flat surface containing rocks and craters. This

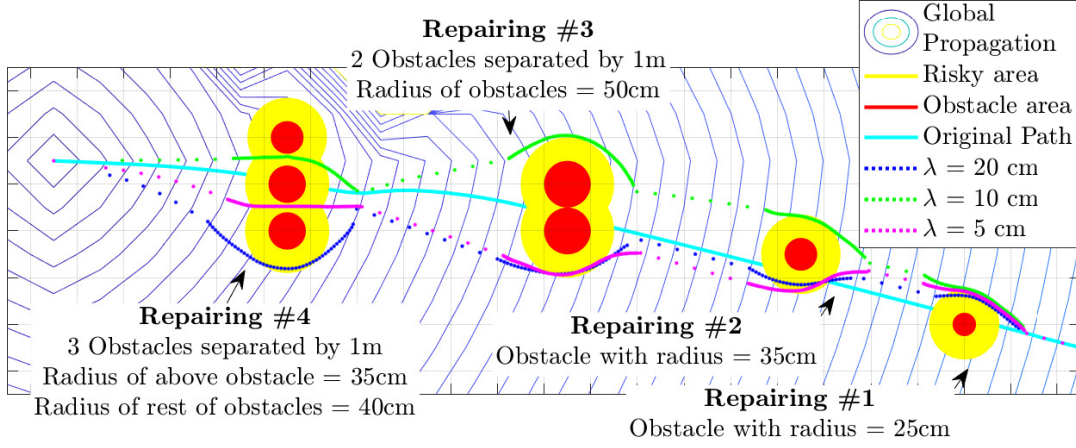


FIGURE 6.6: Resulting Paths from the LPR simulation test using different values of λ .

robot followed a series of pre-planned paths, dynamically updating them using the Conservative approach of the LPR.

Sweeping Approach Simulation Test

The simulation test started with one of the paths computed in the previous section. It is the path that can be found in Figure 6.5d in the bottom right. Some virtual obstacles were placed with different sizes and arrangements along the way. Figure 6.6 depicts them along the path. They have a circular shape with values of radius from 0.25 m to 0.5 m. For computing the risk around the obstacles, the distance d_{risk} was set to 0.5 m. This distance complies with Equation (4.5), given that to extract the path with the gradient descent method a step distance d_{step} of 0.4 m was used. Besides, three different values of resolution λ were chosen to study its influence on the shape of the resulting paths. These values are 0.05 m, 0.1 m and 0.2 m. Having into account the value of resolution Λ of the global layer is 1 m, the values of local resolution λ make each global node \tilde{x}_{ij} subdivide into 400, 100 and 25 local nodes respectively. Figure 6.6 shows the resulting paths after executing LPR and according to the chosen value of λ . Those paths produced using a value of λ of 0.05 m are slightly smoother than the others, finding even a shorter path in the fourth repairing as small gaps between obstacles are addressed. A main drawback of using a finer resolution is the increase in the computation processing that it entails. A numerical analysis is provided in Figure 6.7 to support this statement. Figure 6.7a indicates for each case the number of local nodes that are processed to perform each repairing operation. As can be checked, the finer the resolution the higher this number of processed local nodes. Besides, it is included in this figure an average of the processed local nodes during the executions of the Risk Expansion process. Figure 6.7b presents a plot indicating how many times per case the eikonal equation has been solved. As in the previous plot, this number increases with the resolution of the Local Layer. This number of times considers both the eikonal for the Risk Expansion defined in

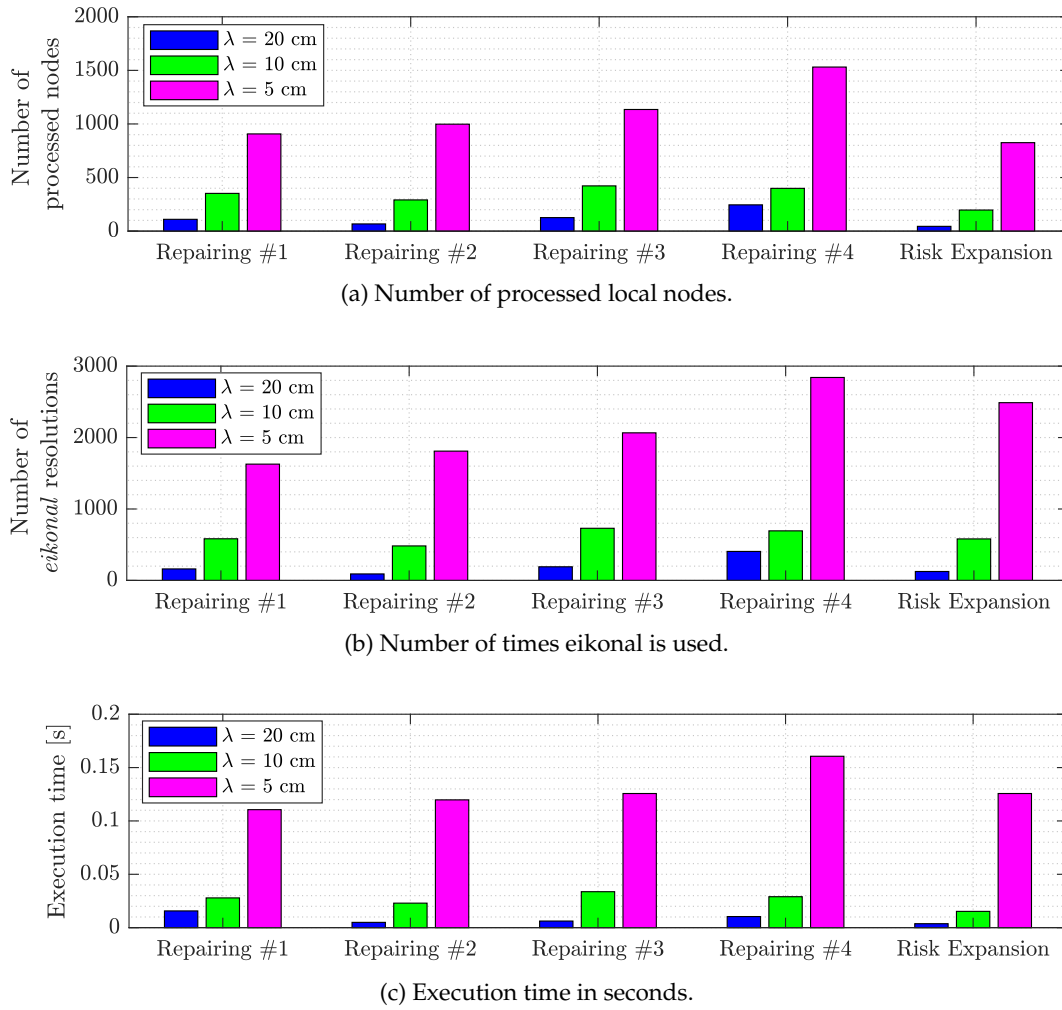


FIGURE 6.7: Results from the LPR simulation test. The shape of the repaired paths after executing the LPR process 4 times is shown (a), as well as information relative to the computational power used, in the form of the number of processed nodes (b), the number of times the eikonal equation is used in total and the elapsed time to do the computation.

Equation (4.7) and the eikonal for the repairing itself in Equation (4.11). Figure 6.7c indicates the execution time taken by the planner to calculate the solution for each case and given the value of λ . The computer that was used to perform this test had an Intel Core i7-7500 as CPU and 12 Gb of RAM memory. The version of MATLAB used was R2017b. Table 6.4 indicates the same results for the GPP to compare the two scales of planning. As can be denoted, the time to compute the GPP is an order of magnitude higher than the LPR in general terms. For this reason, the GPP is meant to be executed just once, at the beginning of the traverse, and even in an offline way. The LPR can be executed more times and in real-time.

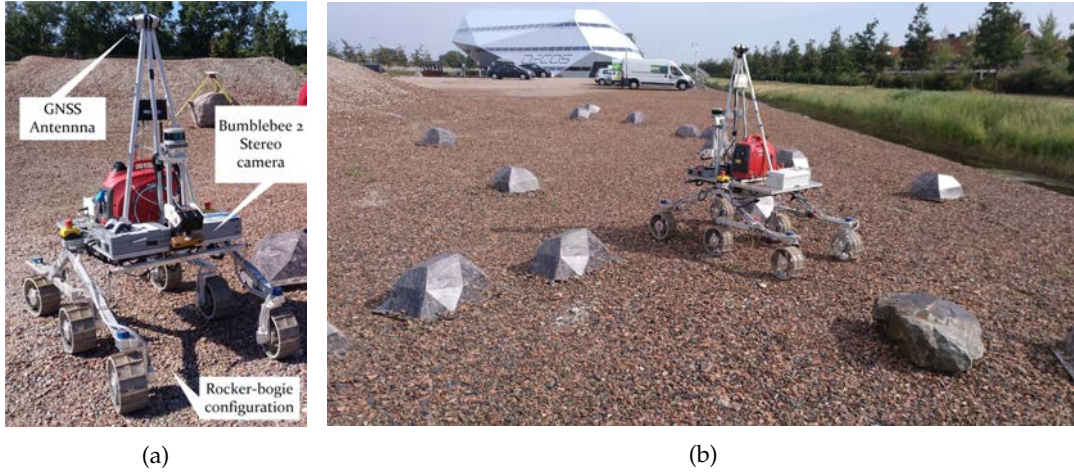


FIGURE 6.8: The Heavy Duty Planetary Rover (HDPR), shown in (a), performing its traverse through the experimental field populated by obstacles shown in (b).

Conservative Approach Field Test

The other approach of LPR, the Conservative approach, was tested using a rover prototype, the Heavy Duty Planetary Rover (HDPR). Figure 6.8 shows images portraying this rover. It is a mobile robot with six active wheels, being four of them steerable (the two at the front and the two at the rear). This rover is capable to perform Full-Ackermann manoeuvres, including the rotation at a fixed position (Point Turn). Besides, its kinematic configuration is based on a rocker-bogie structure with passive joints. A more detailed and insightful description about the characteristics of this robot can be found in the work of Hewitt et al. (2018).

The field test in question was performed on the real terrain from which the 3d model shown in Figure 6.4a is constructed. Figure 6.9a portrays an orthonormal image of the setup. This test emulated an operation to command the rover to follow a path previously defined, e.g. a path from the GPP using the DyMu architecture. The rover found on its way obstacles that had to be autonomously avoided. Figures 6.8b and 6.9a depict some of them. These obstacles were placed randomly over the terrain and it was assumed that when the initial path was created such obstacles were not taken into account. In fact, Figure 6.9b shows how the global layer from the previous tests was used, with a resolution of 1 meter. Instead of opting for an energy-minimization approach, in this particular case, the global path planning operation was executed based on distance minimization. This criterion is equivalent to time minimization as the rover drove at a constant speed v_{ij} of 0.1 m s^{-1} . Besides, the reduction of the proximity to obstacles was also included as an extra criterion. According to Table 3.1 in Chapter 3, the corresponding cost function is the one presented in (6.3).

$$C(\tilde{x}_{ij}) = \frac{1}{v_{ij}} + r_{ij} \quad (6.3)$$

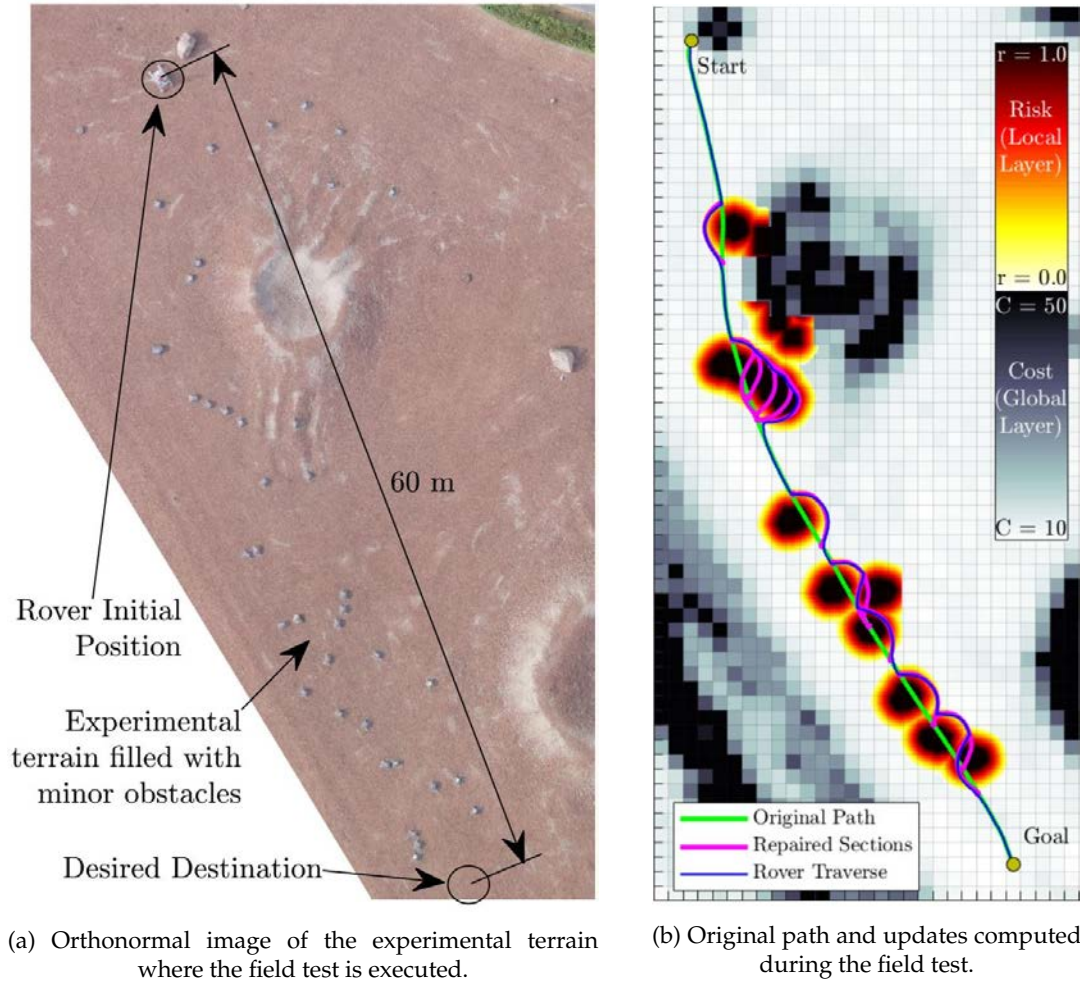


FIGURE 6.9: Paths computed during the field test.

In this case, r_{ij} was defined as a penalization to prevent the rover from getting close to inclined terrains. As a result, this function took the form expressed in Equation (6.4). Here, the penalization grows linearly by intervals of 5 degrees of slope gradient α_{ij} (the maximum inclination of the terrain). Values of α_{ij} greater than 15 degrees are greatly penalized with a risk value of 120, which is a high value compared to the cost $C(\tilde{x}_{ij})$ at zero degrees of slope gradient. This cost on a flat surface, according to (6.3), is $1/v_{ij}$. Given the speed of the rover is 0.1 m s^{-1} , the cost on a flat surface takes the value of 10 s m^{-1} .

$$r_{ij} = \begin{cases} \alpha_{ij}, & \alpha_{ij} \in [0, 5] \text{ degrees} \\ 5 + 2(\alpha_{ij} - 5), & \alpha_{ij} \in [5, 10] \text{ degrees} \\ 15 + 3(\alpha_{ij} - 10), & \alpha_{ij} \in [10, 15] \text{ degrees} \\ 120, & \alpha_{ij} > 15 \text{ degrees} \end{cases} \quad (6.4)$$

The GPP produced the optimal path given the cost function in (6.3). This optimal path is depicted using the green colour in Figure 6.9b. The rover HDPR followed the

TABLE 6.4: Results from the execution of the GPP to produce the path that is later repaired in Figure 6.9b.

Parameter	Value
Visited <i>Global Nodes</i>	10522
Calculations of eikonal Equation	19929
Execution Time	0.2 s

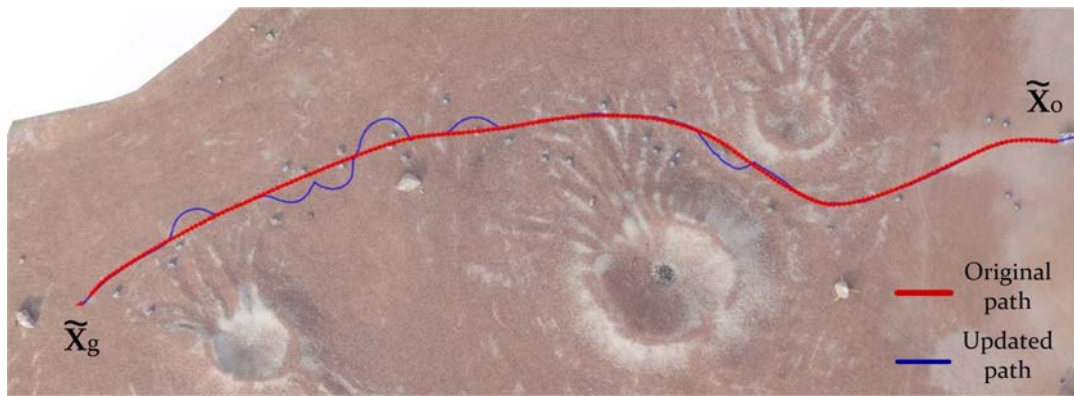
TABLE 6.5: Results from the execution of the GPP to produce the path that is later repaired.

Traverse	Initial Path Distance [m]	Updated Path Distance [m]
Figure 6.9	62.6	79.2 (+26.52 %)
Figure 6.10a	86.0	97.7 (+13.6 %)
Figure 6.10b	45.5	58.9 (+29.45 %)
Figure 6.10c	28.9	38.8 (+34.26 %)
Figure 6.10d	58.4	70.1 (+20.03 %)
Figure 6.10e	171.0	191.0 (+10.47 %)

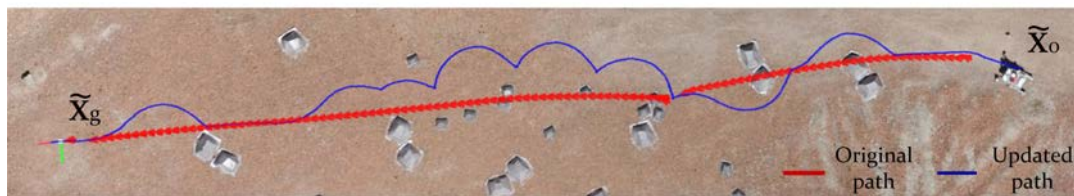
path using a path tracking component based on an algorithm called *Conservative-Pursuit* or *C-Pursuit* (Gerdes et al., 2020). This algorithm defines boundaries around the path and prevents the rover from trespassing them, limiting in this way the admissible tracking error. Moreover, the rover was equipped with an on-board perception system based on a frontal camera. This system was capable to evaluate the elevation of the terrain in the immediate few meters ahead of the rover (see Figure 6.8a). This system can detect any abrupt change in elevation and consider it as an obstacle (Gerdes et al., 2020). Any obstacle detected in this manner was later mapped onto the local layer whenever the LPR was triggered. This process generated local waypoints each time it was called, re-connecting the position of the rover at that moment with the original path. In Figure 6.9b the local sections made up by these local waypoints are coloured in pink. The value of resolution λ chosen for these LPR operations is 0.1 m. During this test there were 13 repairing operations executed in real time. A recording of this experiment with HDPR driving and avoiding obstacles is available online⁴.

In addition to that traverse, a few more were carried out, shown in Figure 6.10. The scenario was the same but with different origin and goal positions and obstacle arrangement. The computation times taken by the planner to perform each path update are presented in Figure 6.11. The specifications of the onboard computer where the DyMu planner was installed are the following: an Intel i7-6600U with 16 Gb RAM and Ubuntu 18.04 as the operating system. As can be denoted in that Figure, the repairing operations were performed in the order of milliseconds, taking the slower one less than 200 milliseconds. With this fact in mind, and considering the

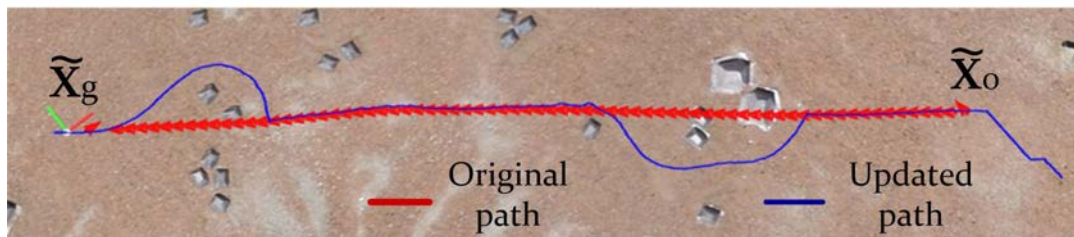
⁴<https://youtu.be/X4mihNTEVGw>



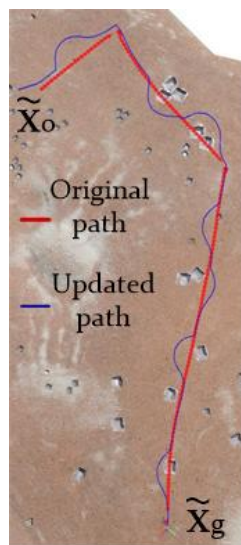
(a)



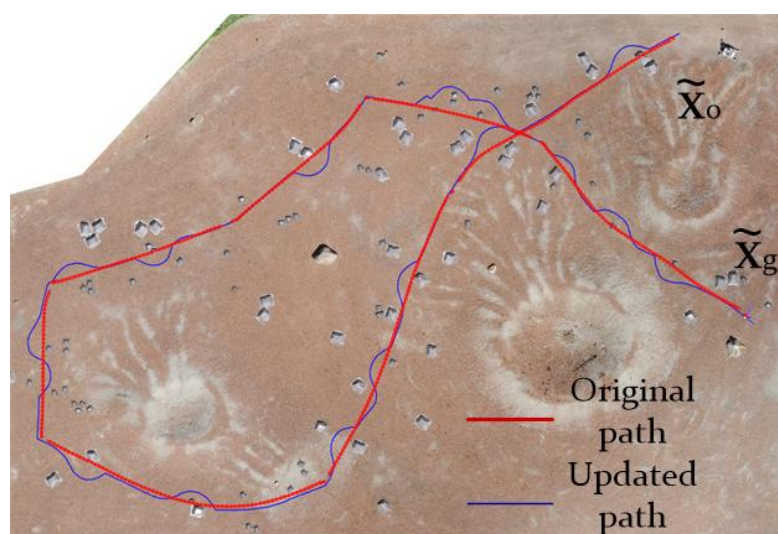
(b)



(c)

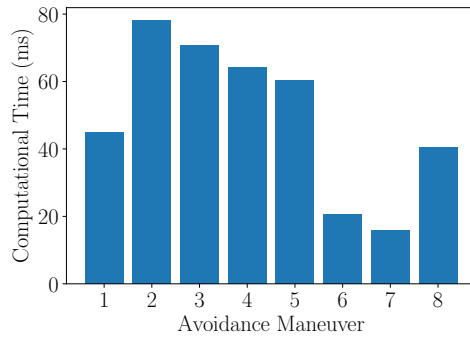


(d)

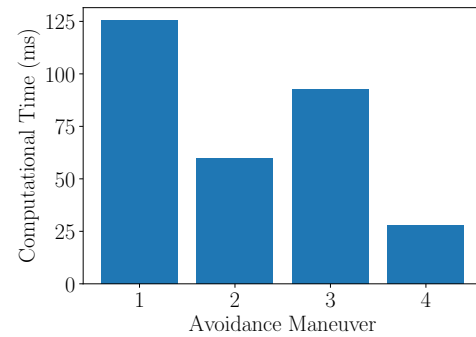


(e)

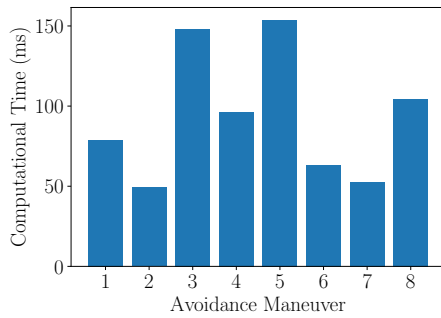
FIGURE 6.10: More traverses performed by HDPR during field tests.



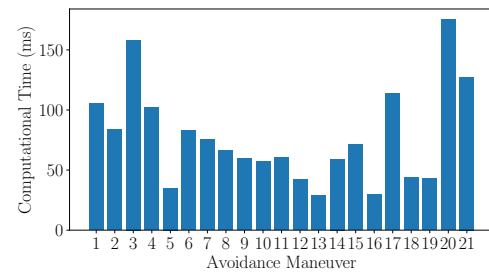
(a) Short traverse 1.



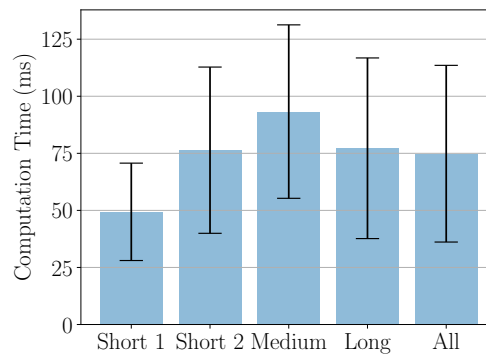
(b) Short traverse 2.



(c) Medium traverse.



(d) Long traverse.



(e) Mean and standard deviation of all traverses.

FIGURE 6.11: Showcase of the computation times that were taken by the HDPR onboard computer to execute the Local Path Repairing. Figures (a), (b), (c), and (d) depict the values for all path updates along the respective path. In (e), the computed mean and standard deviation of each traverse is shown, together with the mean and standard deviation of the computation times for all traverses.

low speed of the rover, it can be concluded that the Local Path Repairing (LPR) can be executed online, given the mentioned computer specifications. Table 6.5 details the initial and updated distances of the planned path. This information gives an idea of how the path length increases with the arrangement of obstacles presented in Figure 6.10. For example, in the traverse shown in Figure 6.10c the density of obstacles is higher, and hence the local waypoints take more distance to reconnect with the original path. From these results the main conclusion extracted is that the Conservative approach has the advantage of not requiring the global layer at all. However, the repairing operations prioritize returning to the original path instead of calculating new global waypoints as in the Sweeping approach, entailing more distance to be driven by the rover.

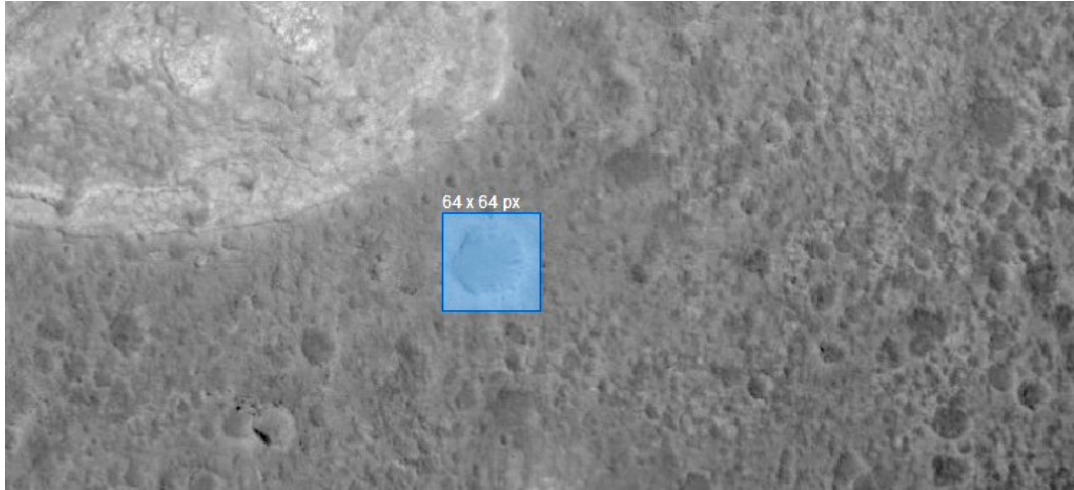
6.4 Anisotropic Path Planning for Traversing Slopes

The purpose of the experiments that are introduced next is to check the use of the anisotropic cost function $Q(\tilde{x}_{ij}, \vec{\psi})$, introduced in Chapter 5. The CAMIS cost functions $C^a(\tilde{x}_{ij})$, $C^l(\tilde{x}_{ij})$ and $C^d(\tilde{x}_{ij})$, previously introduced in (5.21), (5.22) and (5.23), are used to model this anisotropic cost together with the dynamic model of locomotion introduced in Chapter 2. In particular, here only the Normal-driving locomotion is addressed to exclusively focus on analyzing the usage of the anisotropic cost function on scenarios with slopes, rather than focusing on the reconfiguration capability. The results of two numerical simulations and one field experiment are here presented. All the code used to produce the paths of all the tests was written using the Python language and is available online⁵.

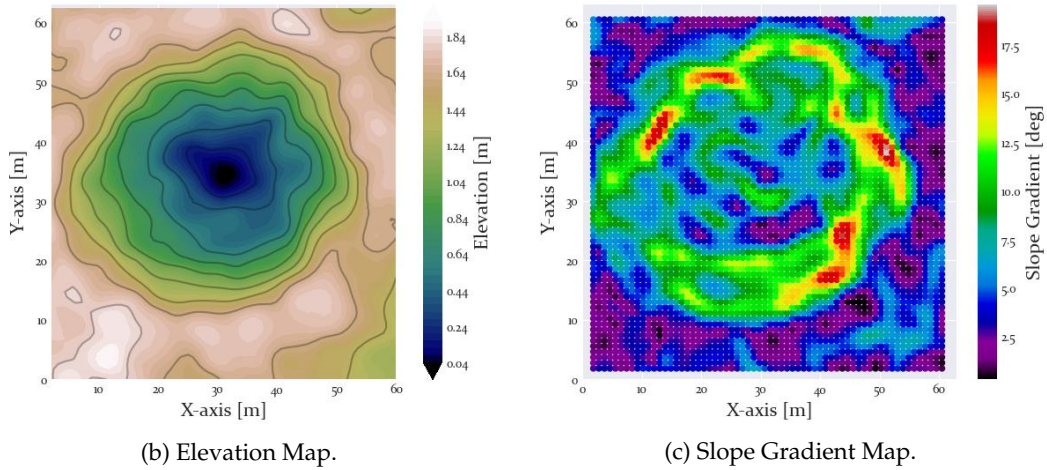
The bi-OUM algorithm was executed in the first simulation to produce a series of paths in a scenario containing a crater. The purpose of this experiment is twofold. First, it is of interest to analyze how the terramechanic functions introduced back in Chapter 3 influence the performance of the anisotropic cost function to find energy-minimizing paths. As a reminder, these functions are the specific resistance ρ_{ij} and the slip ratio σ_{ij} . Moreover, the slope gradient α_{ij} and the aspect direction $\vec{\gamma}_{ij}$ will affect the existing anisotropy. The second objective of this simulation experiment is to figure out how significant is the use of an anisotropic function in contrast with an isotropic function, usable by the FMM and with much lower computational complexity.

The second simulation test and the field experiment focus on the use of the roll weight function w_{ij}^Φ to penalize this orientation angle and minimize the lateral inclination of the robot. This weight function is part of the CAMIS lateral cost $C^l(\tilde{x}_{ij})$. By increasing what w_{ij}^Φ returns the roll minimization takes more significance. The simulation aims to plan a series of paths given different expressions of this weight function. Afterwards, the field experiment was carried out with the four-wheeled

⁵https://github.com/spaceuma/CAMIS_python



(a) Selection of region from the DEM of a crater. The elevation used is half of the original.



(b) Elevation Map.

(c) Slope Gradient Map.

FIGURE 6.12: Preparation of the map used for the simulation tests with the anisotropic planner. It is based on the shape of a real crater on the Martian surface. The resolution of the DEM is 1 m.

robot named Cuádriga. This is the robot whose technical specifications were considered to construct the cost functions in the simulations. Cuádriga, together with the equipment used in the field test, is introduced and described in this section. This robot tracked four of the resulting paths from the second simulation experiment as a first attempt to land the use of the CAMIS functions into reality. In this way, this field test served to clarify what were the main issues found and what is left for future improvement. This section presents at the end the comparison between the expected behaviour according to the plans produced using the CAMIS functions and the recorded data describing what the robot experienced during the tests.

Energy Optimization Simulation Tests

The first simulation test with the anisotropic cost function was carried out using the Digital Elevation Map (DEM) of a crater. This DEM is based on the shape of a real crater located close to where the Spirit rover landed in Mars. The data was obtained

from the High Resolution Imaging Science Experiment (HiRISE)⁶ repository and was adapted to make it present slopes up to 20 degrees. The main reason to do this is that the original presents too pronounced slopes that would be non-traversable, and the key point in this simulation test is to have a large variety of traversable inclined surfaces. Figure 6.12a shows the 80×80 m portion of elevation data that was extracted from the original DEM. Figure 6.12b presents the elevation data describing the shape of this crater together with some contour lines to ease the visualization. The slope gradient that is calculated from this elevation, using Equation 3.4 for this, is shown in Figure 6.12c for all the points in the map.

First of all, it is analyzed how the specific resistance ρ_{ij} and the slip ratio σ_{ij} influence the energy consumption estimation provided by the CAMIS cost function. It is worth mentioning that, in a similar way to isotropic cost functions used by FMM, the constant parameters that multiply the whole function, acting as gains, do not affect the location of the waypoints of the resulting path. These parameters only modify proportionally the values of total cost assigned to the nodes. Moreover, here the roll weight function w_{ij}^Φ is not taken into account, setting it to return a constant value of 1. This is done to decouple it from the rest of the parameters. The effects of w_{ij}^Φ will be addressed in later tests. Since the value returned by the specific resistance ρ_{ij} can be a real number higher than 0 and lower than 1, the simulation test is performed using a discrete set of constant values of ρ_{ij} : 0.15, 0.3, 0.45, 0.6, 0.75 and 0.9. In other words, the planner several times, using one out of the six different values of ρ_{ij} each time for all nodes $\tilde{x}_{ij} \in \tilde{\Omega}$.

$$C^a(\tilde{x}_{ij}) = \frac{I(L, \rho_{ij}, \sigma_{ij}, \alpha_{ij})}{v_{ij}} \quad (6.5)$$

$$C^l(\tilde{x}_{ij}) = \frac{I(L, \rho_{ij}, \sigma_{ij}, 0)}{v_{ij}} \quad (6.6)$$

$$C^d(\tilde{x}_{ij}) = \begin{cases} R_b(\tilde{x}_{ij}), & \alpha_{ij} \in (\alpha_{ij}^{zero} - \alpha_\Delta, \alpha_{ij}^{zero} + \alpha_\Delta) \\ \left| \frac{I(L, \rho_{ij}, \sigma_{ij}, -\alpha_{ij})}{v_{ij}} \right|, & otherwise \end{cases} \quad (6.7)$$

With regards to the slip ratio σ_{ij} , it is defined after two models constructed from experimental data and introduced by Sutoh et al. (2015). These models were constructed from two locomotion subsystems: one using wheels and another using tracks. These two mechanisms were tested on an inclined sandbox, making them work with lunar regolith simulant (Wakabayashi et al., 2009). Figure 6.13 depicts the two functions of slip ratio σ_{ij} that depend on the slope gradient α_{ij} . Moreover, they are expressed in Equation (6.8). With the increase of this slope gradient, both slip ratio functions return higher values for both models, but at different rates. The slip ratio of the Wheel model increases faster than that of the Track model.

⁶http://www.uahirise.org/dtm/dtm.php?ID=ESP_023957_1755 Accessed on 6th December 2021

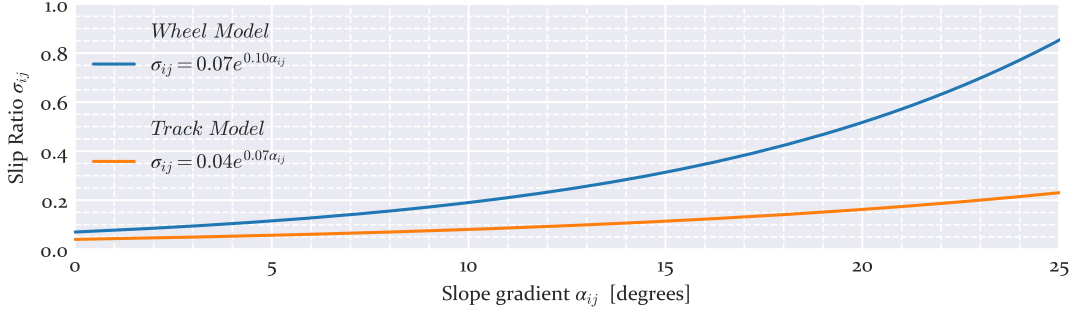


FIGURE 6.13: Slip ratio function of the two models (Wheel and Track) found in the literature (Sutoh et al., 2015), used in the numerical simulation tests.

$$\sigma_{ij} = \begin{cases} 0.07e^{0.1\alpha_{ij}} & \text{if Wheel Model} \\ 0.04e^{0.07\alpha_{ij}} & \text{if Track Model} \end{cases} \quad (6.8)$$

In this way, there are six constant values to define the specific resistance ρ_{ij} and two functions that depend on α_{ij} to define the slip ratio σ_{ij} . Moreover, it is here created an isotropic cost function that is used by a bi-directional version of the FMM. In fact, bi-FMM is equivalent to bi-OUM but with the anisotropy $Y(\tilde{x}_{ij})$ fixed to one, i.e. isotropic. This isotropic cost function takes the highest value of cost from the anisotropic cost function for each value of slope gradient α_{ij} . In other words, the expression for the isotropic cost function $C(\tilde{x}_{ij})$ is equivalent to the CAMIS ascent cost $C^a(\tilde{x}_{ij})$ as shown in (6.9).

$$C(\tilde{x}_{ij}) \equiv C^a(\tilde{x}_{ij}) \quad (6.9)$$

For a better understanding of how the anisotropic and isotropic cost functions are defined in this simulation test, Figure 6.14 is provided. It contains four subfigures, each of them corresponding to a certain configuration of a cost function. Each subfigure is made up of three images. The left image is a 3d view of the cost function, where the below plane is formed by the axes parallel to the ascent-descent and the lateral directions. As a reminder, the descent direction is coincident with the slope aspect direction $\vec{\gamma}_{ij}$. The vertical axis corresponds to the slope gradient α_{ij} . In this way, the cost from 0 (coloured in blue) to 20 (coloured in red) degrees of α_{ij} is defined for all directions relative to the slope aspect. The second image, placed in the middle, depicts the inverse of the cost. It shows the shape of the ellipses that result from each value of slope gradient α_{ij} , using the same scale of colours as in the left image. The right image is a plot showing either the CAMIS functions or the isotropic cost function, together with the anisotropy according to the slope gradient α_{ij} . Figure 6.14a and Figure 6.14b correspond to the case where $\rho_{ij} = 0.15$ and the slip ratio model is the Wheel one. The difference between them is that the first one uses the anisotropic cost function while the second uses an isotropic one. As can be checked, the cost of the isotropic is equal for all directions of the robot, while in the anisotropic case

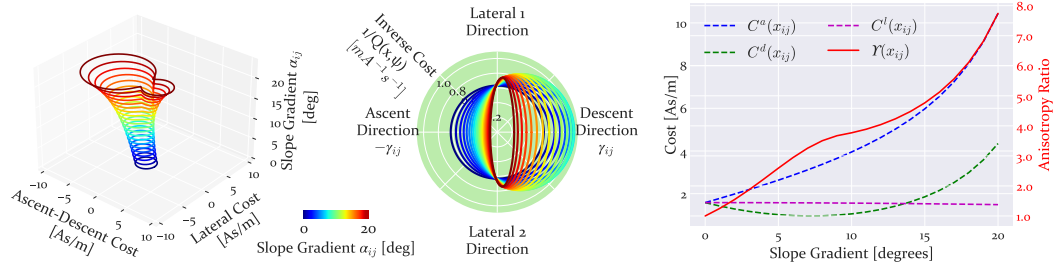
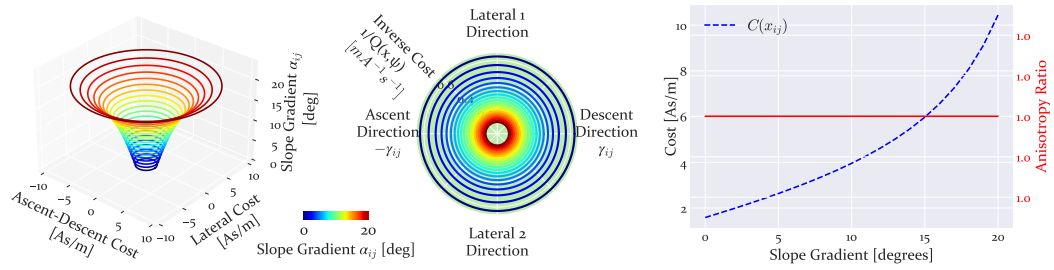
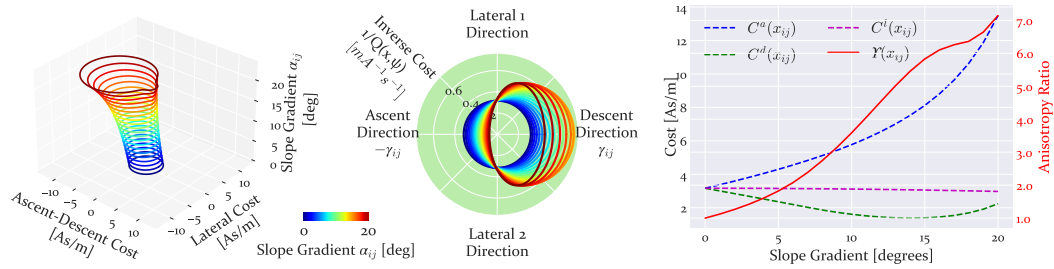
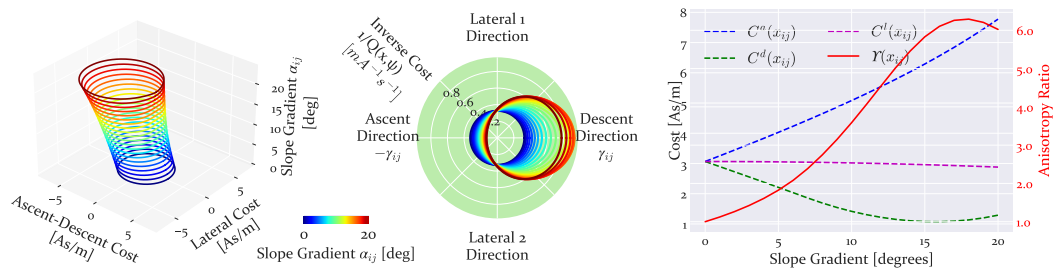
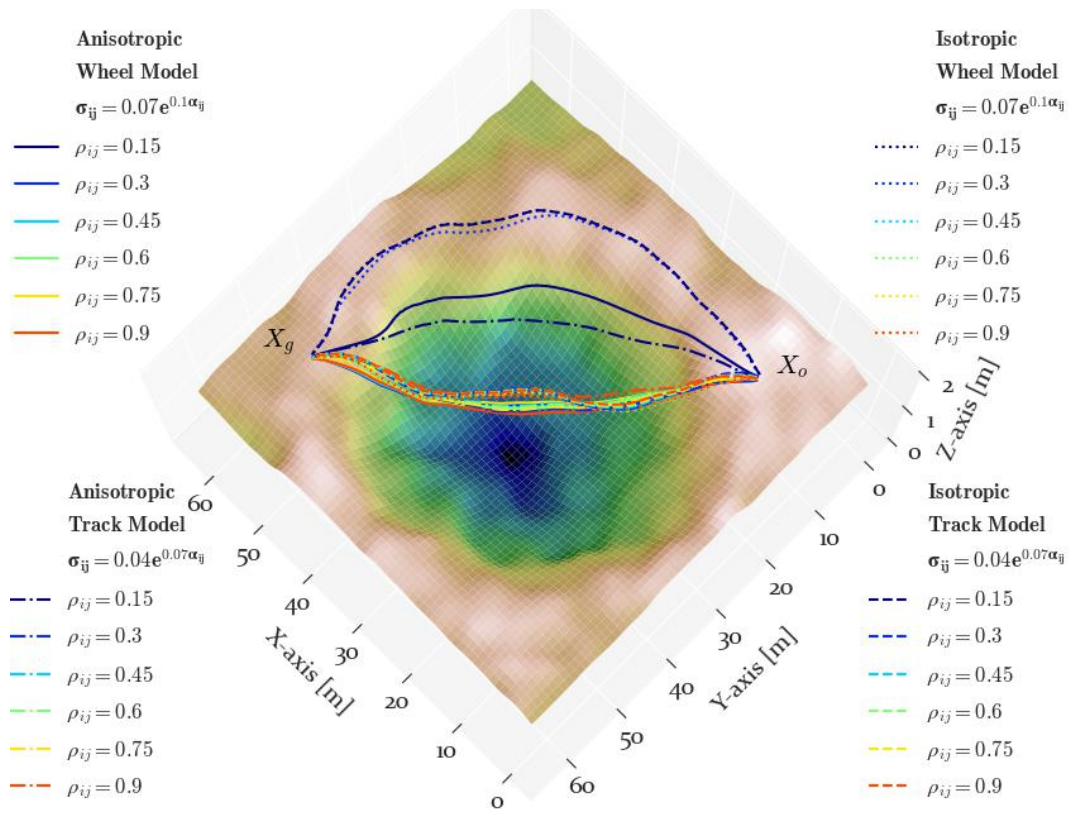
(a) $\rho_{ij} = 0.15$, Wheel model, anisotropic(b) $\rho_{ij} = 0.15$, Wheel model, isotropic(c) $\rho_{ij} = 0.3$, Wheel model, anisotropic(d) $\rho_{ij} = 0.3$, Track model, anisotropic

FIGURE 6.14: Values of cost returned by each anisotropic cost function.

there are differences according to this direction. For values of α_{ij} close to 20 degrees, the inverse of the anisotropic cost takes the shape of an elongated ellipse, while the isotropic cost remains as a circle. In the isotropic case, this circle becomes smaller as the slope gradient increases, while in the anisotropic case this inverse shrinks in the ascent-descent directions, where the relative angle $\beta(\vec{\psi}, \vec{\gamma}_{ij})$ between the heading $\vec{\psi}$ and the aspect $\vec{\gamma}_{ij}$ directions is either 0 or 180 degrees. Figures 6.14c and 6.14d show the anisotropic cost function with ρ_{ij} and each of them using a different slip model. The first uses the Wheel model, while the second uses the Track model. As can be checked, the cost in the first one is higher in the ascent and descent functions (see right images) due to the higher values of slip ratio (see Figure 6.13), while the lateral cost remains the same. This also creates a different anisotropy for values of slope gradient close to 20. The difference in anisotropy is more significant by comparing Figures 6.14a and 6.14c, where an increase in ρ_{ij} entails higher anisotropy.

With the defined anisotropic and isotropic cost functions based on different configurations of terramechanic functions, a series of paths were planned. Figure 6.15a depicts these paths connecting two locations of interest: the origin $\tilde{x}_o = x_o$ and the goal $\tilde{x}_g = x_g$. As can be seen, those paths created with low values of ρ_{ij} , between 0.15 and 0.3, go through different places than the rest. The isotropic cost functions with low ρ_{ij} get further from the slopes and surround the crater, taking more distance to reach the goal. The paths with low ρ_{ij} and generated using an anisotropic planner, bi-OUM, traverse laterally the slopes, keeping the same elevation. This is because although the ascent and descent costs are high, the lateral cost is still lower (see Figure 6.14a) and the anisotropic planner acknowledges this difference in the cost. To do a deeper insight into how the total cost function is calculated, Figures 6.15b and 6.15b depict the solution calculated by two configurations: one anisotropic and one isotropic. The paths with anisotropic cost are calculated using bi-OUM, while bi-FMM is used with those using an isotropic function. As a side note, the grid $\tilde{\Omega}$ used in all cases is a hexagonal regular one, with a resolution Λ of 0.5 m. In Figure 6.15b the bi-OUM makes the wave propagation from the origin enter the crater. This is because it acknowledges the descent cost that is cheaper than the lateral and ascent costs as also seen in Figure 6.14c. On the contrary, the bi-FMM uses the isotropic cost from (6.9) that prevents the wave from propagating towards the crater slopes, as it does not address the differences in cost according to direction. For this reason, in the anisotropic case (with $\rho_{ij} = 0.3$ and using the Wheel model) the path traverses the crater, while the isotropic planner finds another path that circumvents the crater by sticking to horizontal surfaces as much as possible.

Tables 6.6 and 6.7 disclose the values of certain metrics for each configuration used. The first row refers to the number of times the total cost of a node has been updated, either using the explicit equation of the eikonal, expressed in (3.32), for the isotropic case with bi-FMM or the semi-lagrangian approach from (5.8) and (5.9) for the anisotropic case with bi-OUM. This metric gives an idea about the computational load of the planner when calculating the solution using both algorithms. As



(a) 3d view of the scene with the resulting paths.

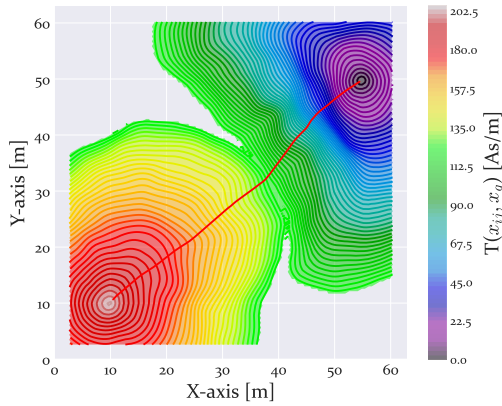
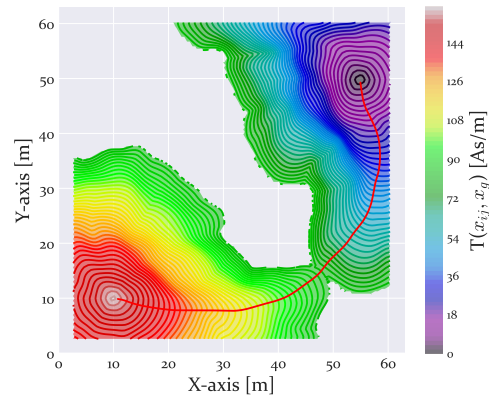
(b) $\rho_{ij} = 0.3$, Wheel model, anisotropic.(c) $\rho_{ij} = 0.3$, Wheel, isotropic Model.

FIGURE 6.15: Results from the first test using anisotropic and isotropic cost functions. The resulting paths connecting two locations, \tilde{x}_o to \tilde{x}_g , are depicted (a). The origin is located at (10 10) m while the goal is at (55 50) m. Note that the 3d view is rotated to provide a better perspective of the obtained paths. The total cost calculated by bi-OUm (a) and bi-FMM (b).

TABLE 6.6: Results of the first anisotropic test with the Wheel model. Each algorithm is executed using its bi-directional version. The first row is the number of times a total cost is updated for a node. The second row is the total cost estimated by the planner. The third row is the corresponding total cost after integrating the anisotropic cost function along the resulting path, even for those paths obtained using bi-FMM. The fourth row is the error committed in the estimation with respect to the result from the integration.

	$\rho_{ij} = 0.15$		$\rho_{ij} = 0.3$		$\rho_{ij} = 0.45$		$\rho_{ij} = 0.6$		$\rho_{ij} = 0.75$		$\rho_{ij} = 0.9$	
	OUM	FMM	OUM	FMM	OUM	FMM	OUM	FMM	OUM	FMM	OUM	FMM
Number of total cost updates	2070887	101127	1750304	117766	924924	118931	625111	116964	517636	115774	471823	115053
Estimated total cost [A s m ⁻¹]	104.90	157.01	200.60	282.78	297.96	392.35	400.99	496.55	499.34	597.62	599.35	704.95
Evaluated total cost [A s m ⁻¹]	108.55	122.91	199.60	241.09	298.07	303.09	398.28	404.07	498.65	503.58	598.22	604.25
Evaluation error [%]	-3.48	21.71	0.50	14.74	-0.03	22.74	0.67	18.62	0.13	15.73	0.18	14.28

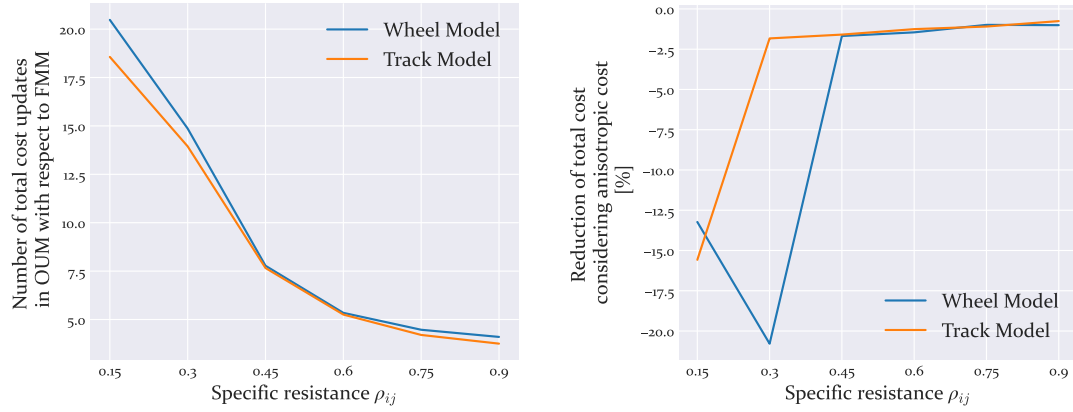
TABLE 6.7: Results of the first anisotropic test with the Track model. Each algorithm is executed using its bi-directional version. The first row is the number of times a total cost is updated for a node. The second row is the total cost estimated by the planner. The third row is the corresponding total cost after integrating the anisotropic cost function along the resulting path, even for those paths obtained using bi-FMM. The fourth row is the error committed in the estimation with respect to the result from the integration.

	$\rho_{ij} = 0.15$		$\rho_{ij} = 0.3$		$\rho_{ij} = 0.45$		$\rho_{ij} = 0.6$		$\rho_{ij} = 0.75$		$\rho_{ij} = 0.9$	
	OUM	FMM	OUM	FMM	OUM	FMM	OUM	FMM	OUM	FMM	OUM	FMM
Number of total cost updates	1987053	107034	1644090	117920	881299	115064	600728	114301	488151	116146	435266	115957
Estimated total cost [A s m ⁻¹]	100.21	150.33	183.35	259.91	272.94	354.56	365.26	448.82	457.44	543.09	551.74	637.27
Evaluated total cost [A s m ⁻¹]	102.04	117.94	182.73	186.08	273.27	277.62	365.33	369.91	457.12	462.12	550.79	554.95
Evaluation error [%]	-1.82	21.54	0.33	28.40	-0.11	21.69	-0.01	17.58	0.06	14.90	0.17	12.91

can be denoted, the values in the anisotropic case with bi-OUM are much higher than those calculated using bi-FMM. This corresponds more or less with the computational complexities of each algorithm, where the OUM is proportional to the FMM but several times higher in the function of the anisotropy. Figure 6.16a indicates how many times the bi-OUM updates the values of total cost with respect to the number of times taken by the bi-FMM. As can be checked in this Figure, for low values of specific resistance ρ_{ij} the number of updates is around 20 times higher than in the isotropic case. This is because, as discussed before regarding the plots in Figure 6.14, the anisotropy increases as the specific resistance is lower, especially when it gets close to zero. As can be also noticed in both tables, this very high anisotropy also produces more error in the total cost that is initially estimated by the planner and the result from integrating the anisotropic cost function along the path. With ρ_{ij} , for the Wheel model case, this error is -3.48% , while for the Track model this error is -1.82% . The errors in the FMM cases are much higher because the initial estimation did not account for the differences in the cost according to the robot direction, as the cost function used was isotropic. Although the computational load is much higher for the case of bi-OUM, it produces paths that save up more total cost (i.e. more energy) than those paths produced by bi-FMM. However, the significance of this reduction in the total cost varies with the specific resistance. For lower values of this terramechanic function, the reduction is higher, as demonstrated in Figure 6.16b. This reduction is calculated according to the values provided in the tables using Equation (6.10). For ρ_{ij} this reduction is close to -13% for the Wheel model and -15% for the Track model. For values of specific resistance ρ_{ij} between 0.15 and 0.45, there is some significant difference between the models. The reduction in the Track model decreases rapidly until being around -2.5% . On the contrary, the Wheel Model at $\rho_{ij} = 0.3$ experiences a reduction of around -20.0% .

$$Reduction[\%] = \frac{OUM \text{ evaluated Total cost} - FMM \text{ evaluated Total cost}}{FMM \text{ evaluated Total cost}} \times 100 \quad (6.10)$$

After thoroughly analysing these results, it is concluded that the use of the anisotropic cost function in scenarios with terrains presenting a specific resistance higher than 0.45 is not recommended. This is because the reduction in the total cost with respect to the isotropic case is not worth the increase in the computational load of the planner. For values of specific resistance lower than 0.45, the slip ratio will determine whether the reduction is significant or not. The Wheel model increases the ascent and descent cost functions at a rate higher than the Track model, and this is why in Figure the reduction of around -20.0% for $\rho_{ij} = 0.3$. If the specific resistance is around 0.15 the reduction is still significant. However, special care must be taken to the high computational load as well as the increase in the errors in the estimation of the total cost made by the planner, which can be reduced by refining the grid. Nevertheless, refining the grid also entails even more computational load as more



(a) Number of times OUM takes more computational load (total cost updates) than the FMM according to the specific resistance ρ_{ij} and each slip ratio model.

(b) A measure of how significant is to consider the directional dependency of the cost according to the specific resistance ρ_{ij} and to each slip ratio model.

FIGURE 6.16: Comparison between the use of bi-OUM and bi-FMM to plan paths on inclined surfaces. The number of times the total cost is updated serves to check the computational load of both approaches (a). The omission of the information in the descent and lateral directions of the isotropic cost function used by bi-FMM is relevant or not according to the terramechanic parameters (b).

nodes have to be visited. Therefore, it is up to the user to decide if it is worth it or not. For example, for the case in which this planning is carried out offline, this may not be a problem, but it may be intractable for online planning in the onboard computer of a robotic vehicle.

Experimental Setup

The simulation and field tests that are presented later in this section are based on a real experimental setup. A DEM is taken out of a terrain presenting slopes using photogrammetry software and geo-referenced aerial imagery. This terrain, part of a dedicated $90000m^2$ outdoor experimental area located at the University of Malaga campus, is shown in Figure 6.17. This is an off-road unstructured scenario that is meant to be used for emulating Search and Rescue operations and testing novel robotic solutions. The Chair of Safety, Emergencies and Disasters at the University of Malaga organizes yearly exercises in which a large-scale disaster response is simulated (Mandow, Serón, et al., 2020). The terrain in question is selected because it has a slope that proves useful to test anisotropic path planning. Figure 6.17d shows a virtual reconstruction of this terrain, a 3d model built with the Pix4Dmapper software (version 4.6.4) (Barbasiewicz et al., 2018). This software is known for creating 3d models of terrain from georeferenced imagery. The area covered by the aerial imagery has an extension of 0.012 square kilometres, with a resolution of 0.56 cm per pixel on average. With an RTK GNSS *Emlid Reach M+* module the model was rectified to reduce the error location. This module was connected to the Andalusian Positioning Network (RAP) (Páez et al., 2017) and provided an RTK accuracy



(a) Panoramic view of the slope considered for the tests.



(b) View of half of the slope.



(c) View from the other side.

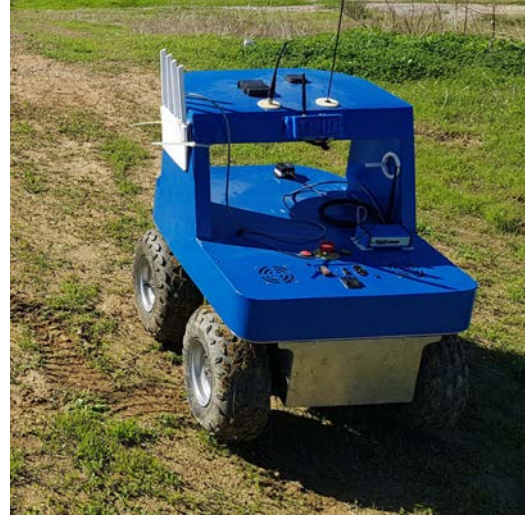


(d) Screenshot showing the 3d model built using the Pix4Dmapper software, together with red arrows indicating the location of the four points of interest used in the tests.

FIGURE 6.17: Showcase of pictures showing the terrain using different perspectives (a)(b)(c) and a screenshot of Pix4Dmapper showing its virtual reconstruction (d).



(a) Frontal view of Cuádriga robot.



(b) Rear view of Cuádriga robot.

FIGURE 6.18: The skid-steering four-wheeled Cuádriga robot.

TABLE 6.8: Specifications of the Cuádriga robot

Parameter	Value
Mass	83 kg
Height	0.81 m
Battery	36 V
Capacity	24 Ah
Controller	Roboteq model AX1500
Current Sensor	ACS754 board + Arduino AtMEGA board
Localization Sensor	RTK GNSS Emlid Reach M+

of 10 cm and even less. The rectification allowed the removal of 2.8 m of the wrong displacement in the UTM-Y axis. Next, with the Pix4Dmapper software, the point cloud was processed and a regular square grid was obtained, having a resolution of 0.1 m. This square grid was later reformatted as a hexagonal grid with a resolution $\Lambda = 1m$.

The CAMIS functions $C^a(\tilde{x}_{ij})$, $C^l(\tilde{x}_{ij})$ and $C^d(\tilde{x}_{ij})$ are built for a robot in particular. This robot is Cuádriga, a four-wheeled platform that drives using skid-steering locomotion. This means each pair of wheels receives a single-speed command. Figure 6.18 shows the pictures of this robot, one of its frontal side (Figure 6.18a) and another of its rear side (Figure 6.18b). Cuádriga was built at the University of Malaga as an experimental mobile testbed to perform inspection and recognisance tasks in disaster response tests. It is also used to validate algorithms and methods for mapping and autonomous navigation (Martinez et al., 2013). Table 6.8 presents some of the specifications of this robot. More data about this robot can be found in the work of Jesús Morales et al. (2010).

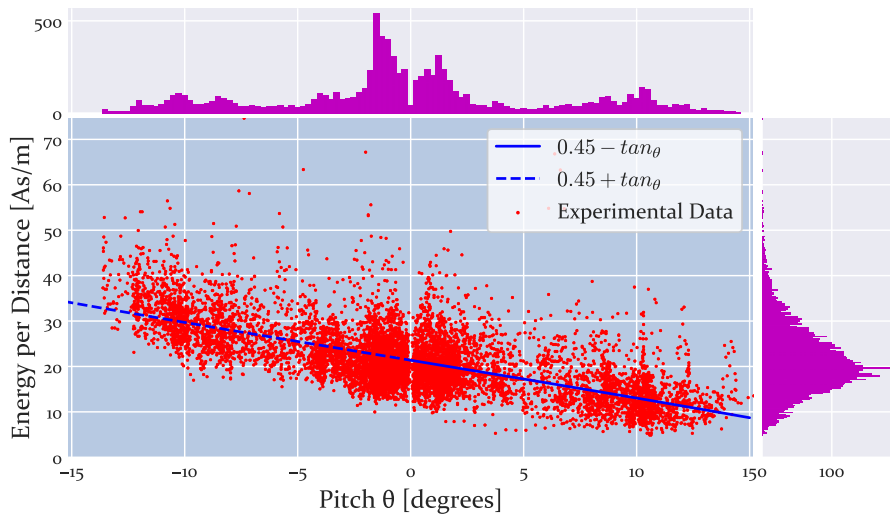
As a preliminary step, some runs with the robot were made on slopes close to the terrain used for the experiments. One of these slopes can be seen in Figures



(a) Cuádriga climbing a slope.



(b) Cuádriga descending the same slope.



(c) Samples of energy per distance obtained from preliminary drives of Cuádriga.

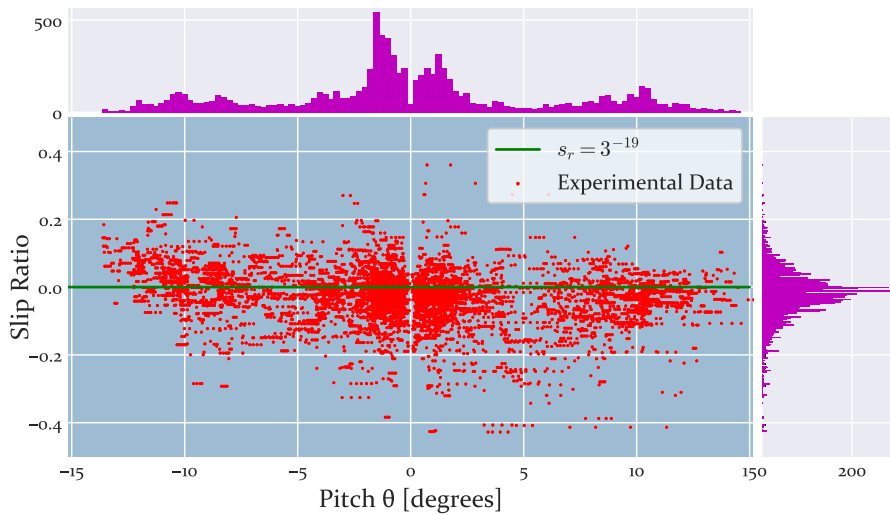
(d) Samples of slip ratio σ_{ij} obtained from preliminary drives of Cuádriga.

FIGURE 6.19: Extraction of ρ_{ij} and σ_{ij} based on preliminary drives of Cuádriga. The plots (b) and (c) are accompanied by histograms that indicate the density of samples along the slip ratio (vertical) and pitch (horizontal) axes. The pitch Θ is negative when the robot is ascending.

6.19a and 6.19b, with Cuádriga driving on top of it. By interpolating the position of the robot on the DEM its orientation was calculated. Figure 6.19c illustrates the recorded current consumption with the pitch angle at the moment each sample was taken. The reference frame of the robot has the X-axis pointing to the front, while the Z-axis points upwards. Using this convention, the negative values of pitch mean the robot is ascending. From the recorded current data, it is clear the current consumption increases when the robot ascends and decreases when it descends. Besides, from this data it is estimated a value of 0.45 for the specific resistance ρ_{ij} . Other relevant parameters are the speed (0.5 m/s), the fraction $m_b d_r / K_w$ (2.43 A·s²/m) and the gravity acceleration g (9.8 m/s²). Figure 6.19d depicts the slip ratio recorded data. It is negligible and hence, σ_{ij} is set to 0. The main justification for this is that the soil in this terrain is compact, which makes the wheels adhere better to the surface. Since the slip effects are not significant and the value of ρ_{ij} is far from zero, it could be deduced at first that CAMIS in this case would not be necessary for energy minimization purposes, as it was demonstrated in the previous simulation that the gain is not worth the increase in the computational load, especially when the slip ratio is as low as zero. However, another criterion may still justify the use of an anisotropic cost function for navigating in irregular terrains: the minimization of roll angle to preserve lateral stability.

Roll Minimization

Prior to the field tests, some simulations were carried out to demonstrate how the roll is minimized. The roll weight function w_{ij}^Φ affects the cost function and serves to penalize the angle of roll Φ in the overall planned course when there are slopes. In this simulation test, it is checked how by varying w_{ij}^Φ the resulting path is modified, either to go parallel to the direction of the slopes or traverse them diagonally. As stated before, this is key to preventing the robot from turning over. The map used for this simulation is based on a portion of the DEM depicted in Figure 6.17. Four red arrows are placed in Figure 6.17d to mark the location of the points that will serve for both origin and goal positions of the test. Two of them are placed on the lower area while the other two are placed on the higher area. Figure 6.21 also indicates, together with the resulting paths that will be later explained, these four locations, referred to as x_a , x_b , x_c and x_d . The information about how the slope gradient α_{ij} and the aspect direction $\vec{\gamma}_{ij}$ are defined on each position of the terrain is illustrated in Figures 6.20a and 6.20b respectively.

The paths depicted in Figure 6.21 and connecting the four locations of interest were generated based on different configurations of the roll weight function: $w_{ij}^\Phi = 1$, $w_{ij}^\Phi = 1 + 3 \tan \alpha_{ij}$ and $w_{ij}^\Phi = 1 + 6 \tan \alpha_{ij}$. These expressions are formulated in that way to be linear with the gradient α_{ij} , as depicted in Figure 6.22. All the paths were planned following two sequences that made them pass through all points of interest. Both sequences are indicated in Figure 6.21, informing about from which

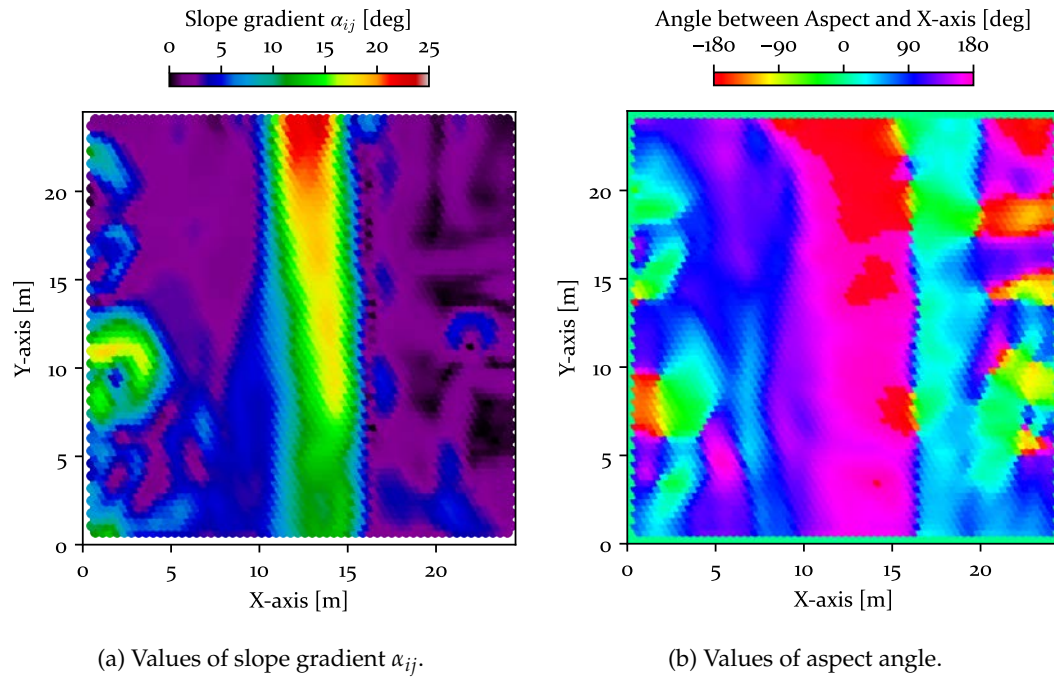


FIGURE 6.20: Slope data describing the shape of the experimental terrain after smoothing the DEM with an average filter.

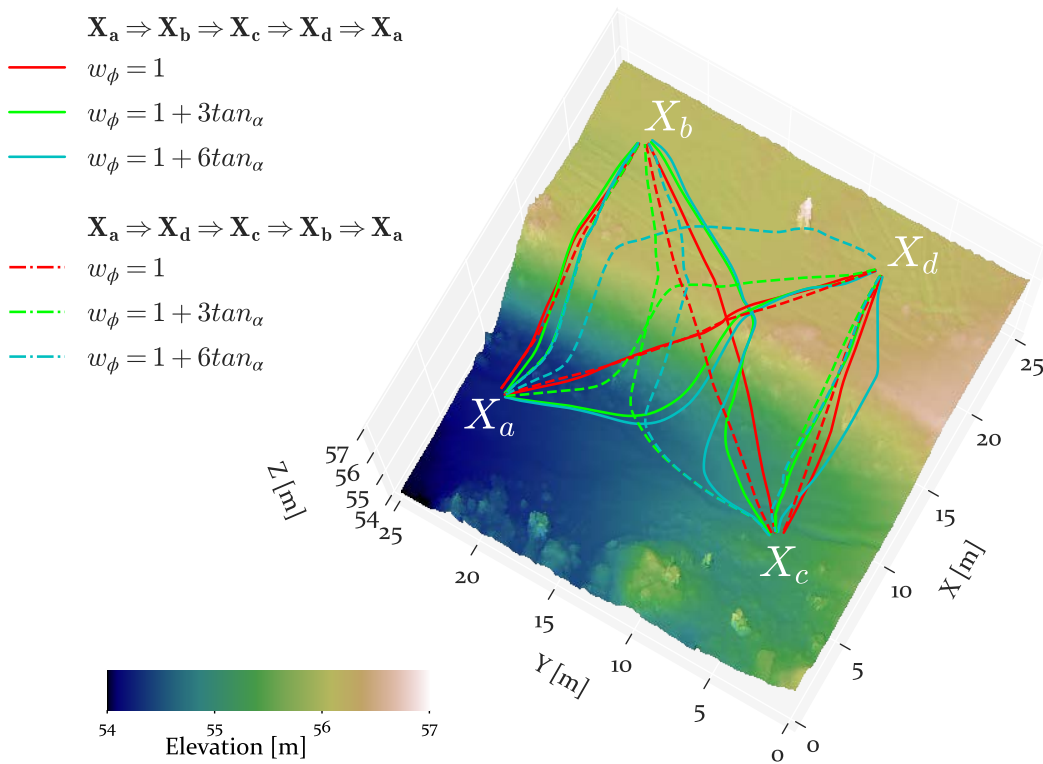


FIGURE 6.21: Paths traversing the slope with certain configurations of weight values. Here $w_\phi = w_{ij}^\Phi$ and the positions of interest are grid nodes, i.e. $x_a = \tilde{x}_a$, $x_b = \tilde{x}_b$, $x_c = \tilde{x}_c$ and $x_d = \tilde{x}_d$.

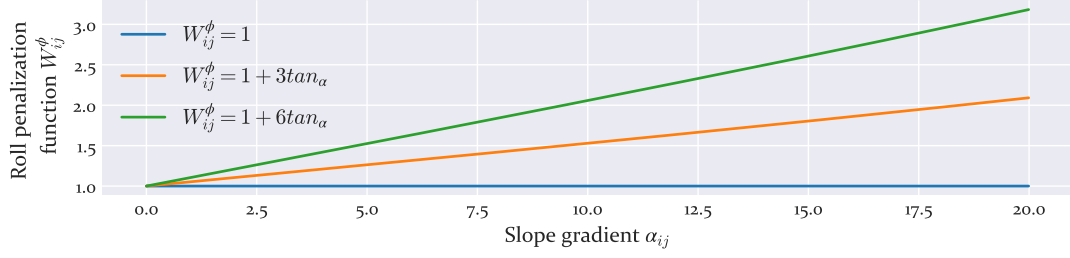


FIGURE 6.22: Roll penalization functions used to minimize the roll angle.

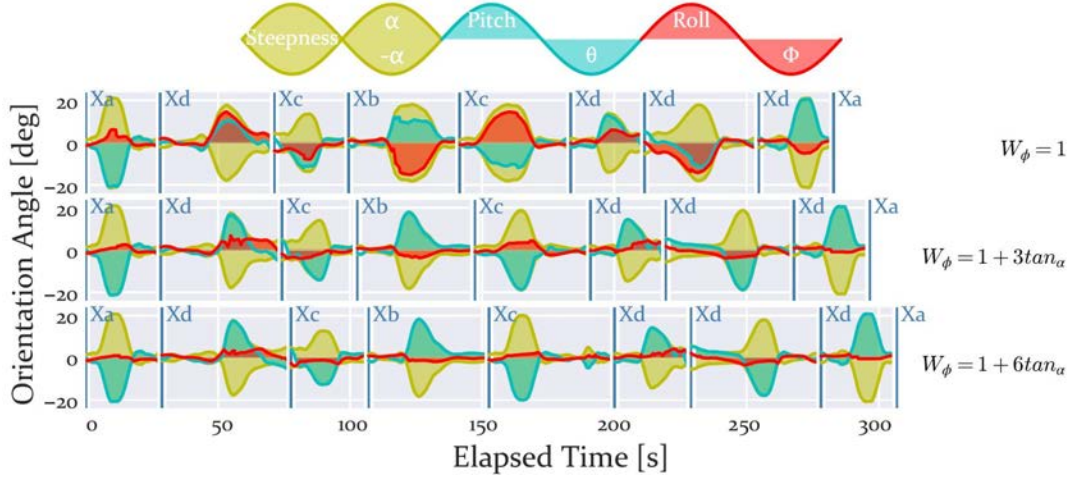


FIGURE 6.23: Orientation angles produced by each round-trip travel from the three models.

point to which point the path goes. Each path connecting two points of interest necessarily makes the robot either ascend or descend the slope in the middle.

The value returned from the roll weight function is equal to one whenever the slope gradient α_{ij} is zero. When α_{ij} increases, two of the proposed roll weight function increase as well, but at different rates (see Figure 6.22). As mentioned, the options tested in this simulation are formulated so they are approximately linear according to α_{ij} . However, there is still room for many other kinds of expression to define the roll weight function (polynomial, using other trigonometric operations, ...). These other options make the weight function increase at different rates, but they are not considered here and are left for future work. An important fact to have in mind is that this rate affects how the paths are more or less parallel to the slope aspect direction. Figure 6.23 plots the expected orientation angles of the robot for each of the sequences of paths and all options of w_{ij}^ϕ . As can be seen, the roll angle is reduced along the traverse, although the elapsed time increases with the weight function. Therefore, a trade-off arises between minimizing roll and not driving longer distances, spending more time. Figure 6.24 plots the distance accumulated by all paths where the roll is higher than a threshold denoted by the X-axis. Here, the planner using $w_{ij}^\phi = 1 + 6 \tan \alpha_{ij}$ only accumulates distance with roll under 4 degrees, meaning the robot would keep its value of roll under that value in all paths.

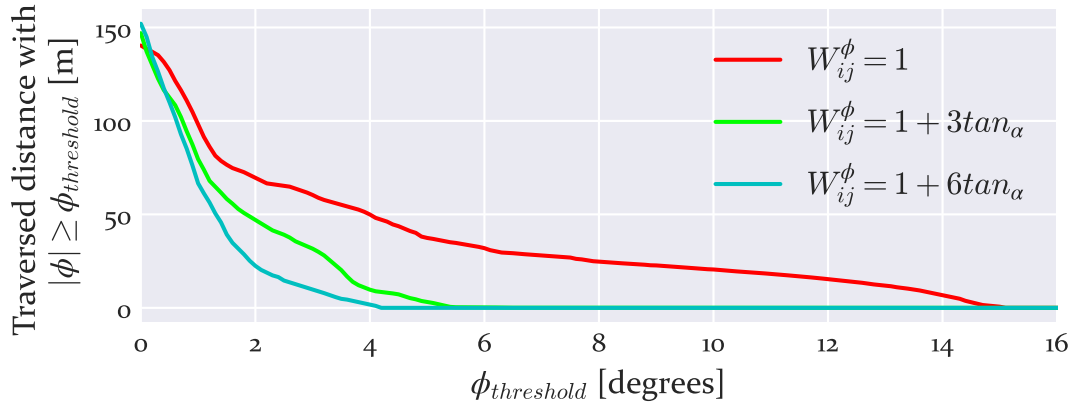


FIGURE 6.24: Traversed distance with an absolute value of roll Φ higher than the threshold indicated in the horizontal axis.

Field Tests

The next and final test was performed with the Cuádriga robot on the real terrain depicted in Figure 6.17. The main purpose of this test is to detect what the use of CAMIS with an anisotropic path planner is missing when being landed onto reality. Cuádriga was commanded to follow some of the calculated paths from the previous simulation. The paths in question are those that were generated using the roll weight function $w_{ij}^{\phi} = 1 + 6 \tan \alpha_{ij}$. These paths connect x_a with x_d and x_b with x_c . Such paths go in two ways, meaning that Cuádriga went from x_a to x_d and thereafter from x_d to x_a . The same for the paths between x_b and x_c . The complex dynamics of the Skid-steering locomotion, together with the path tracking controller, are sources of error that will be revealed in the experiments, as they were not addressed when planning the paths. This is significant considering the paths take lengths of approximately 15 to 25 meters. The focus is put on the current consumption of the robot and the orientation angles it experiences while following the paths.

Figure 6.25 illustrates the planned paths, the position samples of Cuádriga measured while it was driving and the four points of interest x_a , x_b , x_c and x_d . This figure depicts a 3d photo-realistic reconstruction of the area where the experiments were carried out. The trajectories Cuádriga did are close to the planned paths, thanks to the use of a tracking algorithm called *Pure-Pursuit* (S. F. Campbell, 2007). This algorithm was implemented on the Cuádriga onboard computer using the *LabView* software. Its look-ahead distance was set for all experiment runs to 2 meters and the speed commanded to the robot to 0.5m/s. The orientation of the robot was estimated based on the readings of the RTK GNSS *Emlid Reach M+* module equipped by the robot and their interpolation on the DEM depicted in Figure 6.17d. The execution of these experiments was recorded and can be accessed online⁷. Cuádriga followed each of the paths twice and, in each execution, it started from a position slightly far from the origin position. This was made to avoid recording the high current consumption that occurs when the robot starts its movement. Besides, in each execution

⁷https://youtu.be/vJx_v2GR1Sc

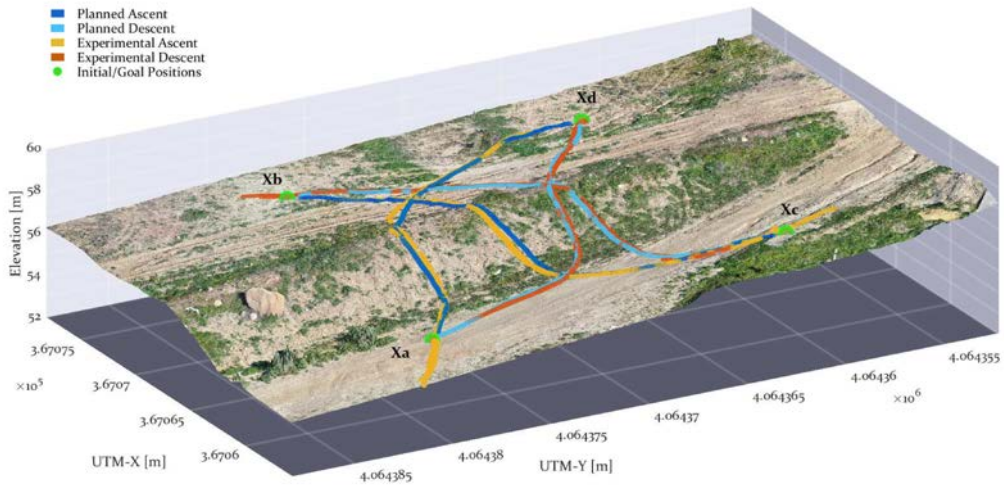
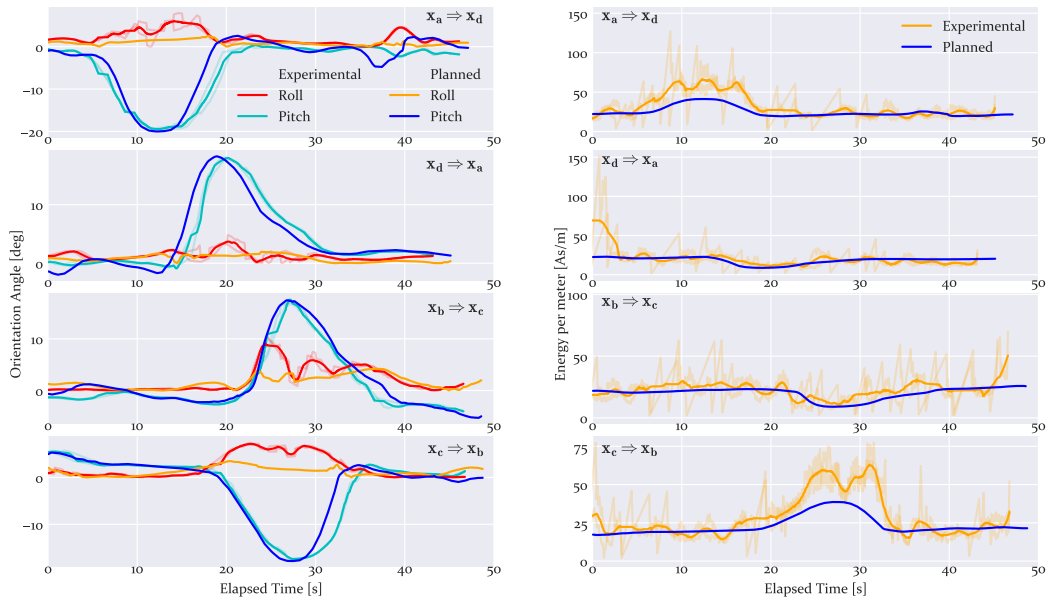


FIGURE 6.25: Paths resulting from the second test with anisotropic cost and the placement of the position samples of Cuádriga, measured with a RTK GNSS antenna.



(a) Comparative of the orientation angles between the estimate of the planned paths and the experimental results from two executions with Cuádriga. The raw experimental data has some transparency, while the solid line is the smoothed average of both executions.

(b) Comparative analysis of the cost/speed between the estimate of the planned paths and the experimental results from two executions with Cuádriga. The raw experimental data has some transparency, while the solid line is the smoothed average of both executions.

FIGURE 6.26: Comparative analysis of the orientation angles and the cost between planning estimations and Cuádriga experience.

Cuádriga stopped at a distance coincident with the Pure-pursuit look-ahead to the goal position.

Figure 6.26a presents a comparison of the orientation angles that were foreseen according to the planned paths and those experienced by Cuádriga in the experiments. The recorded data from the two executions per path (with some transparency in the figure) was averaged and smoothed using a moving mean filter. The two compared orientation angles are the pitch Θ and the roll Φ . It is recalled here that given the local reference frame of the robot the negative values of pitch indicate the robot is ascending, while it descends on the contrary case. Besides, if the roll is negative this means the robot is inclined towards the left side, while it is inclined towards the right side in case the roll is positive. The elapsed time in the X-axis is adjusted so the zero starts when the robot reaches the initial position. As mentioned, the robot stops a certain distance from the goal position and this can be noticed in Figure 6.26a. From the presented results in this comparative, it seems the pitch angle experienced by Cuádriga is close to the foreseen angle. However, the experienced roll angle slightly varies from the expected one, with more pronounced deviations in the paths between x_b and x_c . This is more clear in Tables 6.9 and 6.10. The former presents the mean absolute error (MAE) and the latter the root mean squared error (RMAE) of the recorded data. This error analysis highlights that the roll has a higher error than the pitch. The main reason this happens is that the planned paths create too pronounced curves when entering the slope. This is because of how the cost abruptly changes from isotropic to heavily anisotropic. The *Pure-Pursuit* cannot, with the values of look-ahead and speed set, make the robot perform those pronounced curves and therefore some tracking errors arise. This is easier to check in Figure 6.25 in the path where the robot starts from x_a and goes to x_d . Since the robot, when it attacks and leaves the slope, makes a smooth turn the angle of roll experiences an increase.

Figure 6.26 depicts the current consumption per meter, obtained after dividing the current consumption by the robot speed, experienced by Cuádriga and compared to the plan prediction. The high consumption at the beginning of the x_d to x_a traverse is due to the robot starting in that particular case from the origin position with initial zero speed, instead of a few meters before reaching it. As can be checked, the descent prevision is more accurate than the ascent, an affirmation that is supported by the error analysis presented in Tables 6.9 and 6.10. The existing inaccuracy may be caused by the differences in the terrain conditions when creating the cost function and when performing these tests. For example, the presence of vegetation and the pressure of the wheel may have been different. Moreover, when the robot enters and exits the slope the consumption is higher. This is more clear in the x_c to x_b traverse. This is probably due to the turning cost of the vehicle, which uses Skid-steering, and was not considered in the anisotropic cost function.

From the last experiments, it can be concluded that landing CAMIS to reality is promising to produce plans for long traverses where keeping the robot parallel to the slope directions is a priority. At a local scale, CAMIS should be refined to

TABLE 6.9: Mean Absolute Error (MAE) of experimental data results with respect to planned data

	$\mathbf{x}_a \Rightarrow \mathbf{x}_d$	$\mathbf{x}_d \Rightarrow \mathbf{x}_a$	$\mathbf{x}_b \Rightarrow \mathbf{x}_c$	$\mathbf{x}_c \Rightarrow \mathbf{x}_b$	Total
Roll [deg]	1.35	0.72	1.05	1.62	1.21
Pitch [deg]	2.10	1.21	0.99	1.27	1.41
Energy per meter [As/m]	7.87	4.26	4.34	7.13	6.02

TABLE 6.10: Root Mean Squared Error (RMSE) of experimental data results with respect to planned data

	$\mathbf{x}_a \Rightarrow \mathbf{x}_d$	$\mathbf{x}_d \Rightarrow \mathbf{x}_a$	$\mathbf{x}_b \Rightarrow \mathbf{x}_c$	$\mathbf{x}_c \Rightarrow \mathbf{x}_b$	Total
Roll [deg]	1.83	0.90	1.59	2.29	1.76
Pitch [deg]	2.96	1.88	1.34	2.36	2.25
Energy per meter [As/m]	10.81	7.27	5.41	10.58	8.99

address the constraints in dynamics imposed by the locomotion subsystem as well as the errors that may be produced by the path tracker.

6.5 Summary and Conclusions

This chapter presents the results from the experiments carried out to validate the thesis contributions disclosed in previous chapters. In particular, three of the contributions were validated, each of them corresponding to a different improvement in the use of PDE solving algorithms for path planning.

The first contribution, the use of a cost function based on the dynamic modelling of two locomotion modes, was validated by running simulations on two scenarios. The first scenario presents a customized terrain distribution. It serves to highlight the advantage of using a reconfigurable rover along with a planner that leverages its different locomotion modes. The results from this simulation test confirm this statement. The FMM was executed to find energy-minimizing paths. The consideration of an alternate mode of locomotion exploited the higher adaptability of the reconfigurable rover and let the path planner find a path that demanded less energy. Besides, the obtained results serve to validate the formulation of an isotropic cost function that employs dynamic models of two locomotion modes. The second scenario introduces more complexity by combining different terrains with obstacles and producing a series of paths from different positions. It served as a reaffirmation of the conclusions from the previous test but this time considering obstacles in the cost map. It was analyzed, for every grid node, what was the reduction in the total cost from having available only one locomotion mode to having two.

Later on, another simulation test and a field test serve to validate the second contribution of this thesis. One of the paths computed in the previous simulation test

was selected as the path to be repaired. Then, the functioning of the LPR using the Sweeping approach is analyzed by simulation means considering different arrangements of obstacles that are scattered on the path in question. The results show an existing trade-off between the local layer resolution and the computational load of the planner. The other approach, the Conservative, is studied with a rover prototype on a real terrain containing obstacles to avoid. The results of the field experiment clearly show how the rover always searches for a way to continue following the original path minimizing the deviation due to the sudden detection and avoidance of obstacles.

The simulation and field tests performed using the CAMIS functions for anisotropic path planning have proven to be useful to clarify in which situations their use is more advantageous than simply relying on isotropic cost functions when traversing terrain that includes slopes. The first simulation focuses on using CAMIS for finding the paths that require the least energy in a scenario modelled after a Martian crater. The main conclusion extracted from the results of this simulation is that a high slip ratio produces not only higher cost in the ascent and descent CAMIS functions but also high anisotropy. For scenarios containing slopes with less than 20 degrees of maximum gradient, terrains presenting a specific resistance of less than 0.45 may justify the use of the CAMIS functions at the expense of increasing the computational load. The second simulation serves to validate the use of the roll weight function to minimize the roll. This function increases the cost in the lateral directions given increasing values of slope gradient α_{ij} . In exchange of an increase in the total cost (energy in this case), the roll weight function was demonstrated to prioritize the preservation of the roll angle during the traverse of slopes by making the paths go parallel to the slope aspect direction. Finally, some of the paths from this last simulation were taken as a reference for the autonomous navigation of Cuádriga, a four-wheeled robot. Cuádriga traversed the same slope that was modelled and used in the last simulation, tracking four of the resulting paths. The results that were obtained from this field test reveal some difficulties encountered when landing CAMIS on reality. They include the error produced by the path tracking component and addressing omitted terrain features such as the roughness. Nevertheless, the overall roll and energy consumption were relatively close to the expectations of the planner, after making a comparison between the recorded data and the plan.

Chapter 7

Conclusions and Future Work

"And once the storm is over, you won't remember how you made it through, how you managed to survive. You won't even be sure, whether the storm is really over. But one thing is certain. When you come out of the storm, you won't be the same person who walked in. That's what this storm's all about."

Haruki Murakami
Kafka on the shore, 2002

7.1 Introduction

This text forms the seventh and last chapter of the thesis. This chapter provides the reader with a series of conclusions extracted from the presented contents. Moreover, it discloses at the end ideas that may be of interest to extend and/or improve the thesis contributions in the future.

This thesis served to expose the work carried out to advance in the state of the art of path planning in the context of ground exploration robots. The main motivation of this work appears many times throughout the text. This work targets robots such as planetary rovers and robots for Search And Rescue (SAR) applications. The autonomy of the rovers is growing as the number of missions to visit extraterrestrial places increases. The autonomy of the SAR robots is of interest for future first response operations in the wild. Chapter 1 explained the main justification to increase autonomy. It is to remove the dependency on human intervention. In this way, a robot can perform a larger number of tasks within a time window, such as exploring a place. Part of the autonomy in navigation is path planning. The locomotion capabilities and the awareness of the scenario play important roles in it. Besides, making a planner account for the irregularities in the terrain is a valuable asset. This thesis introduces improvements in path planning to better acknowledge all these features. These improvements make up the first three contributions. As a reminder, Chapter 1 introduced a total of five contributions. The first three of them are the core of this

thesis, advancing the state of the art in path planning. Chapters 3, 4 and 5 detailed them. Chapter 6 presented the fourth contribution. It displayed details about the making of experiments with real prototype mobile robots. These details described the creation of setups to land algorithms from simulation to reality. The results presented confirmed the premises of the second and third contributions. Chapter 2 presented the fifth contribution. It is a classification of the existing path planning algorithms. An exhaustive and comprehensive survey of them led to making this classification.

This chapter comprises two parts. First, Section 7.2, recaps the contributions presented throughout the previous chapters. Second, Section 7.3 discloses ideas for future improvements.

7.2 Thesis Conclusions

The contributions of this thesis constitute a major step in improving the autonomy of mobile robots. Most of the existing PDE planners were only tested within simulation environments. Landing them into reality is a necessary step to identify the main problems a path planned should be aware of. As inquired in the first chapters, efficiency can be critical. It is crucial in extreme applications such as planetary exploration and Search And Rescue (SAR).

This thesis presented advances in several key research lines. Three of them are as follows. First, making a planner aware of the adaptability of reconfigurable robots. Second, managing the scenario uncertainty with information from many sources. Third, modelling the effect of gravity on the traverse of slopes. This section provides next a summary of the main conclusions extracted for each of the different contributions.

Contribution 1: Optimal path planning considering multiple locomotion modes.

Chapter 3 explains the first contribution of this thesis. Besides, Chapter 6 details simulation experiments that support it. This contribution rests on the use of dynamic models of different locomotion modes. These models are combined into a single cost function. This function tackles the mobility skills of mobile robots able to reconfigure their locomotion. As a result, this cost function allows them to adapt better to the terrain conditions. In particular, Chapter 3 introduces two modes: Wheel-walking and Normal-driving. Each of them is, according to the proposed model, more suitable for different values of terramechanic parameters. The models were built based on wheeled-leg rovers. From the results of the simulations it is deduced that Normal-driving is suitable in energetic terms for terrains with low specific resistance ρ_{ij} and low slip ratio σ_{ij} . On the contrary, for terrains with higher ρ_{ij} it is more efficient the use of Wheel-walking locomotion. The exact values to switch between modes are determined by the dynamic configuration of the robot. The path planner generates

an energy-efficient path based on these modes. Moreover, it determines which locomotion mode is better to reach every waypoint. The simulation results prove how the acknowledgement of both modes allows for finding even shorter paths. This is thanks to recognizing the increase in adaptability. The cost function was built according to the specifications of a real reconfigurable rover. Preliminary dynamic models created via simulation tools served as input to this cost function. Finally, the path planning simulations rested on custom maps and left the way to get the terrain parameters out of the scope.

Contribution 2: Multi-scale path planning combining initial long-traverse planning and dynamic local path replanning.

Chapter 4 explains in detail the second contribution of this thesis. It consists of a multi-scale path planning solution based on the FMM. As mentioned in the analysis of the state of the art in path planning, this kind of algorithm is not usually employed for local planning. Besides, the capability to replan was still in the early stages. There were some scarce approaches aiming at updating the path at the same scale of planning (e.g. global). The second contribution of this thesis proposes using a grid that combines a global and a local layer, the Multi-layered grid. With this grid, the planner can make a very long traverse and also repair it at certain sections with limited environment information. This is thanks to this special grid having the capability to handle maps of different sizes and resolutions. The Dynamic Multilayered path planner (DyMu) arises as a path planning architecture capable to tackle both global and local planning using this grid. It rests on a heuristic version of the FMM to perform the local update of the path. This method is generally not compatible with kinematic restrictions. Yet, DyMu takes advantage of the fact that many rovers are capable to turn while keeping their position. This simplifies the mobility conditions that must be addressed at a local scale since the motion does not need to be bounded to any turning radius. DyMu was validated through plenty of tests. Chapter 6 presented the details of these tests. A rover, HDPR, managed to drive hundreds of meters in a fully autonomous fashion. The Conservative approach of the Local Path Repairing (LPR) was used in these tests. The other approach, Sweeping, was validated through numerical simulation. It remains to be tested using a robotic platform. Moreover, a later conference publication presented an improved version of DyMu. It was written together with the ESA Robotics and Automation Section (Paz-Delgado et al., 2020). This publication details more experiments carried out with the HDPR robot in a round-trip mission. Finally, DyMu has the potential to be integrated with other PDE Solving planners. This is because of the synergy between both layers, through the use of heuristics in the local layer based on global information.

Contribution 3: Creation of a direction-dependant cost function for planning the optimal traverse on slopes.

Chapter 5 introduces CAMIS. It is an anisotropic cost function for PDE path planners on scenarios with inclined surfaces. CAMIS makes the planner find the path that better adjusts to some criteria. For example, minimizing energy consumption or preserving lateral stability. The latter is equal to minimizing the experienced roll angle. To do this, the anisotropic cost function takes into account the aspect direction of the slope and its gradient. The CAMIS cost function is modelled after the inverse of a displaced ellipse. This complies with the requirements of anisotropic PDE planners such as OUM. The results from two simulations and one field experiment served to validate the use of CAMIS. In these experiments, a robot must find the optimal path to traverse scenarios containing slopes. The first simulation served to study how the terrain properties affect the anisotropic cost function. Besides, it serves to figure out if it is worth using an anisotropic planner rather than an isotropic one. The results state that, for scenarios with inclined surfaces, anisotropic planners perform much better when the specific resistance is low and the slip ratio is high. The reduction in total cost (energy consumption) is significant, in exchange for increasing the computational time. Under other conditions, the isotropic is still worthier, for example using FMM, as its computational complexity is much lower. The second simulation served to analyze the use of a custom weighted function to minimize the roll angle. This function increases the cost in the directions perpendicular to that of any slope, i.e. Lateral directions. The results prove how this function makes the path go more parallel to the direction of the slopes. As a result, the obtained path reduces the angle of roll the robot experiences while tracking it. The field experiment put the light on the difficulties to land the current version of CAMIS on local planning. The results show how the tracking errors the robot experiences increase the angle of roll. Besides, the cost function does not address the cost of turning. Still, the robot kept parallel to the slope direction. The estimations of the planner about the orientation and the energy consumption were overall close to the real values.

Contribution 4: Field tests using experimental mobile platforms.

Chapter 6 presents two field experiments, one to check DyMu and another to check CAMIS. The first of them consisted of using a mobile testbed platform in the form of a rover prototype. It has six wheels, four of them steerable, and a rocker-bogie kinematic configuration. It is the HDPR, and was equipped with a frontal *Bumblebee* stereo camera. The navigation software, based on the RoCK robotic middleware, addresses the information coming from the camera. It also commands the motors. A GNSS antenna was installed onboard and a fixed station provided the localization. It provided an RTK precision of a few centimetres or even less. The scenario, a rocky terrain resembling a martian landscape, contains craters bigger than the rover. Besides, rock mock-ups were scattered all over the place. A previous publication

(Gerdes et al., 2020) provides further information about this setup. With regards to the second field experiment, a four-wheeled mobile platform was used. It was equipped with an RTK GNSS *Emlid Reach M+* module connected to the Andalusian Positioning Network (RAP). In this way, the precision in the localization was also in the order of centimetres. An electric current sensor, the *ACS754* board, was installed together with an *Arduino AtMEGA* board. They served to measure the current supplied to the motors and as the basis to build the CAMIS anisotropic cost function. The region of interest had a slope in the middle. This terrain was modelled thanks to the geo-referenced images and photogrammetry techniques. The setup of the two field experiments served to have a better understanding of the proposed path planning solutions. It allowed to take DyMu and CAMIS to real applications.

Contribution 5: General classification of most relevant existing path planning algorithms used along with ground mobile robots.

The last contribution of this thesis complements the previous ones. It results from the time invested in the literature search. This contribution consists of the use of a method to classify path planners. It organizes the path planning algorithms found by the author of this thesis. This classification divides these algorithms into four classes: Reactive Computing, Soft Computing, C-Space Search and Optimal Control. Algorithms falling into the Reactive Computing class are mostly used in path planning for local obstacle avoidance. Soft Computing algorithms are configurable by the user through fuzzy rules, neural networks or the parameters that define Evolutionary techniques. Later on, C-Space Search algorithms extract a path out of a grid. This grid can be dynamically created or static. Finally, Optimal Control algorithms generate the optimal path from a well-known environment. The proposed classification, presented in Chapter 2, serves as a starting point for anyone interested in choosing a path planner for an autonomous navigation application.

7.3 Future Work

This thesis presented a series of advances in the state of the art of path planning. Nevertheless, this is a field that still has room to improve. This section comprises those key ideas that are out of the scope of this thesis but serve as a continuation of the presented contributions. They are explained in full detail next:

- **Improvements on the cost function for reconfigurable rovers and their landing to reality.** An important open line of research is the analysis on the accuracy of the proposed dynamic locomotion models and the real rover prototype. Wheel-walking and Normal-driving were modelled via simulation tools based on the characteristics of a real robotic platform named ExoTeR. Nevertheless, it still remains to perform an experiment on different kinds of terrain with this

platform to check if the proposed terramechanic models are suitable in reality. Besides, it would be interesting to further study the characterization of the terrain using the specific resistance ρ_{ij} , the slip ratio σ_{ij} and maybe even including other functions such as the drift angle or the sinkage. Moreover, it would be possible to not make a rigid separation between both locomotion modes, but rather consider them as two extremes of hybrid locomotion. In fact, during a stride of the Wheel-walking both wheels could roll at different rates instead of keeping one of them stationary. By controlling the difference in the rotational speeds of the wheels some kind of turning manoeuvre could be obtained as result. The extension of the model to different angles of roll and pitch would be of interest to have a more accurate representation in the anisotropic path planning. Besides, acceleration and deceleration would be of interest to consider, including the moments of inertia of the vehicle. A more detailed analysis of different values of speed would be of interest as well. The consideration of the cost of turning would be also desirable, especially in cases where the robot drives using skid-steering locomotion.

- **Improvements in the DyMu global-local path planning strategy.** It would be of great interest in the future to make a rover estimate terrain parameters locally, using exteroceptive sensors for instance. Besides, the local information could be compared with the global to improve the initial estimations, perhaps using machine learning algorithms. As future work, the Local Path Repairing (LPR) could be improved to not just find the shortest path (or at least an approximation given the heuristics) but also have the possibility to find the one that entails minimal energy consumption. Besides, there are other algorithms used to solve the path planning at a local scale, e.g. the Reactive Computing algorithms. It would be interesting to investigate how to integrate these algorithms into the DyMu multi-layered grid. Another improvement would be the consideration of mobile obstacles, in order to handle dynamic environments with people moving. An example of this, in irregular terrain, would be a disaster scenario with first responders on the field. Finally, the Global Path Planning (GPP) could be carried out by an anisotropic planner such as OUM. In this way, the local planner would need to use heuristics that not only consider the total cost but also the characteristic directions of the global solution.
- **Improvements in the use of CAMIS.** As seen during the simulation experiments, high anisotropy starts to introduce errors in the solution. A solution to this is to refine the grid, but increasing the number of nodes would exponentially increase the computational load of the planner. For this reason, the use of adaptive grids that depend on anisotropy would be an interesting solution to explore. Since this anisotropy arises due to the terrain features, this grid would then adapt to the shape and composition of the terrain. Other algorithms such as the FSM have been demonstrated to work with anisotropic

functions as well (Qian et al., 2007). Comparing their performance with OUM and bi-OUM would be interesting. A future upgrade to CAMIS would be the consideration of the risk of overturning in the longitudinal axis, similarly as in the lateral axis as done with the weight roll function and as was addressed by Shum et al. (2015). Instead of using the same slip model for the ascent and the descent, a more refined model could be used instead, addressing the particular differences between ascending and descending (L. Ding et al., 2013). Moreover, experiments involving anisotropic planning and obstacles would be of interest. PDE solving planners produce degenerated solutions when the cost map presents discontinuities. This is solved in the eikonal case by setting repulsive potential fields around obstacles. A similar approach would be of interest for the general case using the HJB equation. In an analogous way, (Y. Liu and Bucknall, 2016) proposed the edition of the cost map to tackle the initial and final orientations of the path while using FMM. This could be adapted to anisotropic planners like OUM.

Bibliography

- Ajanović, Zlatan, Michael Stolz, and Martin Horn (2018). "Energy-efficient driving in dynamic environment: Globally optimal MPC-like motion planning framework". In: *Advanced Microsystems for Automotive Applications 2017*. Springer, pp. 111–122.
- Alajlan, Maram, Anis Koubaa, Imen Chaari, Hachemi Bennaceur, and Adel Ammar (2013). "Global path planning for mobile robots in large-scale grid environments using genetic algorithms". In: *2013 International Conference on Individual and Collective Behaviors in Robotics (ICBR)*. IEEE, pp. 1–8.
- Alenezi, Mohammad R, Abdullah M Almeshal, et al. (2018). "Optimal Path Planning for a Remote Sensing Unmanned Ground Vehicle in a Hazardous Indoor Environment". In: *Intelligent Control and Automation 9.04*, p. 147.
- Algfoor, Zeyad Abd, Mohd Shahrizal Sunar, and Hoshang Kolivand (2015). "A comprehensive study on pathfinding techniques for robotics and video games". In: *International Journal of Computer Games Technology 2015*, p. 7.
- Arslan, Oktay and Panagiotis Tsiotras (2013). "Use of relaxation methods in sampling-based algorithms for optimal motion planning". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2421–2428.
- Arvidson, Raymond E, James W Ashley, JF Bell, Matthew Chojnacki, Joanna Cohen, TE Economou, William H Farrand, R Ferguson, Iris Fleischer, Paul Geissler, et al. (2011). "Opportunity Mars Rover mission: Overview and selected results from Purgatory ripple to traverses to Endeavour crater". In: *Journal of Geophysical Research: Planets* 116.E7.
- Arvidson, Raymond E, JF Bell, P Bellutta, Nathalie A Cabrol, JG Catalano, J Cohen, Larry S Crumpler, DJ Des Marais, TA Estlin, WH Farrand, et al. (2010). "Spirit Mars Rover Mission: Overview and selected results from the northern Home Plate Winter Haven to the side of Scamander crater". In: *Journal of Geophysical Research: Planets* 115.E7.
- Azkarate, Martin, Levin Gerdes, Tim Wiese, Martin Zwick, Marco Pagnamenta, Javier Hidalgo Carrio, Pantelis Poulakis, and Carlos Perez-del-Pulgar (2022). "Design, Testing, and Evolution of Mars Rover Testbeds: European Space Agency Planetary Exploration". In: *IEEE Robotics & Automation Magazine*.

- Azkarate, Martin, Martin Zwick, Javier Hidalgo-Carrio, Robin Nelen, Tim Wiese, Pantelis Poulakis, Luc Joudrier, and Gianfranco Visentin (2015). "First Experimental investigations on Wheel-Walking for improving Triple-Bogie rover locomotion performances". In: *13th Symposium on Advanced Space Technologies in Robotics and Automation*. ESA, pp. 1–6.
- Bajracharya, Max, Mark W Maimone, and Daniel Helmick (2008). "Autonomy for mars rovers: Past, present, and future". In: *Computer* 41.12.
- Bak, Stanley, Joyce McLaughlin, and Daniel Renzi (2010). "Some improvements for the fast sweeping method". In: *SIAM Journal on Scientific Computing* 32.5, pp. 2853–2874.
- Barbasiewicz, Adrianna, Tadeusz Widurski, and Karol Daliga (2018). "The analysis of the accuracy of spatial models using photogrammetric software: Agisoft Photoscan and Pix4D". In: *E3S Web of Conferences*. Vol. 26. EDP Sciences, p. 00012.
- Barraquand, Jerome, Bruno Langlois, and J-C Latombe (1992). "Numerical potential field techniques for robot path planning". In: *IEEE transactions on systems, man, and cybernetics* 22.2, pp. 224–241.
- Bellman, Richard (1966). "Dynamic programming". In: *Science* 153.3731, pp. 34–37.
- Benamar, Faiz and Christophe Grand (2013). "Quasi-static motion simulation and slip prediction of articulated planetary rovers using a kinematic approach". In: *Journal of Mechanisms and Robotics* 5.2, p. 021002.
- Bergman, Kristoffer, Oskar Ljungqvist, and Daniel Axehill (2020). "Improved path planning by tightly combining lattice-based path planning and optimal control". In: *IEEE Transactions on Intelligent Vehicles*.
- Biesiadecki, Jeffrey J and Mark W Maimone (2006). "The mars exploration rover surface mobility flight software driving ambition". In: *2006 IEEE Aerospace Conference*. IEEE, 15–pp.
- Birk, Andreas and Stefano Carpin (2006). "Rescue robotics—a crucial milestone on the road to autonomous systems". In: *Advanced Robotics* 20.5, pp. 595–605.
- Blum, Tamir, William Jones, and Kazuya Yoshida (2020). "PPMC training algorithm: A deep learning based path planner and motion controller". In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*. IEEE, pp. 193–198.
- Bogue, Robert (2019). "Disaster relief, and search and rescue robots: the way forward". In: *Industrial Robot: the international journal of robotics research and application*.
- Borenstein, Johann, Yoram Koren, et al. (1991). "The vector field histogram-fast obstacle avoidance for mobile robots". In: *IEEE transactions on robotics and automation* 7.3, pp. 278–288.
- Bresina, John L, Ari K Jónsson, Paul H Morris, and Kanna Rajan (2005). "Activity Planning for the Mars Exploration Rovers." In: *ICAPS*, pp. 40–49.

- Brooks, Christopher A and Karl Iagnemma (2012). "Self-supervised terrain classification for planetary surface exploration rovers". In: *Journal of Field Robotics* 29.3, pp. 445–468.
- Brunner, Michael, Torsten Fiolka, Dirk Schulz, and Christopher M Schlick (2015). "Design and comparative evaluation of an iterative contact point estimation method for static stability estimation of mobile actively reconfigurable robots". In: *Robotics and Autonomous Systems* 63, pp. 89–107.
- Campbell, Sean, Niall O'Mahony, Anderson Carvalho, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh (2020). "Path planning techniques for mobile robots a review". In: *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*. IEEE, pp. 12–16.
- Campbell, Stefan Forrest (2007). "Steering control of an autonomous ground vehicle with application to the DARPA urban challenge". PhD thesis. Massachusetts Institute of Technology.
- Canudas-de-Wit, Carlos, Panagiotis Tsotras, Efstathios Velenis, Michel Basset, and Gerard Gissinger (2003). "Dynamic friction models for road/tire longitudinal interaction". In: *Vehicle System Dynamics* 39.3, pp. 189–226.
- Cao, Jingang et al. (2016). "Robot global path planning based on an improved ant colony algorithm". In: *Journal of Computer and Communications* 4.02, p. 11.
- Carsten, Joseph, Arturo Rankin, Dave Ferguson, and Anthony Stentz (2007). "Global path planning on board the mars exploration rovers". In: *2007 IEEE Aerospace Conference*. IEEE, pp. 1–11.
- Carsten, Joseph, Arturo Rankin, Dave Ferguson, and Anthony Stentz (2009). "Global planning on the Mars Exploration Rovers: Software integration and surface testing". In: *Journal of Field Robotics* 26.4, pp. 337–357.
- Chiang, Chia Hsun, Po Jui Chiang, Jerry Chien-Chih Fei, and Jin Sin Liu (2007). "A comparative study of implementing Fast Marching Method and A* SEARCH for mobile robot path planning in grid environment: Effect of map resolution". In: *2007 IEEE Workshop on Advanced Robotics and Its Social Impacts*. IEEE, pp. 1–6.
- Choi, Sunglok, Jaehyun Park, Eulgyoon Lim, and Wonpil Yu (2012). "Global path planning on uneven elevation maps". In: *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, pp. 49–54.
- Cong, Yee Zi and SG Ponnambalam (2009). "Mobile robot path planning using ant colony optimization". In: *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, pp. 851–856.
- Costa, Márcia M and Manuel F Silva (2019). "A survey on path planning algorithms for mobile robots". In: *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, pp. 1–7.
- Creager, C, S Moreland, K Skonieczny, K Johnson, V Asnani, and R Gilligan (2012). "Benefit of" Push-Pull" Locomotion for Planetary Rover Mobility". In: *Earth and Space 2012: Engineering, Science, Construction, and Operations in Challenging Environments*, pp. 11–20.

- Creager, Colin, Kyle Johnson, Mark Plant, Scott Moreland, and Krzysztof Skonieczny (2015). "Push-pull locomotion for vehicle extrication". In: *Journal of Terramechanics* 57, pp. 71–80.
- Daniel, Kenny, Alex Nash, Sven Koenig, and Ariel Felner (2010). "Theta*: Any-angle path planning on grids". In: *Journal of Artificial Intelligence Research* 39, pp. 533–579.
- Dao, Thi-Kien, Tien-Szu Pan, and Jeng-Shyang Pan (2016). "A multi-objective optimal mobile robot path planning based on whale optimization algorithm". In: *2016 IEEE 13th International Conference on Signal Processing (ICSP)*. IEEE, pp. 337–342.
- De Canete, Javier Fernandez, Cipriano Galindo, and Inmaculada Garcia-Moral (2011). *System Engineering and Automation: An Interactive Educational Approach*. Springer Science & Business Media.
- Delmerico, Jeffrey, Stefano Mintchev, Alessandro Giusti, Boris Gromov, Kamilo Melo, Tomislav Horvat, Cesar Cadena, Marco Hutter, Auke Ijspeert, Dario Floreano, et al. (2019). "The current state and future outlook of rescue robotics". In: *Journal of Field Robotics* 36.7, pp. 1171–1191.
- Detrixhe, Miles, Frédéric Gibou, and Chohong Min (2013). "A parallel fast sweeping method for the Eikonal equation". In: *Journal of Computational Physics* 237, pp. 46–55.
- Dijkstra, Edsger W (1959). "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1, pp. 269–271.
- Ding, Liang, Haibo Gao, Zongquan Deng, Junlong Guo, and Guangjun Liu (2013). "Longitudinal slip versus skid of planetary rovers' wheels traversing on deformable slopes". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2842–2848.
- Doğan, L and U Yüzgeç (2018). "Robot Path Planning using Gray Wolf Optimizer". In: *Proceedings-International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES'18)*.
- Dolgov, Dmitri, Sebastian Thrun, Michael Montemerlo, and James Diebel (2010). "Path planning for autonomous vehicles in unknown semi-structured environments". In: *The international journal of robotics research* 29.5, pp. 485–501.
- Dunlap, Damion D, Charmane V Caldwell, and Emmanuel G Collins (2010). "Non-linear model predictive control using sampling and goal-directed optimization". In: *2010 IEEE International Conference on Control Applications*. IEEE, pp. 1349–1356.
- Effati, Meysam, Jean-Sebastien Fiset, and Krzysztof Skonieczny (2020). "Considering slip-track for energy-efficient paths of skid-steer rovers". In: *Journal of Intelligent & Robotic Systems* 100.1, pp. 335–348.
- Elbanhawi, Mohamed and Milan Simic (2014). "Sampling-based robot motion planning: A review". In: *Ieee access* 2, pp. 56–77.

- Elmi, Zahra and Mehmet Önder Efe (2018). "Multi-objective grasshopper optimization algorithm for robot path planning in static environments". In: *2018 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, pp. 244–249.
- Elmi, Zahra and Mehmet Önder Efe (2020). "Online path planning of mobile robot using grasshopper algorithm in a dynamic and unknown environment". In: *Journal of Experimental & Theoretical Artificial Intelligence*, pp. 1–19.
- Engedy, István and Gábor Horváth (2010). "Artificial neural network based local motion planning of a wheeled mobile robot". In: *2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE, pp. 213–218.
- Fankhauser, Péter and Marco Hutter (2018). "ANYmal: a unique quadruped robot conquering harsh environments". In: *Research Features* 126, pp. 54–57.
- Fausto, Fernando, Adolfo Reyna-Orta, Erik Cuevas, Ángel G Andrade, and Marco Perez-Cisneros (2020). "From ants to whales: metaheuristics for all tastes". In: *Artificial Intelligence Review* 53.1, pp. 753–810.
- Ferguson, Dave and Anthony Stentz (2005). "The Field D* algorithm for improved path planning and replanning in uniform and non-uniform cost environments". In: *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-19*.
- Festa, Adriano, Roberto Guglielmi, Christopher Hermosilla, Athena Picarelli, Smita Sahu, Achille Sassi, and Francisco J. Silva (2017). "Hamilton–Jacobi–Bellman Equations". In: *Optimal Control: Novel Directions and Applications*. Ed. by Daniela Tonon, Maria Soledad Aronna, and Dante Kalise. Cham: Springer International Publishing, pp. 127–261. ISBN: 978-3-319-60771-9. DOI: 10.1007/978-3-319-60771-9_2. URL: https://doi.org/10.1007/978-3-319-60771-9_2.
- Fiorini, Paolo and Zvi Shiller (1998). "Motion planning in dynamic environments using velocity obstacles". In: *The International Journal of Robotics Research* 17.7, pp. 760–772.
- Fox, Dieter, Wolfram Burgard, and Sebastian Thrun (1997). "The dynamic window approach to collision avoidance". In: *IEEE Robotics & Automation Magazine* 4.1, pp. 23–33.
- Gammell, Jonathan D, Siddhartha S Srinivasa, and Timothy D Barfoot (2014). "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2997–3004.
- Gammell, Jonathan D, Siddhartha S Srinivasa, and Timothy D Barfoot (2015). "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs". In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 3067–3074.
- Ganganath, Nuwan, Chi-Tsun Cheng, and K Tse Chi (2015). "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains". In: *IEEE transactions on industrial informatics* 11.3, pp. 601–611.

- Ganganath, Nuwan, Chi-Tsun Cheng, Tyrone Fernando, Herbert HC Iu, and K Tse Chi (2018). "Shortest path planning for energy-constrained mobile platforms navigating on uneven terrains". In: *IEEE Transactions on Industrial Informatics* 14.9, pp. 4264–4272.
- Gao, Yang and Steve Chien (2017). "Review on space robotics: Toward top-level science through space exploration". In: *Science Robotics* 2.7.
- Garrido, Santiago, David Álvarez, and Luis Moreno (2016). "Path Planning for Mars Rovers Using the Fast Marching Method". In: *Robot 2015: Second Iberian Robotics Conference*. Springer, pp. 93–105.
- Garrido, Santiago, Mariéa Malfaz, and Dolores Blanco (2013). "Application of the fast marching method for outdoor motion planning in robotics". In: *Robotics and Autonomous Systems* 61.2, pp. 106–114.
- Ge, Shuzhi Sam and Yun J Cui (2002). "Dynamic motion planning for mobile robots using potential field method". In: *Autonomous robots* 13.3, pp. 207–222.
- Gerdes, Levin, Martin Azkarate, José Ricardo Sánchez-Ibáñez, Luc Joudrier, and Carlos Jesús Perez-del-Pulgar (2020). "Efficient autonomous navigation for planetary rovers with limited resources". In: *Journal of Field Robotics* 37.7, pp. 1153–1170.
- Ghosh, Saradindu, Pratap K Panigrahi, and Dayal R Parhi (2017). "Analysis of FPA and BA meta-heuristic controllers for optimal path planning of mobile robot in cluttered environment". In: *IET Science, Measurement & Technology* 11.7, pp. 817–828.
- Gillespie, Thomas D (1992). *Fundamentals of vehicle dynamics*. Vol. 400. Society of automotive engineers Warrendale, PA.
- Gómez, Javier V, David Álvarez, Santiago Garrido, and Luis Moreno (2019). "Fast methods for eikonal equations: an experimental survey". In: *IEEE Access* 7, pp. 39005–39029.
- González, David, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi (2015). "A review of motion planning techniques for automated vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 17.4, pp. 1135–1145.
- Govindaraj, Shashank, Jeremi Gancet, Diego Urbina, Wiebke Brinkmann, Nabil Aouf, Simon Lacroix, Mateusz Wolski, Francisco Colmenero, Michael Walshe, Cristina Ortega, et al. (2019). "PRO-ACT: Planetary Robots Deployed for Assembly and Construction of Future Lunar ISRU and Supporting Infrastructures". In: *Proceedings of the 15th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA-2019)*.
- Groves, Keir, Emili Hernandez, Andrew West, Thomas Wright, and Barry Lennox (2021). "Robotic Exploration of an Unknown Nuclear Environment Using Radiation Informed Autonomous Navigation". In: *Robotics* 10.2, p. 78.
- Gruning, Veronica, Jesse Pentzer, Sean Brennan, and Karl Reichard (2020). "Energy-Aware Path Planning for Skid-Steer Robots Operating on Hilly Terrain". In: *2020 American Control Conference (ACC)*. IEEE, pp. 2094–2099.

- Han, Woong-Gie, Seung-Min Baek, and Tae-Yong Kuc (1997). "Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots". In: *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*. Vol. 3. IEEE, pp. 2747–2751.
- Hart, Peter E, Nils J Nilsson, and Bertram Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.
- Henkel, Christian, Alexander Bubeck, and Weiliang Xu (2016). "Energy efficient dynamic window approach for local path planning in mobile service robotics". In: *IFAC-PapersOnLine* 49.15, pp. 32–37.
- Hewitt, Robert A, Evangelos Boukas, Martin Azkarate, Marco Pagnamenta, Joshua A Marshall, Antonios Gasteratos, and Gianfranco Visentin (2018). "The katwijk beach planetary rover dataset". In: *The International Journal of Robotics Research* 37.1, pp. 3–12.
- Hines, Thomas, Kazys Stepanas, Fletcher Talbot, Inkyu Sa, Jake Lewis, Emili Hernandez, Navinda Kottege, and Nicolas Hudson (2021). "Virtual Surfaces and Attitude Aware Planning and Behaviours for Negative Obstacle Navigation". In: *IEEE Robotics and Automation Letters* 6.2, pp. 4048–4055.
- Hossain, Md Arafat and Israt Ferdous (2015). "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique". In: *Robotics and Autonomous Systems* 64, pp. 137–141.
- Howard, Thomas M and Alonzo Kelly (2007). "Optimal rough terrain trajectory generation for wheeled mobile robots". In: *The International Journal of Robotics Research* 26.2, pp. 141–166.
- Huang, Han-Pang and Shu-Yun Chung (2004). "Dynamic visibility graph for path planning". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE, pp. 2813–2818.
- Hutter, Marco, Christian Gehring, Andreas Lauber, Fabian Gunther, Carmine Dario Bellicoso, Vassilios Tsounis, Péter Fankhauser, Remo Diethelm, Samuel Bachmann, Michael Blösch, et al. (2017). "Anymal-toward legged robots for harsh environments". In: *Advanced Robotics* 31.17, pp. 918–931.
- Ichter, Brian, Edward Schmerling, and Marco Pavone (2017). "Group Marching Tree: Sampling-based approximately optimal motion planning on GPUs". In: *2017 First IEEE International Conference on Robotic Computing (IRC)*. IEEE, pp. 219–226.
- Injarapu, Anantha Sai Hari Haran V and Suresh Kumar Gawre (2017). "A survey of autonomous mobile robot path planning approaches". In: *2017 International conference on recent innovations in signal processing and embedded systems (RISE)*. IEEE, pp. 624–628.
- Inotsume, H., T. Kubota, and D. Wettergreen (2020). "Robust Path Planning for Slope Traversing under Uncertainty in Slip Prediction". In: *IEEE Robotics and Automation Letters*, pp. 1–1. ISSN: 2377-3774. DOI: 10.1109/LRA.2020.2975756.

- Inotsume, Hiroaki, Colin Creager, David Wettergreen, and WRL Whittaker (2016). "Finding routes for efficient and successful slope ascent for exploration rovers". In: *The International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*.
- Ishigami, Genya, Keiji Nagatani, and Kazuya Yoshida (2007). "Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2361–2366.
- Ishigami, Genya, Keiji Nagatani, and Kazuya Yoshida (2011). "Path planning and evaluation for planetary rovers based on dynamic mobility index". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 601–606.
- Jeong, Won-Ki and Ross T Whitaker (2008). "A fast iterative method for eikonal equations". In: *SIAM Journal on Scientific Computing* 30.5, pp. 2512–2534.
- Jorge, Vitor AM, Roger Granada, Renan G Maidana, Darlan A Jurak, Guilherme Heck, Alvaro PF Negreiros, Davi H Dos Santos, Luiz MG Gonçalves, and Alexandre M Amory (2019). "A survey on unmanned surface vehicles for disaster robotics: Main challenges and directions". In: *Sensors* 19.3, p. 702.
- Joshi, Maulin M and Mukesh A Zaveri (2011). "Reactive navigation of autonomous mobile robot using neuro-fuzzy system". In: *International Journal of Robotics and Automation (IJRA)* 2.3, p. 128.
- Joyeux, Sylvain, Jakob Schwendner, Frank Kirchner, Ajish Babu, Felix Grimminger, Janosch Machowinski, Patrick Paranhos, and Christopher Gaudig (2011). "Intelligent Mobility". In: *KI-Künstliche Intelligenz* 25.2, pp. 133–139.
- Kalmár-Nagy, Tamás, Raffaello D'Andrea, and Pritam Ganguly (2004). "Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle". In: *Robotics and Autonomous Systems* 46.1, pp. 47–64.
- Kao, Chiu-Yen, Stanley Osher, and Yen-Hsi Tsai (2005). "Fast sweeping methods for static Hamilton–Jacobi equations". In: *SIAM journal on numerical analysis* 42.6, pp. 2612–2632.
- Kaplan, Adam, Nathaniel Kingry, Paul Uhing, and Ran Dai (2016). "Time-optimal path planning with power schedules for a solar-powered ground robot". In: *IEEE Transactions on Automation Science and Engineering* 14.2, pp. 1235–1244.
- Karaman, Sertac and Emilio Frazzoli (2011). "Sampling-based algorithms for optimal motion planning". In: *The international journal of robotics research* 30.7, pp. 846–894.
- Kavraki, Lydia E, Petr Svestka, J-C Latombe, and Mark H Overmars (1996). "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4, pp. 566–580.
- Khatib, Maher, H Jaouni, Raja Chatila, and Jean-Paul Laumond (1997). "Dynamic path modification for car-like nonholonomic mobile robots". In: *Proceedings of international conference on robotics and automation*. Vol. 4. IEEE, pp. 2920–2925.

- Khatib, Oussama (1985). "Real-time obstacle avoidance for manipulators and mobile robots". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE, pp. 500–505.
- Kimmel, Ron and James A Sethian (2001). "Optimal algorithm for shape from shading and path planning". In: *Journal of Mathematical Imaging and Vision* 14.3, pp. 237–244.
- Klamt, Tobias, Diego Rodriguez, Max Schwarz, Christian Lenz, Dmytro Pavlichenko, David Droschel, and Sven Behnke (2018). "Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1–8.
- Koenig, Sven and Maxim Likhachev (2002a). "D* Lite". In: *Aaai/iaai* 15.
- Koenig, Sven and Maxim Likhachev (2002b). "Incremental a*". In: *Advances in neural information processing systems*, pp. 1539–1546.
- Koenig, Sven, Maxim Likhachev, and David Furcy (2004). "Lifelong planning A*". In: *Artificial Intelligence* 155.1-2, pp. 93–146.
- Kogan, Dmitriy and R Murray (2006). "Optimization-based navigation for the DARPA Grand Challenge". In: *Conference on Decision and Control (CDC)*.
- Köster, Moritz, Ezgi Kayhan, Miriam Langeloh, and Stefanie Hoehl (2020). "Making sense of the world: infant learning from a predictive processing perspective". In: *Perspectives on psychological science* 15.3, pp. 562–571.
- Kruijff-Korbayová, Ivana, Luigi Freda, Mario Gianni, Valsamis Ntouskos, Václav Hlaváč, Vladimír Kubelka, Erik Zimmermann, Hartmut Surmann, Kresimir Dulic, Wolfgang Rottner, et al. (2016). "Deployment of ground and aerial robots in earthquake-struck amatrice in Italy (brief report)". In: *2016 IEEE international symposium on safety, security, and rescue robotics (SSRR)*. IEEE, pp. 278–279.
- Krüsi, Philipp, Paul Furgale, Michael Bosse, and Roland Siegwart (2017). "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments". In: *Journal of Field Robotics* 34.5, pp. 940–984.
- Kuffner, James J and Steven M LaValle (2000). "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE, pp. 995–1001.
- Kuwata, Yoshiaki, Michael T Wolf, Dimitri Zarghitzky, and Terrance L Huntsberger (2013). "Safe maritime autonomous navigation with COLREGS, using velocity obstacles". In: *IEEE Journal of Oceanic Engineering* 39.1, pp. 110–119.
- Laine, Mickael, Chihiro Tamakoshi, Meven Touboul, John Walker, and Kazuya Yoshida (2018). "Initial design characteristics, testing and performance optimisation for a lunar exploration micro-rover prototype". In: *Advances in Astronautics Science and Technology* 1.1, pp. 111–117.
- LaValle, Steven M (2006). *Planning algorithms*. Cambridge university press.

- Lele, Ajey (2014). "Mars Missions: Past, Present and Future". In: *Mission Mars*. Springer, pp. 85–92.
- Lester, Dan F, Kip V Hodges, and Robert C Anderson (2017). "Exploration telepresence: A strategy for optimizing scientific research at remote space destinations". In: *Science Robotics* 2.7, Art–No.
- Levchenko, Igor, Shuyan Xu, Stéphane Mazouffre, Michael Keidar, and Kateryna Bazaka (2019). "Mars colonization: beyond getting there". In: *Global Challenges* 3.1, p. 1800062.
- Likhachev, Maxim and Dave Ferguson (2009). "Planning long dynamically feasible maneuvers for autonomous vehicles". In: *The International Journal of Robotics Research* 28.8, pp. 933–945.
- Likhachev, Maxim, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun (2008). "Anytime search in dynamic graphs". In: *Artificial Intelligence* 172.14, pp. 1613–1643.
- Lindsay, John B, Daniel R Newman, and Anthony Francioni (2019). "Scale-Optimized Surface Roughness for Topographic Analysis". In: *Geosciences* 9.7, p. 322.
- Liu, Yuanchang and Richard Bucknall (2016). "The angle guidance path planning algorithms for unmanned surface vehicle formations by using the fast marching method". In: *Applied Ocean Research* 59, pp. 327–344.
- Liu, Yuanchang, Wenwen Liu, Rui Song, and Richard Bucknall (2017). "Predictive navigation of unmanned surface vehicles in a dynamic maritime environment when using the fast marching method". In: *International Journal of Adaptive Control and Signal Processing* 31.4, pp. 464–488.
- Lu, Li and Dunwei Gong (2008). "Robot path planning in unknown environments using particle swarm optimization". In: *2008 Fourth International Conference on Natural Computation*. Vol. 4. IEEE, pp. 422–426.
- Lumelsky, Vladimir and Alexander Stepanov (1986). "Dynamic path planning for a mobile automaton with limited information on the environment". In: *IEEE transactions on Automatic control* 31.11, pp. 1058–1063.
- Luo, Qiang, Haibao Wang, Yan Zheng, and Jingchang He (2020). "Research on path planning of mobile robot based on improved ant colony algorithm". In: *Neural Computing and Applications* 32.6, pp. 1555–1566.
- Ma, Cedric S and Robert H Miller (2006). "MILP optimal path planning for real-time applications". In: *2006 American Control Conference*. IEEE, 6–pp.
- Mac, Thi Thoa, Cosmin Copot, Duc Trung Tran, and Robin De Keyser (2016). "Heuristic approaches in robot path planning: A survey". In: *Robotics and Autonomous Systems* 86, pp. 13–28.
- Mac, Thi Thoa, Cosmin Copot, Duc Trung Tran, and Robin De Keyser (2017). "A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization". In: *Applied Soft Computing* 59, pp. 68–76.

- Maimone, Mark W, P Chris Leger, and Jeffrey J Biesiadecki (2007). "Overview of the mars exploration rovers' autonomous mobility and vision capabilities". In: *IEEE international conference on robotics and automation (ICRA) space robotics workshop*.
- Malenkov, MI and VA Volov (2017). "Wheel-walking propulsion unit of a planetary rover with active suspension". In: *Russian Engineering Research* 37.12, pp. 1033–1040.
- Mandow, Anthony, Jorge L Martinez, Jesus Morales, Jose L Blanco, Alfonso Garcia-Cerezo, and Javier Gonzalez (2007). "Experimental kinematics for wheeled skid-steer mobile robots". In: *2007 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, pp. 1222–1227.
- Mandow, Anthony, Javier Serón, Francisco Pastor, and Alfonso García-Cerezo (2020). "Experimental Validation of a Robotic Stretcher for Casualty Evacuation in a Man-Made Disaster Exercise". In: *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, pp. 241–245.
- Marin, Leonardo, Marina Vallés, Angel Soriano, Angel Valera, and Pedro Albertos (2013). "Event-based localization in ackermann steering limited resource mobile robots". In: *IEEE/ASME Transactions on Mechatronics* 19.4, pp. 1171–1182.
- Martinez, Jorge L, Anthony Mandow, Antonio Reina, Tomás J Cantador, Jesus Morales, and Alfonso Garcia-Cerezo (2013). "Navigability analysis of natural terrains with fuzzy elevation maps from ground-based 3D range scans". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1576–1581.
- Miki, Takahiro, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter (2022). "Learning robust perceptive locomotion for quadrupedal robots in the wild". In: *Science Robotics* 7.62.
- Mirjalili, Seyedali and Jin Song Dong (2020). "Introduction to nature-inspired algorithms". In: *Nature-Inspired Optimizers*. Springer, pp. 1–5.
- Mirjalili, Seyedali and Andrew Lewis (2016). "The whale optimization algorithm". In: *Advances in engineering software* 95, pp. 51–67.
- Miró, Jaime Valls, Gaultier Dumonteil, Christoph Beck, and Gamini Dissanayake (2010). "A kynodynamic metric to plan stable paths over uneven terrain". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 294–299.
- Mishkin, Andrew H, Jack C Morrison, Tam T Nguyen, Henry W Stone, Brian K Cooper, and Brian H Wilcox (1998). "Experiences with operations and autonomy of the mars pathfinder microrover". In: *1998 IEEE aerospace conference proceedings (Cat. No. 98TH8339)*. Vol. 2. IEEE, pp. 337–351.
- Mohanty, Prases K and Dayal R Parhi (2014). "A new intelligent motion planning for mobile robot navigation using multiple adaptive neuro-fuzzy inference system". In: *Applied Mathematics & Information Sciences* 8.5, p. 2527.
- Mohanty, Prases Kumar and Dayal R Parhi (2013). "Cuckoo search algorithm for the mobile robot navigation". In: *International Conference on Swarm, Evolutionary, and Memetic Computing*. Springer, pp. 527–536.

- Morales, Jesus, Jorge L Martinez, Anthony Mandow, Alfonso J Garcíea-Cerezo, and Salvador Pedraza (2009). "Power consumption modeling of skid-steer tracked mobile robots on rigid terrain". In: *IEEE Transactions on Robotics* 25.5, pp. 1098–1108.
- Morales, Jesús, Jorge L Martíénez, Anthony Mandow, Alejandro Pequeño-Boter, and Alfonso Garcíea-Cerezo (2010). "Simplified power consumption modeling and identification for wheeled skid-steer robotic vehicles on hard horizontal ground". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4769–4774.
- Moreland, Scott, Krzysztof Skonieczny, David Wettergreen, Vivake Asnani, Colin Creager, and Heather Oravec (2011). "Inching locomotion for planetary rover mobility". In: *2011 Aerospace Conference*. IEEE, pp. 1–6.
- Muirhead, Brian K and Ashley Karp (2019). "Mars Sample Return Lander Mission Concepts". In: *2019 IEEE Aerospace Conference*. IEEE, pp. 1–9.
- Muñoz, Pablo and María D R-Moreno (2012). "S-Theta: low steering path-planning algorithm". In: *Research and Development in Intelligent Systems XXIX*. Springer, pp. 109–121.
- Muñoz, Pablo, María D R-Moreno, and Bonifacio Castaño (2016). "3Dana: Path Planning on 3D Surfaces". In: *Research and Development in Intelligent Systems XXXIII: Incorporating Applications and Innovations in Intelligent Systems XXIV* 33. Springer, pp. 177–191.
- Muñoz, Pablo, María D R-Moreno, and Bonifacio Castaño (2017). "3Dana: a path planning algorithm for surface robotics". In: *Engineering Applications of Artificial Intelligence* 60, pp. 175–192.
- Murphy, Robin R (2004). "Trial by fire [rescue robots]". In: *IEEE Robotics & Automation Magazine* 11.3, pp. 50–61.
- Murphy, Robin R (2012). "A decade of rescue robots". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5448–5449.
- Murphy, Robin R, Satoshi Tadokoro, and Alexander Kleiner (2016). "Disaster robotics". In: *Springer Handbook of Robotics*. Springer, pp. 1577–1604.
- Muthukumaran et al. (2019). "Optimal path planning for an autonomous mobile robot using dragonfly algorithm". In: *International Journal of Simulation Modelling* 18.3, pp. 397–407.
- Nagatani, Keiji, Seiga Kiribayashi, Yoshito Okada, Kazuki Otake, Kazuya Yoshida, Satoshi Tadokoro, Takeshi Nishimura, Tomoaki Yoshida, Eiji Koyanagi, Mineo Fukushima, et al. (2013). "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots". In: *Journal of Field Robotics* 30.1, pp. 44–63.
- Nash, Alex, Kenny Daniel, Sven Koenig, and Ariel Felner (2007). "Theta*: any-angle path planning on grids". In: *AAAI*. Vol. 7, pp. 1177–1183.
- Nash, Alex and Sven Koenig (2013). "Any-angle path planning". In: *AI Magazine* 34.4, pp. 85–107.

- Nash, Alex, Sven Koenig, and Maxim Likhachev (2009). "Incremental Phi*: Incremental any-angle path planning on grids". In: *Twenty-First International Joint Conference on Artificial Intelligence*.
- Nash, Alex, Sven Koenig, and Craig Tovey (2010). "Lazy Theta*: Any-angle path planning and path length analysis in 3D". In: *Twenty-Fourth AAAI Conference on Artificial Intelligence*.
- Niksirat, Parna, Adriana Daca, and Krzysztof Skonieczny (2020). "The effects of reduced-gravity on planetary rover mobility". In: *The International Journal of Robotics Research* 39.7, pp. 797–811.
- Noreen, Iram, Amna Khan, and Zulfiqar Habib (2016). "Optimal path planning using RRT* based approaches: a survey and future directions". In: *Int. J. Adv. Comput. Sci. Appl* 7.11, pp. 97–107.
- Norouzi, Mohammad, Jaime Valls Miro, and Gamini Dissanayake (2017). "Planning stable and efficient paths for reconfigurable robots on uneven terrain". In: *Journal of Intelligent & Robotic Systems* 87.2, pp. 291–312.
- Ono, Masahiro, Thoams J Fuchs, Amanda Steffy, Mark Maimone, and Jeng Yen (2015). "Risk-aware planetary rover operation: Autonomous terrain classification and path planning". In: *2015 IEEE aerospace conference*. IEEE, pp. 1–10.
- Otsu, Kyohei, Guillaume Matheron, Sourish Ghosh, Olivier Toupet, and Masahiro Ono (2020). "Fast approximate clearance evaluation for rovers with articulated suspension systems". In: *Journal of Field Robotics* 37.5, pp. 768–785.
- Páez, R, C Torrecillas, I Barbero, and M Berrocoso (2017). "Regional positioning services as economic and construction activity indicators: the case study of Andalusian Positioning Network (Southern Spain)". In: *Geocarto international* 32.1, pp. 44–58.
- Papadakis, Panagiotis (2013). "Terrain traversability analysis methods for unmanned ground vehicles: A survey". In: *Engineering Applications of Artificial Intelligence* 26.4, pp. 1373–1385.
- Papadopoulos, Evangelos and Michael Misailidis (2007). "On differential drive robot odometry with application to path planning". In: *2007 European Control Conference (ECC)*. IEEE, pp. 5492–5499.
- Park, Byungjae, Jinwoo Choi, and Wan Kyun Chung (2018). "Incremental hierarchical roadmap construction for efficient path planning". In: *ETRI Journal* 40.4, pp. 458–470.
- Patel, Nildeep, Richard Slade, and Jim Clemmet (2010). "The ExoMars rover locomotion subsystem". In: *Journal of Terramechanics* 47.4, pp. 227–242.
- Patle, BK, Anish Pandey, DRK Parhi, A Jagadeesh, et al. (2019). "A review: On path planning strategies for navigation of mobile robot". In: *Defence Technology*.
- Paz-Delgado, Gonzalo Jesús, Martin Azkarate, José Ricardo Sánchez-Ibáñez, Carlos Jesús Pérez-del-Pulgar, Levin Gerdes, and Alfonso J Garcíea-Cerezo (2020).

- “Improving Autonomous Rover Guidance in Round-Trip Missions Using a Dynamic Cost Map”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 7014–7019.
- Pérez-del-Pulgar, Carlos Jesús, JR Sánchez, AJ Sánchez, Martin Azkarate, and Gianfranco Visentin (2017). “Path planning for reconfigurable rovers in planetary exploration”. In: *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, pp. 1453–1458.
- Pérez, Lucía Hilario, Marta Covadonga Mora Aguilar, Nicolás Montés Sánchez, and Antonio Falcó Montesinos (2018). “Path Planning Based on Parametric Curves”. In: *Advanced Path Planning for Mobile Entities*, p. 125.
- Petres, Clement, Yan Pailhas, Yvan Petillot, and Dave Lane (2005). “Underwater path planing using fast marching algorithms”. In: *Oceans 2005-Europe*. Vol. 2. IEEE, pp. 814–819.
- Philippesen, Roland (2007). “E* Interpolated Graph Replanner”. In: *Workshop Proceedings on Algorithmic Motion Planning for Autonomous Robots in Challenging Environments, held in conjunction with the IEEE International Conference on Intelligent Robots and Systems (IROS)*. Vol. 69.
- Philippesen, Roland, Sascha Kolski, Kristijan Macek, and Björn Jensen (2008). “Mobile robot planning in dynamic environments and on growable costmaps”. In: *Workshop on Planning with Cost Maps at the IEEE Intl. Conf. on Robotics and Automation*.
- Plonski, Patrick A, Pratap Tokekar, and Volkan Isler (2013). “Energy-efficient path planning for solar-powered mobile robots”. In: *Journal of Field Robotics* 30.4, pp. 583–601.
- Qian, Jianliang, Yong-Tao Zhang, and Hong-Kai Zhao (2007). “A fast sweeping method for static convex Hamilton–Jacobi equations”. In: *Journal of Scientific Computing* 31.1, pp. 237–271.
- Quinlan, Sean and Oussama Khatib (1993). “Elastic bands: Connecting path planning and control”. In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, pp. 802–807.
- Raghavan, Vignesh Sushrutha, Dimitrios Kanoulas, Arturo Laurenzi, Darwin G Caldwell, and Nikos G Tsagarakis (2019). “Variable configuration planner for legged-rolling obstacle negotiation locomotion: Application on the centauro robot”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4738–4745.
- Raja, Purushothaman and Sivagurunathan Pugazhenthithi (2012). “Optimal path planning of mobile robots: A review”. In: *International journal of physical sciences* 7.9, pp. 1314–1320.
- Raja, Rekha, Ashish Dutta, and KS Venkatesh (2015). “New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover”. In: *Robotics and Autonomous Systems* 72, pp. 295–306.

- Ram, Ashwin, Gary Boone, Ronald Arkin, and Michael Pearce (1994). "Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation". In: *Adaptive behavior* 2.3, pp. 277–305.
- Ratliff, Nathan, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa (2009). "CHOMP: Gradient optimization techniques for efficient motion planning". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 489–494.
- Ravankar, Abhijeet, Ankit A Ravankar, Yukinori Kobayashi, Yohei Hoshino, and Chao-Chung Peng (2018). "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges". In: *Sensors* 18.9, p. 3170.
- Reid, William, Robert Fitch, Ali H Göktoğan, and Salah Sukkarieh (2019). "Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover". In: *Journal of Field Robotics*.
- Rohmer, Eric, Giulio Reina, and Kazuya Yoshida (2010). "Dynamic Simulation-Based Action Planner for a Reconfigurable Hybrid Leg–Wheel Planetary Exploration Rover". In: *Advanced Robotics* 24.8-9, pp. 1219–1238.
- Rohmer, Eric, Tomoaki Yoshida, Kazunori Ohno, Keiji Nagatani, Satoshi Tadokoro, and Eiji Konayagi (2010). "Quince: A collaborative mobile robotic platform for rescue robots research and development". In: *The Abstracts of the international conference on advanced mechatronics: toward evolutionary fusion of IT and mechatronics: ICAM 2010.5*. The Japan Society of Mechanical Engineers, pp. 225–230.
- Rösmann, Christoph, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram (2012). "Trajectory modification considering dynamic constraints of autonomous robots". In: *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, pp. 1–6.
- Rothrock, Brandon, Ryan Kennedy, Chris Cunningham, Jeremie Papon, Matthew Heverly, and Masahiro Ono (2016). "Spoc: Deep learning-based terrain classification for mars rover missions". In: *AIAA SPACE 2016*, p. 5539.
- Rowe, Neil C and Ron S Ross (1990). "Optimal grid-free path planning across arbitrarily-contoured terrain with anisotropic friction and gravity effects". In: *IEEE Transactions on Robotics and Automation* 6.5, pp. 540–553. DOI: 10.1109/70.62043.
- Sakayori, Go and Genya Ishigami (2017). "Energy efficient slope traversability planning for mobile robot in loose soil". In: *2017 IEEE International Conference on Mechatronics (ICM)*. IEEE, pp. 99–104.
- Sánchez-Ibáñez, José Ricardo, Carlos J Pérez-del-Pulgar, and Alfonso García-Cerezo (2021). "Path Planning for Autonomous Mobile Robots: A Review". In: *Sensors* 21.23, p. 7898.
- Sangeetha, V, Raghunathan Krishankumar, KS Ravichandran, and Samarjit Kar (2021). "Energy-efficient green ant colony optimization for path planning in dynamic 3D environments". In: *Soft Computing* 25.6, pp. 4749–4769.

- Sangeetha, Viswanathan, Raghunathan Krishankumar, Kattur Soundarapandian Ravichandran, Fausto Cavallaro, Samarjit Kar, Dragan Pamucar, and Abbas Mardani (2021). "A Fuzzy Gain-Based Dynamic Ant Colony Optimization for Path Planning in Dynamic Environments". In: *Symmetry* 13.2, p. 280.
- Saraswathi, Mbl, Gunji Bala Murali, and BBVL Deepak (2018). "Optimal path planning of mobile robot using hybrid cuckoo search-bat algorithm". In: *Procedia computer science* 133, pp. 510–517.
- Seppänen, Hannes and Kirsi Virrantaus (2015). "Shared situational awareness and information quality in disaster management". In: *Safety Science* 77, pp. 112–122.
- Sethian, James A (1999). "Fast Marching Methods". In: *SIAM review* 41.2, pp. 199–235. DOI: 10.1137/S0036144598347059.
- Sethian, James A and Alexander Vladimirsky (2003). "Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms". In: *SIAM Journal on Numerical Analysis* 41.1, pp. 325–363.
- Sethian, James Albert (1999). *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Vol. 3. Cambridge university press.
- Shi, Weiren, Kai Wang, and Simon X Yang (2009). "A fuzzy-neural network approach to multisensor integration for obstacle avoidance of a mobile robot". In: *Intelligent Automation & Soft Computing* 15.2, pp. 289–301.
- Shum, Alex, Kirsten Morris, and Amir Khajepour (2015). "Direction-dependent optimal path planning for autonomous vehicles". In: *Robotics and Autonomous Systems* 70, pp. 202–214.
- Shum, Alex, Kirsten Morris, and Amir Khajepour (2016). "Convergence rate for the ordered upwind method". In: *Journal of Scientific Computing* 68.3, pp. 889–913.
- Silvagni, Mario, Andrea Tonoli, Enrico Zenerino, and Marcello Chiaberge (2017). "Multipurpose UAV for search and rescue operations in mountain avalanche events". In: *Geomatics, Natural Hazards and Risk* 8.1, pp. 18–33.
- Šišlák, David, Premysl Volf, and Michal Pechoucek (2009). "Accelerated A* trajectory planning: Grid-based path planning comparison". In: *Proceedings of the 19th International Conference on Automated Planning & Scheduling (ICAPS)*, pp. 74–81.
- Souissi, Omar, Rabie Benatitallah, David Duvivier, AbedlHakim Artiba, Nicolas Belanger, and Pierre Feyzeau (2013). "Path planning: A 2013 survey". In: *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*. IEEE, pp. 1–8.
- Stentz, Anthony (1994). "Optimal and efficient path planning for partially-known environments". In: *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, pp. 3310–3317.
- Stentz, Anthony et al. (1995). "The focussed d* algorithm for real-time replanning". In: *IJCAI*. Vol. 95, pp. 1652–1659.
- Strub, Marlin P and Jonathan D Gammell (2020a). "Adaptively Informed Trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics". In: 2020

- IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3191–3198.
- Strub, Marlin P and Jonathan D Gammell (2020b). “Advanced BIT*(ABIT*): Sampling-based planning with advanced graph-search techniques”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 130–136.
- Sun, Huihui, Weijie Zhang, YU Runxiang, and Yujie Zhang (2021). “Motion planning for mobile Robots—focusing on deep reinforcement learning: A systematic Review”. In: *IEEE Access*.
- Sun, Wen, Jur van den Berg, and Ron Alterovitz (2016). “Stochastic extended LQR for optimization-based motion planning under uncertainty”. In: *IEEE Transactions on Automation Science and Engineering* 13.2, pp. 437–447.
- Sun, Zheng and John H Reif (2005). “On finding energy-minimizing paths on terrains”. In: *IEEE Transactions on Robotics* 21.1, pp. 102–114.
- Sutoh, Masataku, Masatsugu Otsuki, Sachiko Wakabayashi, Takeshi Hoshino, and Tatsuki Hashimoto (2015). “The right path: comprehensive path planning for lunar exploration rovers”. In: *IEEE Robotics & Automation Magazine* 22.1, pp. 22–33.
- Taghavifar, Hamid, Subhash Rakheja, and Giulio Reina (2020). “A novel optimal path-planning and following algorithm for wheeled robots on deformable terrains”. In: *Journal of Terramechanics*.
- Takei, Ryo and Richard Tsai (2013). “Optimal trajectories of curvature constrained motion in the hamilton-jacobi formulation”. In: *Journal of Scientific Computing* 54.2, pp. 622–644.
- Tang, Kit-Sang, Kim-Fung Man, Sam Kwong, and Qun He (1996). “Genetic algorithms and their applications”. In: *IEEE signal processing magazine* 13.6, pp. 22–37.
- Tharwat, Alaa, Mohamed Elhoseny, Aboul Ella Hassanien, Thomas Gabel, and Arun Kumar (2019). “Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm”. In: *Cluster Computing* 22.2, pp. 4745–4766.
- Tonon, Daniela, Maria Soledad Aronna, and Dante Kalise (2017). *Optimal control: Novel directions and applications*. Vol. 1. Springer.
- Tsai, Pei-Wei, Thi-Kien Dao, et al. (2016). “Robot path planning optimization based on multiobjective grey wolf optimizer”. In: *International Conference on Genetic and Evolutionary Computing*. Springer, pp. 166–173.
- Vadakkepat, Prahlad, Kay Chen Tan, and Wang Ming-Liang (2000). “Evolutionary artificial potential fields and their application in real time robot path planning”. In: *Proceedings of the 2000 congress on evolutionary computation*. CEC00 (Cat. No. 00TH8512). Vol. 1. IEEE, pp. 256–263.
- Vagale, Anete, Rachid Oucheikh, Robin T Bye, Ottar L Osen, and Thor I Fossen (2021). “Path planning and collision avoidance for autonomous surface vehicles I: a review”. In: *Journal of Marine Science and Technology*, pp. 1–15.

- Vago, Jorge L, Frances Westall, Andrew J Coates, Ralf Jaumann, Oleg Korablev, Valérie Ciarletti, Igor Mitrofanov, Jean-Luc Josset, Maria Cristina De Sanctis, Jean-Pierre Bibring, et al. (2017). "Habitability on early Mars and the search for biosignatures with the ExoMars Rover". In: *Astrobiology*.
- Valero-Gomez, Alberto, Javier V Gomez, Santiago Garrido, and Luis Moreno (2013). "The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories". In: *IEEE Robotics & Automation Magazine* 20.4, pp. 111–120.
- Van Den Berg, Jur, Sachin Patil, and Ron Alterovitz (2017). "Motion planning under uncertainty using differential dynamic programming in belief space". In: *Robotics Research*. Springer, pp. 473–490.
- Wakabayashi, Sachiko, Hitoshi Sato, and Shin-Ichiro Nishida (2009). "Design and mobility evaluation of tracked lunar vehicle". In: *Journal of Terramechanics* 46.3, pp. 105–114.
- Wang, Hao, Jie Duan, Maoli Wang, Jingbo Zhao, and Zhenzhen Dong (2018). "Research on robot path planning based on fuzzy neural network algorithm". In: *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, pp. 1800–1803.
- Wang, Lanfei, Jiangming Kan, Jun Guo, and Chao Wang (2019). "3D path planning for the ground robot with improved ant colony optimization". In: *Sensors* 19.4, p. 815.
- Wang, Lin-Lin and Li-Xin Pan (2020). "Research on SBMPC Algorithm for Path Planning of Rescue and Detection Robot". In: *Discrete Dynamics in Nature and Society* 2020.
- Wang, Meng et al. (2005). "Fuzzy logic based robot path planning in unknown environment". In: *2005 International Conference on Machine Learning and Cybernetics*. Vol. 2. IEEE, pp. 813–818.
- Wang, Yaobing (2021). "Space Robot Teleoperation System". In: *Space Robotics*. Singapore: Springer Singapore, pp. 227–245. DOI: 10.1007/978-981-15-4902-1_10. URL: https://doi.org/10.1007/978-981-15-4902-1_10.
- Weih Jr, Robert C and Tabitha L Mattson (2004). "Modeling slope in a geographic information system". In: *Journal of the Arkansas Academy of Science* 58.1, pp. 100–108.
- Wen, Zhi-qiang and Zi-xing Cai (2006). "Global path planning approach based on ant colony optimization algorithm". In: *Journal of Central South University of Technology* 13.6, pp. 707–712.
- Wiese, Tim (2017). "3D Kinematic Modeling and Evaluation of Rough-Terrain Locomotion Modes for an ExoMars-like Mobility Subsystem". MA thesis. Technische Universität München.
- Wilkie, David, Jur Van Den Berg, and Dinesh Manocha (2009). "Generalized velocity obstacles". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5573–5578.
- Wong, Jo Yung (2008). *Theory of ground vehicles*. John Wiley & Sons.

- Woods, Mark, Andy Shaw, Dave Barnes, Dave Price, Derek Long, and Derek Pullan (2009). "Autonomous science for an ExoMars Rover-like mission". In: *Journal of Field Robotics* 26.4, pp. 358–390.
- Wu, Mulun, Shi-Lu Dai, and Chenguang Yang (2020). "Mixed reality enhanced user interactive path planning for omnidirectional mobile robot". In: *Applied Sciences* 10.3, p. 1135.
- Xie, Li, Christian Henkel, Karl Stol, and Weiliang Xu (2018). "Power-minimization and energy-reduction autonomous navigation of an omnidirectional Mecanum robot via the dynamic window approach local trajectory planning". In: *International Journal of Advanced Robotic Systems* 15.1.
- Xu, Bin, Daniel J Stilwell, and Andrew J Kurdila (2013). "Fast path re-planning based on fast marching and level sets". In: *Journal of Intelligent & Robotic Systems* 71.3, pp. 303–317.
- Xu, Jinshan, Yongreng Chen, Xi Yang, and Meiyu Zhang (2019). "Fast Marching Based Path Generating Algorithm in Anisotropic Environment With Perturbations". In: *IEEE Access*.
- Xu, Qi-Lei, Tao Yu, and Jie Bai (2017). "The mobile robot path planning with motion constraints based on Bug algorithm". In: *2017 Chinese Automation Congress (CAC)*. IEEE, pp. 2348–2352.
- Yan, Yupei and Yangmin Li (2016). "Mobile robot autonomous path planning based on fuzzy logic and filter smoothing in dynamic environment". In: *2016 12th World congress on intelligent control and automation (WCICA)*. IEEE, pp. 1479–1484.
- Yap, Peter, Neil Burch, Robert Craig Holte, and Jonathan Schaeffer (2011). "Block A*: Database-driven search with applications in any-angle path-planning". In: *Twenty-Fifth AAAI Conference on Artificial Intelligence*.
- Yi, Chuho, Seungdo Jeong, and Jungwon Cho (2012). "Map representation for robots". In: *The Smart Computing Review* 2.1, pp. 18–27.
- Yoshida, Kazuya and Hiroshi Hamano (2002). "Motion dynamics of a rover with slip-based traction model". In: *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. Vol. 3. IEEE, pp. 3155–3160.
- You, Xiaoming, Kai Liu, and Sheng Liu (2016). "A chaotic ant colony system for path planning of mobile robot". In: *International Journal of Hybrid Information Technology* 9.1, pp. 329–338.
- Yu, Xiaoqiang, Ping Wang, and Zexu Zhang (2021). "Learning-Based End-to-End Path Planning for Lunar Rovers with Safety Constraints". In: *Sensors* 21.3, p. 796.
- Zafar, Mohd Nayab and JC Mohanta (2018). "Methodology for path planning and optimization of mobile robots: A review". In: *Procedia computer science* 133, pp. 141–152.
- Zavlangas, Panagiotis G and Spyros G Tzafestas (2003). "Motion control for mobile robot obstacle avoidance and navigation: a fuzzy logic-based approach". In: *Systems Analysis Modelling Simulation* 43.12, pp. 1625–1637.

- Zhang, Fuhai, Ning Li, Tiegang Xue, Ying Zhu, Rupeng Yuan, and Yili Fu (2019). "An Improved Dynamic Window Approach Integrated Global Path Planning". In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, pp. 2873–2878.
- Zhang, Han-ye, Wei-ming Lin, and Ai-xia Chen (2018). "Path planning for the mobile robot: A review". In: *Symmetry* 10.10, p. 450.
- Zhang, Haojie, Yudong Zhang, and Tiantian Yang (2020). "A survey of energy-efficient motion planning for wheeled mobile robots". In: *Industrial Robot: the international journal of robotics research and application*.
- Zhang, Tao, Kun Xu, Zhixiao Yao, Xilun Ding, Zeng Zhao, Xuyan Hou, Yong Pang, Xiaoming Lai, Wenming Zhang, Shuting Liu, et al. (2019). "The progress of extraterrestrial regolith-sampling robots". In: *Nature Astronomy* 3.6, pp. 487–497.
- Zhang, Yong, Dun-wei Gong, and Jian-hua Zhang (2013). "Robot path planning in uncertain environment using multi-objective particle swarm optimization". In: *Neurocomputing* 103, pp. 172–185.
- Zhou, Zhiyu, Junjie Wang, Zefei Zhu, Donghe Yang, and Jiang Wu (2018). "Tangent navigated robot path planning strategy using particle swarm optimized artificial potential field". In: *Optik* 158, pp. 639–651.
- Zhu, Anmin and Simon X Yang (2009). "An adaptive neuro-fuzzy controller for robot navigation". In: *Recent Advances in Intelligent Control Systems*. Springer, pp. 277–307.

Resumen de la Tesis Doctoral

Planificación Extrema de Caminos para Robots Móviles de Exploración

Desde antaño los humanos hemos invertido numerosos esfuerzos en expandir los límites de nuestro conocimiento. Hemos llegado a cada vez más lugares a lo largo y ancho de este mundo, intentando entender los detalles acerca de todo lo que nos rodea. Además, nos hemos establecido en algunos de esos lugares formando colonias. En la actualidad, con un mundo cada vez más globalizado y la humanidad presente en gran parte de la superficie de nuestro planeta, la siguiente frontera la constituye el espacio profundo. Soñamos con colonizar otros mundos como Marte. Sin embargo, esto requiere solventar en las próximas décadas multitud de problemas de índole tecnológico, ético y legal (Levchenko et al., 2019). Por el momento, las agencias espaciales han ido enviando en las últimas décadas múltiples tipos de sistemas robóticos para explorar estos entornos extraterrestres. Estas tecnologías pueden beneficiarse de las que son actualmente empleadas en la Tierra y viceversa. Por ejemplo, para llevar a cabo la primera respuesta ante una situación de catástrofe el uso de agentes autónomos de exploración adquiere importancia al permitir adquirir un mejor conocimiento sobre la situación. Esto puede servir de ayuda de cara a la búsqueda de posibles víctimas y para mejorar la planificación de las acciones de los operarios de primera intervención (Seppänen et al., 2015).

Exploración, reconocimiento, inspección. Todos estos términos hacen referencia al acto de aventurarse en entornos parcial o totalmente desconocidos con la intención de reducir la incertidumbre que reina en ellos. Este tipo de acciones puede reportar grandes beneficios puesto que la nueva información adquirida permite aumentar la capacidad de planificar futuras acciones. Esto está presente en nosotros desde el momento en el que nacemos: a medida que crecemos nos aventuramos, adquirimos nueva información, aprendemos y, como resultado, nuestras posibilidades para sobrevivir y seguir interactuando con el entorno se ven aumentadas. Además, el hecho de percibir fenómenos que no comprendemos a corta edad nos motiva a buscar un mejor entendimiento acerca de estos (Köster et al., 2020). Es importante recalcar un hecho: la exploración lleva implícita la realización de una acción física, una acción motora. Esta acción puede consistir en moverse, desplazarse de un punto en el espacio a otro. Posiblemente este desplazamiento resulta en una mejora en la percepción del entorno al realizarla desde otra perspectiva. Esta información mejora nuestro

entendimiento y nuestra comprensión acerca de la situación, y, por lo tanto, se abren más posibilidades a la hora de realizar futuras acciones.

El acto de explorar para nuestro beneficio ya no es exclusivo de los humanos. En las últimas décadas un mayor número de sistemas autónomos han sido puestos en escena. Hay multitud de razones por las que optar por estos sistemas. Una de las más importantes es el evitar poner en riesgo vidas humanas en ciertas situaciones. El diseño de estos vehículos robóticos es diferente de acuerdo con el medio en el que se desplazan: mar, aire, espacio, tierra... El último de estos es donde el foco de esta tesis está puesto: robots móviles de tierra. Esta tesis presenta los resultados del trabajo llevado a cabo para mejorar las capacidades autónomas de este tipo de vehículos. En otras palabras, presenta aquellos algoritmos y técnicas que determinan el modo en el que el vehículo decide por dónde desplazarse y cómo. En aras de delimitar mejor el alcance de esta tesis, se define como vehículo de tierra todo aquel que emplea una superficie para desplazarse sobre la misma. Aquí se asume que la acción de la gravedad permite a estos vehículos permanecer un contacto constante con la superficie. Esta tesis por lo tanto no considera cualquier maniobra que implique hacer que el robot se separe de la superficie, como es el caso de aquellos robots capaces de realizar maniobras de saltos. Así pues, esta tesis pone el foco en robots con ruedas diseñados para ser exploradores. Este es el caso de los rovers: vehículos robóticos que son usados para explorar superficies extraterrestres como la de otros planetas y satélites. Estos sistemas deben navegar a través de zonas donde la presencia de humanos es todavía inviable. Además, de forma similar a los vehículos con ruedas empleados en misiones de búsqueda y rescate, el poder tratar las particularidades de un terreno complejo y desestructurado es de interés para poder preparar un curso de acción adecuado, minimizando o incluso prescindiendo de la intervención humana.

En muchas arquitecturas de navegación una de las partes más importantes es la encargada de la generación de caminos. Cada uno de estos caminos sirven de guía para el desplazamiento del vehículo tras un proceso de deliberación. Pueden conectar la propia posición del vehículo (la posición de origen) con un destino de interés (la posición de meta). También conocido como guía, el proceso por el cual se genera el camino se denomina planificación de caminos. Esta última denominación es la que se emplea aquí en esta tesis. Además, esta planificación de caminos está orientada aquí a ser usada con escenarios desestructurados, escenarios presentando un terreno irregular. Este tipo de lugares constituyen un reto para el sistema de navegación de cualquier robot móvil. Este sistema debe ser efectivo, permitiendo llevar al robot de un lugar a otro de forma segura, y eficiente, minimizando los recursos invertidos en ello (por ejemplo, la energía). Sin embargo, las irregularidades del terreno pueden en algunos casos afectar negativamente al movimiento del robot. Por este motivo el planificador de caminos debe tener en cuenta las capacidades locomotoras del robot. Además, la naturaleza extrema de estos escenarios puede limitar las capacidades de percepción de no solo el robot sino también de cualquier medio

externo como puede ser un dron o un satélite. Esto se traduce en una disponibilidad limitada de la información que describe el escenario. El planificador debe entonces tener este hecho en cuenta y actualizar el camino de manera dinámica (replanificar) cuando sea necesario de acuerdo con las actualizaciones en dicha información. A continuación, se introducen dos breves comentarios sobre campos de aplicación de la navegación autónoma con condiciones extremas. Estas son la exploración de entornos planetarios extraterrestres y la exploración de lugares que han sufrido una catástrofe en la Tierra. En el primer caso, un rover debe navegar a través de entornos que presentan un terreno complicado y disponiendo de limitada información sobre los mismos. En el segundo caso, un robot móvil debe desplazarse a través de escenarios todoterreno donde el tiempo y/o el consumo de energía son críticos para llevar a cabo su misión.

Exploración planetaria

La superficie de otros cuerpos celestes podría albergar pistas acerca del origen de la vida y del universo. Además, estudiando su historia geológica y los materiales que se encuentran en ella podríamos entender algunos de los procesos que ocurren en la Tierra (T. Zhang et al., 2019). Además, existen recursos valiosos que podrían ser aprovechados por la industria y las agencias espaciales en futuras misiones espaciales. Estos recursos (p.ej. agua en la Luna) podrían servir de ayuda a la hora de establecer colonias humanas en lugares extraterrestres. Sin embargo, ir a estos lugares a inspeccionarlos requiere el uso de exploradores. Hoy en día existen multitud de dificultades para transportar y mantener a salvo humanos en un viaje espacial. Por este motivo, su presencia prolongada en entornos extraterrestres es todavía inviable. Así pues, el uso de agentes robóticos o rovers, más baratos de mantener y sin ser criaturas vivas, permitiría realizar la tarea de llevar y operar instrumentos científicos en escenarios planetarios y en el espacio (Gao et al., 2017).

Aunque el uso de robots para la exploración planetaria extraterrestre está justificado, todavía existen multitud de cuestiones. Una de ellas tiene que ver con las comunicaciones entre las estaciones en la Tierra y estos robots. El retardo en el envío y recibimiento de mensajes cuando estos sistemas están en la Luna puede llegar a ser de entre 3 y 10 segundos, lo cual permite el uso de teleoperación hasta cierto nivel (Y. Wang, 2021). Sin embargo, las comunicaciones entre las estaciones en la Tierra y los rovers marcianos empleando la red Deep Space Network (DSN) pueden requerir de hasta aproximadamente 40 minutos para llevar un mensaje a la ida y a la vuelta (Lester et al., 2017). Este es un enorme retardo que imposibilita el control directo de estos sistemas: el operador debe esperar hasta que llegue el siguiente mensaje, analizar la información contenida en él y, posteriormente, enviar el siguiente mensaje de vuelta a Marte (Bresina et al., 2005). Además, las comunicaciones entre la Tierra y Marte están limitadas por la órbita del Mars Relay Network, permitiéndolas tan solo unas pocas veces por cada sol marciano (el cual equivale aproximadamente

un día terráqueo). Debido a ello, los rovers tan solo pueden navegar distancias cortas por cada sol.

Para compensar los problemas expuestos en las comunicaciones, se ha ido mejorando de forma progresiva la autonomía de los rovers que han sido enviados a Marte. En 1997, la National Aeronautics and Space Administration (NASA) envió a Marte el primer rover capaz de navegar con cierta autonomía con el nombre de Sojourner (Bajracharya et al., 2008). Este relativamente pequeño robot era capaz de llevar a cabo autónomamente operaciones simples tales como moverse hasta cierto punto (Mishkin et al., 1998). Llegó a alcanzar una distancia total de unos 100 metros, sirviendo como una prueba de concepto satisfactoria para enviar más robots de exploración a Marte. La siguiente misión, la Mars Exploration Rover (MER), consistió en el envío de dos rovers gemelos al planeta rojo: Spirit (Arvidson, Bell, et al., 2010) y Opportunity (Arvidson, Ashley, et al., 2011). Para esta ocasión, la NASA incrementó las capacidades de autonomía de ambos robots, dotándoles de un planificador local de caminos llamado GESTALT (Biesiadecki et al., 2006). Durante la realización de esta misión, los ingenieros e investigadores de la NASA se dieron cuenta de que el uso de un planificador local no era suficiente para alcanzar una navegación autónoma completa. El rover podía quedarse atascado en diversas situaciones en las que se encontraba parcialmente rodeado de rocas. Por este motivo se instaló también un planificador global de caminos. De esta manera, se podía inicialmente planificar una larga travesía mientras que el planificador local generaba dinámicamente caminos que evitaran los obstáculos (Maimone et al., 2007). Este sistema de autonomía se mantuvo para el rover de la misión Mars Science Laboratory (MSL), el rover Curiosity, allá por 2011 (Lele, 2014). Sin embargo, el envío manual de comandos por parte de los operarios de tierra es todavía la estrategia más común a la hora de navegar en Marte. Normalmente, para cada día marciano (o sol) una serie de tareas eran planificadas con anterioridad y posteriormente enviadas a estos vehículos.

Una de las futuras misiones es la del envío del rover Rosalind Franklin al planeta rojo. Este envío forma parte de la campaña ExoMars liderada por la Agencia Espacial Europea (ESA). Inicialmente planificada para 2020 (Vago et al., 2017), ha sido recientemente retrasada para el 2022 al momento en el que esta tesis está siendo escrita. El principal propósito de esta misión es la de extraer y analizar muestras del subsuelo marciano para buscar cualquier pista de vida actual o pasada en el planeta rojo. Una de las principales particularidades de este rover es la disposición de articulaciones actuadas en la parte superior de cada una de sus patas. Estas le permiten desplegar y replegar cada una de sus patas de una manera similar a la acción de andar. Por este motivo el modo de locomoción en cuestión es denominado Wheel-walking (Patel et al., 2010; Woods et al., 2009). Este modo hace que el rover aumente la tracción de sus ruedas en terrenos arenosos, tal y como fue demostrado en experimentos previos (Azkarate, Zwick, et al., 2015). De esta forma, el rover podría evitar quedarse atrapado en una situación similar a la que experimentó el rover

Spirit en 2009 (Arvidson, Bell, et al., 2010). Otra futura misión es la Mars Sample Return (MSR) (Muirhead et al., 2019). Esta consiste en la recogida de muestras de material marciano guardadas previamente en tubos y enviarlas de vuelta a la Tierra. Estos tubos son depositados en la superficie marciana con anterioridad por el rover Perseverance de la NASA, el cual llegó al planeta rojo en febrero de 2021. A continuación, otro rover construido por la ESA, el Sample Fetch Rover (SFR), tendrá que llevar a cabo la tarea de ir a recogerlos. Para esta misión la eficiencia vendrá dada por el número de muestras que puedan ser tomadas dentro de un intervalo de tiempo determinado. Así pues, se necesitarán grandes capacidades en cuanto a movilidad y autonomía. De acuerdo con el sistema de clasificación de la autonomía propuesta por la European Cooperation for Space Standardization, un nivel E4 (navegación autónoma completa) sería el idóneo para que el rover SFR cumpla con las expectativas impuestas, así como para cualquier otro futuro rover con que deba llevar a cabo tareas cada vez más complejas. Un ejemplo de esto último lo constituye el uso de sistemas robóticos móviles que permitan construir infraestructura sobre la superficie lunar (Govindaraj et al., 2019).

En resumen, mejorar la autonomía de los rovers está justificado por dos razones. En primer lugar, la teleoperación directa de estos sistemas es compleja o incluso irrealizable. Cualquier otro tipo de estrategia para comandar estos sistemas sufrirá de retardos en el envío de mensajes. Esto compromete la eficiencia de la misión como consecuencia. En segundo lugar, la complejidad de las tareas que se prevé que los rovers realicen en el futuro está aumentando. De hecho, el futuro rover SFR constituye un ejemplo de ello: deberá buscar determinadas muestras, aproximarse a ellas, recogerlas con ayuda de un manipulador y finalmente llevarlas y depositarlas en un módulo marciano. Todo ello llevado a cabo dentro de un limitado margen de tiempo.

Robótica de búsqueda y rescate

De acuerdo con investigaciones previas, existen varias acciones que influyen en cómo actuar ante una situación de catástrofe: mitigación, preparación, respuesta y recuperación (Jorge et al., 2019). La tercera de ellas, la respuesta, corresponde a toda aquella tarea llevada a cabo justo después de que el desastre ocurra. Desde la tragedia acontecida con los edificios del World Trade Center, que colapsaron debido a un atentado terrorista en 2001, ha habido decenas de casos en los que el uso de plataformas móviles robóticas ha resultado beneficioso (Murphy, 2004). Estas han servido de ayuda en situaciones de desastre de índole tanto natural (por ejemplo, un terremoto) como artificial (por ejemplo, una mina que se derrumba total o parcialmente). En estos casos, un robot o equipo de robots debe realizar operaciones de exploración para, entre otros, buscar víctimas potenciales o inspeccionar el estado de cualquier estructura dañada (Murphy et al., 2016). Por ejemplo, en 2013 un tsunami golpeó la costa este de la isla principal de Japón. Esto tuvo como consecuencia serios daños en algunos de los edificios de una planta de energía nuclear en Fukushima, provocando

como resultado una fusión. Varios robots móviles fueron enviados como parte de la respuesta de emergencia para atender este desastre, siendo tele-operados de manera que pudieran alcanzar áreas contaminadas por la radiación, dañinas para cualquier ser humano (Nagatani et al., 2013). Otro ejemplo lo constituye el uso de un robot para inspeccionar un par de iglesias que fueron dañadas después del gran terremoto que ocurrió en Italia en 2016 (Kruijff-Korbayová et al., 2016). La principal razón por la que estos robots fueron empleados era el elevado riesgo de que ambas iglesias se derrumbaran, lo cual sería perjudicial para cualquier operador humano situado en su interior. Otra operación de rescate consiste en poner a salvo a personas que se han quedado atrapadas en montañas debido a accidentes o incluso elementos externos tales como avalanchas (Silvagni et al., 2017).

Aunque está demostrado que la teleoperación de estos sistemas es segura y fiable, esta requiere la intervención de un número de operarios que ya es limitado de por sí (Birk et al., 2006). Además, las comunicaciones inalámbricas pueden ser comprometidas en escenarios tales como una mina o una planta nuclear (Murphy, 2012), lo cual motiva a mejorar las capacidades de autonomía de los robots de rescate (Bogue, 2019). De esta manera, un operario podría enviar comandos a un mayor número de robots gracias a la automatización de las tareas de navegación de bajo nivel. Otra dificultad que presenta la teleoperación de robots móviles de rescate es la de poder controlar el movimiento cuando estos presentan un gran número de grados de libertad. La interacción de estos sistemas con un terreno irregular no es algo trivial, puesto que preservar su estabilidad o evitar que colisionen son tareas difíciles para el usuario y requieren el uso de interfaces complejas tales como exoesqueletos (Klamt et al., 2018). Un entorno desestructurado puede presentar grandes dificultades en forma de rocas, cuevas o terrenos con distintas propiedades terramecánicas. Todo esto hace surgir la necesidad de desarrollar planificadores de caminos que tengan en cuenta estos hechos y que permitan que los vehículos de exploración encuentren caminos seguros y óptimos (Delmerico et al., 2019). Para ello el uso de robots dotados de patas es prometedor, gracias a su adaptabilidad a múltiples tipos de terreno irregular (Miki et al., 2022). Por último, es de interés dotar a los robots de rescate de capacidades que les permitan socorrer rápidamente a cualquier víctima de un desastre.

Contribuciones

Esta tesis presenta y detalla el trabajo realizado para crear una serie de contribuciones al estado del arte de la planificación de trayectorias para robots móviles de tierra. En particular, esta tesis se centra en robots móviles que puedan ser empleados en los campos de la exploración planetaria y la búsqueda y rescate. Tal y como se ha explicado anteriormente, es de suma importancia que estos robots puedan navegar a través de entornos desestructurados en exteriores de una forma totalmente autónoma, sin requerir de ningún tipo de teleoperación. Como paso preliminar, se

ha realizado una exhaustiva revisión de la literatura en la que se han encontrado ciertas cuestiones a mejorar. Estas cuestiones abarcan el uso de distintos modos de locomoción, la combinación de mapas con diferentes formatos y la consideración de la irregularidad del terreno. Las contribuciones de esta tesis, conformando un total de cinco, tratan estas cuestiones y son las que se exponen a continuación:

- **Contribución 1: Planificación óptima de trayectorias considerando múltiples modos de locomoción.** Los entornos desestructurados tales como el de un escenario catastrófico o uno extraterrestre pueden suponer un desafío para las capacidades locomotoras de cualquier robot móvil de tierra. La composición del terreno puede no sólo hacer aumentar el consumo energético del robot sino también restringir su movimiento. En un caso extremo, el terreno podría incluso dejarlo inmovilizado, provocando el fallo en la misión. Además, la forma misma del terreno podría también hacer que fuera más difícil atravesarlo. Por este motivo, dotar a los robots de una configuración cinemática capaz de ejecutar múltiples modos de locomoción puede significar que estos se adapten mejor al terreno. De esta manera, los robots pueden ver aumentadas sus posibilidades de navegación. Tener en consideración esta capacidad es necesario en un planificador de trayectorias para aprovechar las ventajas que supone. Además, el planificador en sí mismo debería ser capaz ya de entrada de encontrar el camino óptimo. Teniendo esto en mente, la primera contribución de esta tesis es la del uso de un planificador de caminos óptimo junto a una función de coste basada en la capacidad de reconfigurar la locomoción. En particular, esta contribución tiene como referencia el ya mencionado rover Rosalind Franklin, capaz de ejecutar dos modos de locomoción: Normal-driving y Wheel-walking.
- **Contribución 2: Planificación de caminos a múltiple escala combinando la planificación inicial de una larga travesía con la replanificación dinámica a escala local.** La información que describe el terreno puede provenir de diferentes fuentes y en formatos distintos. Por ejemplo, en exploración planetaria la información inicial sobre un entorno proveniente de imágenes por satélite. Estas pueden dar una estimación preliminar de lo que el rover pueda encontrarse en su camino. Sin embargo, existe todavía una cierta incertidumbre subyacente debida a que las fuentes no son del todo precisas. Por este motivo, la información inicial puede ser dinámicamente complementada con la información proporcionada por el rover mientras este navega. Un ejemplo de ello es cuando el rover detecta obstáculos en su camino que no fueron considerados durante la planificación inicial. Ello se traduce en la necesidad de hacer uso de un planificador de caminos dinámico que tenga en cuenta estas actualizaciones en los datos sobre el entorno. Además, este planificador debe ser capaz de tratar mapas del entorno con un tamaño y resolución diferentes con relación al mapa inicial. Esta tesis presenta una contribución en forma

de planificador de caminos dinámico que funciona a escala global y local. La mayor novedad en este planteamiento es el empleo de algoritmos basados en la resolución de ecuaciones en derivadas parciales, los cuales son globalmente óptimos por definición.

- **Contribución 3: Creación de una función de coste que depende de la dirección para la planificación óptima de caminos en terrenos con cuestas.** La tercera contribución de esta tesis se centra en la navegación de escenarios que presentan superficies inclinadas. La principal consideración para esta contribución es el hecho de que el consumo energético del robot al atravesar una cuesta depende de su orientación. Esto es debido a que la gravedad afecta de manera diferente según la dirección que lleve el robot. Por lo tanto, es necesario el empleo de una función de coste basado en un modelo dinámico que tenga esto en cuenta. En concreto, esta contribución hace uso de un planificador de caminos basado en la resolución de ecuaciones en derivadas parciales con la dicha función de coste basada en la interacción robot-terreno. El planificador empleado aquí es distinto a los empleados en las anteriores contribuciones: depende no solo de la posición sino también de la dirección del vehículo, por lo que es categorizado como anisotrópico. Además, se incluye en esta aplicación el uso de una función de riesgo que sirva para preservar la estabilidad lateral del robot tratando de minimizar en la medida de lo posible el ángulo de alabeo.
- **Contribución 4: Experimentos de campo empleando plataformas móviles experimentales.** La cuarta contribución presentada en esta tesis es la validación de las soluciones introducidas en la segunda y tercera contribución empleando robots reales. Para ello un prototipo de rover y una plataforma móvil basada en el modo de locomoción Skid-steering son empleadas en terrenos irregulares de exteriores. Para ambos casos, se han llevado a cabo experimentos de campo que emulaban operaciones de navegación completas involucrando la planificación de caminos. Esta tesis profundiza en cómo estos experimentos fueron llevados a cabo e incluye reflexiones acerca de su ejecución.
- **Contribución 5: Clasificación general de los algoritmos de planificación de trayectorias más relevantes y empleados con robots móviles sobre tierra.** El número de opciones para planificar caminos que existe en la literatura es enorme. Ello puede abrumar a cualquier persona que trate de elegir un algoritmo de planificación de trayectorias para integrarlo en cualquier sistema de navegación. Por este motivo, la quinta y última contribución de esta tesis es el análisis de la mayor parte de algoritmos de planificación de trayectorias encontrados en la literatura. Como resultado de este análisis, se ha propuesto el uso de un método de clasificación que permita, de forma general, entender más fácilmente las diferentes propuestas que existen en términos de funcionalidad y aplicabilidad.

Contexto y motivación

La principal motivación para la realización del trabajo presentado en esta tesis es la de hacer avanzar el campo de la navegación autónoma para los robots móviles. Existen múltiples niveles de autonomía y la planificación de caminos es clave para alcanzar la autonomía plena. Por ello, este trabajo se centra en mejorar este tipo de algoritmo y su uso en ciertas situaciones. Este avance permite al autor alcanzar el título de Doctor en Mecatrónica por parte de la Escuela de Ingenierías Industriales de la Universidad de Málaga. El autor de esta tesis ha realizado el pertinente trabajo en el Departamento de Ingeniería de Sistemas y Automática, siendo a su vez miembro del Laboratorio de Robótica Espacial de la Universidad de Málaga.

El Departamento de Ingeniería de Sistemas y Automática posee una larga trayectoria en el desarrollo de soluciones robóticas dentro del campo de la búsqueda y rescate. Cada año organiza las Jornadas de Seguridad, Emergencias y Catástrofes, donde se realiza cada año una demostración pública de los sistemas robóticos de rescate. Durante sus estudios de doctorado, el autor ha participado en estas jornadas proporcionando el mismo tipo de planificadores de caminos usados en la primera y segunda contribución de esta misma tesis. Además, esta tesis es apoyada por dos proyectos financiados por el gobierno español, proveyendo recursos financieros, equipamiento y la infraestructura necesaria. Estos proyectos son los siguientes:

- **Multi-Robot System for Cooperation with First Response Human and Canine Rescue Teams in Catastrophe Scenarios (FIRST-ROB).** Este proyecto tiene como número de referencia DPI2015-65186-R. Se centra en el uso de sistemas multi-robot para recopilar información en las primeras fases de la respuesta a un escenario de desastre. Este proyecto ha contribuido a la primera, a la segunda y a la cuarta contribución de esta tesis.
- **Towards Resilient UGV and UAV Manipulator Teams for Robotic Search and Rescue Tasks (TRUST-ROB).** Este proyecto tiene como número de referencia RTI2018-093421-B-I00. Se centra en mejorar la resiliencia de los sistemas mecatrónicos que forman parte de los equipos de respuesta multi-robot. Este proyecto ha contribuido a la tercera y cuarta contribución de esta tesis.

El Laboratorio de Robótica Espacial de la Universidad de Málaga comenzó a funcionar a partir del acuerdo de colaboración entre esta institución y la Agencia Espacial Europea (ESA). Como parte de sus estudios de doctorado, el autor ha llevado a cabo tres estancias de investigación. Estas fueron realizadas en el Centro Europeo de Investigación y Tecnología Espacial (ESTEC), con una duración total de un año. El acuerdo de colaboración entre este centro y la Universidad de Málaga empezó al mismo tiempo en el que comenzaron los estudios de doctorado del autor de esta tesis. Este acuerdo sigue en activo en el momento en el que esta tesis es escrita. Además, el autor colaboró en un proyecto europeo financiado por la Comisión

Europea a través del programa Horizon 2020. Ambos proyectos son detallados a continuación:

- **Autonomous Routing on Extreme Surfaces (ARES).** Este proyecto empezó en 2016 con el número de contrato 4000118072/16/NL/LvH/gp. Su propósito original era la mejora de la autonomía de rovers reconfigurables, creando planificadores de caminos capaces de trabajar con múltiples modos de locomoción. En una segunda fase, el proyecto fue renovado en 2018 con el objetivo de mejorar misiones de búsqueda y recogida de muestras. Esto fue motivado por la necesidad de mejorar las capacidades de navegación autónoma de robots de exploración para futuras misiones a la Luna y Marte. En el contexto de este proyecto se realizaron las primeras dos contribuciones, así como la cuarta contribución de esta tesis. La Sección de Robótica y Automatización de la ESA proporcionaron al autor las herramientas, instalaciones y sistemas robóticos para llevar a cabo algunos de los experimentos descritos en esta tesis. Este proyecto ayudó a llevar a cabo la primera, segunda y cuarta contribución de esta tesis.
- **Autonomous Decision making for very long traverses (ADE).** Este proyecto comenzó en marzo de 2019 y duró hasta mayo de 2021. Financiado por la Comisión Europea bajo el programa Horizon 2020, el principal objetivo de este proyecto era el de mejorar la autonomía implementada en un prototipo de rover equipado con un manipulador robótico. La Universidad de Málaga participó como uno de los miembros del consorcio, con la responsabilidad de desarrollar los métodos para hacer que el sistema se aproximara a la localización de una muestra. En este proyecto, el objetivo de estos métodos era el de producir el movimiento del brazo necesario para posicionar el efector final sobre dicha localización. El proyecto finalizó con una serie de experimentos de campo donde el software creado por cada miembro del consorcio fue integrado. Este proyecto contribuyó a la realización de la quinta contribución de esta tesis. Además, los algoritmos presentados en esta tesis sirvieron como la base para implementar los métodos usados en este proyecto para planificar autónomamente las trayectorias para el prototipo de rover.

Publicaciones

Como parte de la diseminación de los resultados de los mencionados proyectos de investigación se realizaron varias publicaciones. Estas fueron publicadas tanto en conferencias como en revistas. Algunas de ellas avalan las contribuciones que presenta esta tesis y son las siguientes:

- **Path Planning for Reconfigurable Rovers in Planetary Exploration.** Autores: Pérez-del-Pulgar, C. J., Sánchez, J. R., Sánchez, A. J., Azkarate, M., Visentin, G. Publicado en *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pág. 1453-1458, en 2017. Esta publicación de conferencia respalda

la primera contribución de esta tesis. Detalla el uso de un planificador de caminos que considera múltiples modos de locomoción para un robot móvil planetario con ruedas. Se empleó un entorno de simulación basado en el software V-REP para validar la solución desarrollada. También se demostró cómo, mediante la consideración de estos modos de locomoción en áreas que contienen distintos tipos de terreno, el consumo total de energía puede ser minimizado de manera óptima. Esta publicación es parcialmente financiada por el proyecto ARES.

- **Dynamic Path Planning for Reconfigurable Rovers using a Multi-layered Grid.** Autores: Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., Azkarate, M., Gerdes, L., García-Cerezo, A. Publicado en *Engineering Applications of Artificial Intelligence* (2019 Factor de impacto: 4.201, Q1), vol. 86, pág. 32-42, en 2019. Partiendo de la base de la anterior publicación de conferencia, esta publicación de revista no solo considera el uso de rovers reconfigurables, sino también el uso de información sobre el entorno proveniente de varias fuentes. Ello es posible gracias al uso de una malla construida con dos capas, junto a una estrategia de planificación de caminos que combina la planificación inicial en una capa con la actualización dinámica del camino en la otra. Esta solución es validada a través de simulaciones numéricas, así como experimentos de campo. De esta manera, esta publicación aporta a tres de las contribuciones de esta tesis, todas ellas menos la tercera y la quinta. Esta publicación es parcialmente financiada por los proyectos ARES y FIRST-ROB.
- **Path Planning for Autonomous Mobile Robots: a Review.** Autores: Sánchez-Ibáñez, J.R., Pérez-del-Pulgar, C.J., García-Cerezo, A. (2021). Publicado en *Sensors* (2020 Factor de impacto: 3.576, Q1), vol. 21, pág. 7898, en 2021. Esta publicación de revista se crea tras el análisis de algoritmos de planificación de trayectorias a lo largo de los últimos años durante el periodo del doctorado del autor. Como resultado, esta publicación presenta un novedoso sistema de clasificación que cubre la mayor parte de planificadores que se pueden encontrar en la literatura. Esta publicación fue financiada por el proyecto ADE y respalda la quinta contribución de esta tesis.
- **Optimal Path Planning using CAMIS: Continuous Anisotropic Model for Inclined Surfaces.** Autores: Sánchez-Ibáñez, J.R., Pérez-del-Pulgar, C.J., Serón, J., García-Cerezo, A. Manuscrito en revisión para publicación en revista en el momento en el que esta tesis es escrita. Debido a que el consumo energético de los robots móviles de tierra es diferente según la orientación que tengan encima de una cuesta, es deseable disponer de un planificador de caminos que tenga en cuenta este hecho. Esta publicación de revista presenta los detalles sobre CAMIS, un modelo de coste que representa este comportamiento anisotrópico junto a un planificador de caminos compatible. Además, la inclinación lateral es también tenida en cuenta para reducir la posibilidad de

volcado. De manera similar a la anterior publicación, esta presenta resultados de tanto simulaciones como experimentos de campo. Esta publicación apoya a las contribuciones tercera y cuarta y es financiada por el proyecto TRUST-ROB.

Por otro lado, las primeras dos publicaciones que avalan esta tesis fueron precedidas cada una de ellas por una publicación en las conferencias Advanced Space Technologies in Robotics and Automation (ASTRA) e International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS) respectivamente:

- **Path Planning for Reconfigurable Rovers in Planetary Exploration.** Autores: Sánchez-Ibáñez, J. R., Perez-del-Pulgar, C. J., Azkarate, M. Publicado en *Advanced Space Technologies in Robotics and Automation (ASTRA)*, en 2017. Compartiendo título con la publicación de conferencia que apoya esta tesis, esta publicación se centra más en los aspectos técnicos acerca del entorno de simulación empleado junto al software robótico.
- **Multi-scale path planning for a planetary exploration vehicle with multiple locomotion modes.** Autores: Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., Azkarate, M. Publicado en *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, en 2018. Esta corta publicación da algunas pinceladas acerca del uso de un grid multi-capa para la planificación de caminos, lo cual está totalmente detallado en la segunda publicación que avala esta tesis.

Las contribuciones presentadas en esta tesis y avaladas por las publicaciones mencionadas también han influido en otras. Aquí se recopilan algunas de estas publicaciones en las que el doctorando autor de esta tesis ha participado como co-autor:

- **Choosing the Best Locomotion Mode in Reconfigurable Rovers.** Autores: Pérez-del-Pulgar, C. J., Romeo-Manrique, P., Paz-Delgado, G. J., Sánchez-Ibáñez, J. R., Azkarate, M. Publicado en *Electronics* (2019 Factor de impacto: 2.412, Q2), vol. 8, pág. 818, in 2019. Esta publicación de conferencia avanza en el modelo realizado para la primera contribución y también presenta un método para decidir cuándo pasar del modo Wheel-walking a Normal-driving, de acuerdo con la información proporcionada por los sensores equipados en un rover reconfigurable.
- **Efficient Autonomous Navigation for Planetary Rovers with Limited Resources.** Authors: Gerdes, L., Azkarate, M., Sánchez-Ibáñez, J. R., Joudrier, L., Perez-del-Pulgar, C. J. Publicado en *Journal of Field Robotics* (2020 Factor de Impacto: 3.581, Q1), vol. 37, pág. 1153-1170, en 2020. Esta publicación de revista es un reporte de campo de los experimentos llevados a cabo cerca de las instalaciones de ESA-ESTEC, empleando un prototipo experimental de rover. El planificador de caminos empleado es el mismo que se describe en la publicación de revista *Dynamic Path Planning for Reconfigurable Rovers using a Multi-layered Grid*.

- **Improving Autonomous Rover Guidance in Round-Trip Missions Using a Dynamic Cost Map.** Autores: Paz-Delgado, G. J., Azkarate, M., Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., Gerdes, L., García-Cerezo, A. J. Publicado en *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7014-7019, en 2020. En esta publicación de conferencia se presentó una mejora al planificador de caminos introducido en la publicación de revista anterior *Dynamic Path Planning for Reconfigurable Rovers using a Multi-layered Grid*. Se centra en misiones de ida y vuelta como la que se espera que el rover SFR lleve a cabo en el futuro.
- **Coupled path and motion planning for a rover-manipulator system.** Autores: Sánchez-Ibáñez, J. R., Paz-Delgado, G. J., Romeo-Manrique, P., Pérez-del-Pulgar, C. J., Azkarate, M. Publicado en *Advanced Space Technologies in Robotics and Automation (ASTRA)*, en 2019. Esta publicación de conferencia presenta los resultados obtenidos en la simulación de la planificación de movimientos de un manipulador móvil. El algoritmo empleado para planificar el camino de la base móvil es el mismo empleado en las contribuciones primera y segunda de esta tesis.
- **Enhancing Mobile Manipulation with Synchronized Arm-Locomotion Control.** Autores: Sánchez-Ibáñez, J. R., Domínguez, R., Cordes, F., Pérez-del-Pulgar, C. J. Publicado en *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, en 2020. Esta publicación de conferencia presenta los resultados preliminares de unos experimentos realizados con el rover *SherpaTT* del Centro Alemán de Investigación en Inteligencia Artificial (DFKI). Estos experimentos consistieron en validar el método de planificación de caminos y movimientos desarrollado como parte del proyecto ADE.

Estructura de la tesis

La estructura de esta tesis se basa en siete capítulos. Todos ellos han sido escritos para presentar y desarrollar cada una de las contribuciones mencionadas que conforman esta tesis. Cada capítulo sigue una cierta estructura, empezando con una breve introducción sobre el concepto o conceptos que van a ser presentados. Después, el capítulo proporciona un desarrollo de estos conceptos, exponiéndolos en detalle. Más tarde cada capítulo termina con un pequeño resumen y algunas conclusiones extraídas acerca de lo expuesto. Estas conclusiones incluyen el cómo los conceptos presentados encajan con el trabajo de la tesis y la opinión del autor con respecto a estos. Los capítulos que dan forma a la tesis son los siguientes:

- **Capítulo 1: Introducción.** Plantea las principales cuestiones que motivan la realización de esta tesis. Después, proporciona una detallada exposición de

los objetivos clave a alcanzar. A continuación, este capítulo introduce el contexto bajo el cual se enmarca el trabajo llevado a cabo. Finalmente detalla la estructura de la tesis.

- **Capítulo 2: Estado del arte en planificación de caminos para robots móviles de tierra.** Este capítulo introduce una revisión exhaustiva de la planificación de trayectorias, lo cual constituye la quinta y última contribución que avala esta tesis. Proporciona un análisis de gran parte de los algoritmos de planificación de caminos que existen en la literatura. También propone un método de clasificación que pretende facilitar el entendimiento de estos algoritmos. A continuación, este capítulo presenta un análisis de los principales criterios empleados en la planificación de caminos para navegación autónoma.
- **Capítulo 3: Planificación de caminos óptima e isotrópica para rovers reconfigurables.** Desarrolla la primera contribución de la tesis acerca del uso de varios modos de locomoción en la planificación de caminos. Este capítulo detalla el proceso por el que se crea un modelo que considera varios modos de locomoción. Al final explica las implicaciones que conlleva hacer uso de este modelo en un planificador de caminos isotrópico basado en ecuaciones en derivadas parciales.
- **Capítulo 4: Planificación de caminos dinámica y multicapa.** Este capítulo se centra en la segunda contribución de esta tesis. Expone el problema de emplear información proveniente de distintas fuentes en navegación autónoma. Además, destaca el problema de cómo hacer que un camino sea actualizado dinámicamente. Después detalla la solución propuesta, consistente en una arquitectura de planificación de caminos multicapa. A continuación, este capítulo finaliza con una reflexión acerca del uso de la novedosa solución presentada.
- **Capítulo 5: Modelo de coste anisotrópico para la navegación sobre terrenos inclinados.** La tercera contribución de esta tesis se detalla en este capítulo. Aquí se da un paso más en la formulación del problema de la planificación de caminos. Este consiste en tener en cuenta la diferencia en el coste energético de acuerdo con su orientación en una cuesta. Esto conlleva el empleo de un planificador de caminos anisotrópico junto a un modelo de coste dependiente de la dirección que tenga en cuenta este nuevo criterio.
- **Capítulo 6: Experimentos.** Este capítulo presenta la cuarta y última contribución de esta tesis. Detalla cada uno de los experimentos llevados a cabo para validar las contribuciones anteriores. Algunos de ellos han sido llevados a cabo con plataformas experimentales reales, incluyendo prototipos de rovers y robots móviles de rescate. Después este capítulo presenta los resultados obtenidos en estos experimentos junto a un análisis y reflexión acerca de los mismos.

- **Capítulo 7: Conclusiones y trabajo futuro.** El último capítulo de esta tesis presenta en primer lugar un resumen y conclusiones sobre el trabajo presentado en los anteriores capítulos. Después introduce una exposición sobre las contribuciones presentadas poniendo el foco sobre aquellas cuestiones todavía por resolver. También se indican aquellas mejoras que podrían realizarse en los algoritmos expuestos. El capítulo detalla cómo estas podrían resultar en posibles futuras líneas de investigación.

Conclusiones de la Tesis Doctoral

Las contribuciones presentadas en esta tesis constituyen un gran paso en la mejora de la autonomía de los robots móviles. La mayoría de los planificadores basados en la resolución de ecuaciones en derivadas parciales han sido aplicados únicamente en simulación. Llevarlos a una aplicación real es un paso necesario para identificar cuáles son los principales problemas que debería tener en cuenta un planificador. Como se ha incidido en los primeros capítulos, el ser eficiente puede ser algo crítico. Ello es crucial en aplicaciones extremas tales como la exploración planetaria y la búsqueda y rescate.

Esta tesis avanza en varias líneas de investigación claves. Tres de ellas son las siguientes. Primera, construir un planificador que tenga en cuenta la adaptabilidad de los robots reconfigurables. Segundo, manejar la incertidumbre del escenario a través de información proveniente de diversas fuentes. Tercero, modelar el efecto de la gravedad en travesías sobre cuevas. A continuación, se muestran las conclusiones extraídas para cada una de las contribuciones presentadas.

Contribución 1: Planificación óptima de trayectorias considerando múltiples modos de locomoción.

El Capítulo 3 explica la primera contribución de esta tesis. Además, el Capítulo 6 presenta los detalles de los experimentos en simulación que la sostienen. Esta contribución se apoya en el uso de modelos dinámicos de diferentes modos de locomoción. Estos modelos son combinados en una única función de coste. Esta función tiene en cuenta la movilidad de los robots capaces de reconfigurar su locomoción. Como resultado, esta función de coste les permite adaptarse mejor a las condiciones del terreno. En particular, el Capítulo 3 introduce dos modos: Wheel-walking y Normal-driving. Cada uno de ellos es, de acuerdo con el modelo propuesto, más adecuado para diferentes valores de parámetros terramecánicos. Estos modelos fueron contruidos en base a rovers con patas y ruedas. Los resultados de las simulaciones muestran que Normal-driving es más adecuado en términos energéticos para terrenos con poca resistencia específica y derrape. Por el contrario, para terrenos con mayor resistencia específica es más eficiente emplear el modo Wheel-walking. Los valores exactos para cambiar entre modos son elegidos según la configuración dinámica del robot. El planificador crea caminos que son eficientes energéticamente según estos modos. Además, el planificador determina qué modo de locomoción es el más adecuado para alcanzar cada posición del camino. Los resultados de las simulaciones demuestran cómo el tener en cuenta ambos modos permite encontrar

caminos más cortos. Ello es gracias al reconocimiento de la mejora en la adaptabilidad. La función de coste ha sido creada en base a las especificaciones de un rover reconfigurable real. Los modelos dinámicos preliminares creados mediante simulación han servido como base para la función de coste. Finalmente, las simulaciones de la planificación de caminos se realizaron en mapas artificiales y dejan el método para obtener los parámetros del terreno fuera del alcance de esta tesis.

Contribución 2: Planificación de caminos a múltiple escala combinando la planificación inicial de una larga travesía con la replanificación dinámica a escala local.

El Capítulo 4 explica con detalle la segunda contribución de esta tesis. Esta se basa en una planificación de caminos a múltiple escala basada en el método Fast Marching. Como se menciona en el análisis del estado del arte en planificación de caminos, este tipo de algoritmo no se emplea normalmente en planificación local. Además, la capacidad para replanificar estaba todavía en sus primeras etapas. Había algunas propuestas para actualizar el camino a la misma escala que con la que se ha planificado (por ejemplo, la global). La segunda contribución de esta tesis propone el uso de una malla que combina capas local y global, la malla multicapa. Con esta malla, el planificador puede generar una travesía muy larga y también repararla en ciertas secciones disponiendo de información limitada del entorno. Esto es gracias a la capacidad de trabajar con mapas de diferentes tamaños y resoluciones que posee esta malla especial. El planificador de caminos dinámico multicapa (DyMu) destaca como arquitectura de planificación capaz de planificar a escalas global y local empleando esta malla. Se basa en una versión heurística del método Fast Marching para realizar la actualización local del camino. Este método generalmente no es compatible con restricciones cinemáticas. Sin embargo, DyMu aprovecha la capacidad que tienen muchos rovers de rotar en un mismo sitio. Ello simplifica las condiciones de movilidad que tienen que ser tenidas en cuenta a escala local, puesto que el movimiento no está restringido a ningún radio de curvatura. DyMu fue validado a través de varios experimentos. El Capítulo 6 presenta con todo detalle estos experimentos. Un rover, HDPR, pudo moverse unos cientos de metros de manera totalmente autónoma. El método Conservativo del proceso Local Path Repairing (LPR) fue empleado en estos experimentos. El otro método, Sweeping, fue validado a través de simulación numérica y todavía queda validarlo con una plataforma robótica también. Además, una posterior publicación de conferencia presentó una versión mejorada de DyMu. Fue escrita junto a la Sección de Robótica y Automatización de la ESA (Paz-Delgado et al., 2020). Esta publicación detalla más experimentos que fueron llevados a cabo con el robot HDPR en una misión de ida y vuelta. Finalmente, DyMu tiene la capacidad de ser implementado con otros planificadores basados en la resolución de derivadas parciales. Ello es debido a la sinergia entre ambas capas, con el uso de funciones heurísticas en la capa local basados en información global.

Contribución 3: Creación de una función de coste que depende de la dirección para la planificación óptima de caminos en terrenos con cuestas.

El Capítulo 5 introduce CAMIS. Este es un método para construir una función de coste anisotrópica para ser empleada por planificadores de caminos en escenarios con cuestas. CAMIS hace que el planificador encuentre el camino que se ajuste mejor a determinados criterios. Estos pueden ser la minimización del consumo de energía o preservar la estabilidad lateral. Esto último es equivalente a minimizar el ángulo de alabeo del robot. Para ello, la función anisotrópica de coste toma en consideración la dirección de la cuesta y su gradiente. CAMIS es modelado en base a la función polar inversa de una elipse desplazada de su centro. Con ello, CAMIS cumple con los requerimientos de planificadores anisotrópicos de caminos tales como OUM. Los resultados de dos experimentos de simulación y un experimento de campo sirven para validar el uso de CAMIS en planificación de caminos. En estos experimentos, un robot planificó caminos óptimos para atravesar escenarios con cuestas y alcanzar varios puntos. La primera simulación sirvió para estudiar cómo las propiedades del terreno afectan a la función anisotrópica de coste. Además, sirvió para averiguar cuándo es más adecuado usar un planificador anisotrópico en vez de uno isotrópico. Los resultados muestran que, para escenarios con superficies inclinadas, los planificadores anisotrópicos funcionan mucho mejor cuando el coeficiente de resistencia específica es bajo y el derrape es alto. La reducción en el coste total (energía consumida) es significativa, a cambio de aumentar la carga computacional. Bajo otras condiciones, el planificador isotrópico es más adecuado, por ejemplo, empleando FMM, puesto que su complejidad computacional es mucho menor. La segunda simulación sirvió para analizar el uso de una función de penalización para minimizar el ángulo de alabeo. Esta función aumenta el coste en las direcciones perpendiculares a la de la cuesta. Los resultados muestran cómo esta función hace que el camino vaya más paralelo a la dirección de las cuestas. Como resultado, el camino obtenido reduce el ángulo de alabeo que el robot experimenta cuando avanza. El experimento de campo puso el foco en las dificultades que surgen para aterrizar CAMIS en planificación local. Los resultados muestran cómo los errores de seguimiento que el robot experimenta hacen crecer el ángulo de alabeo. Además, la función de coste no considera el coste de giro. Aun así, el robot se mantuvo paralelo a la dirección de la cuesta. Las estimaciones del planificador sobre la orientación y el consumo de energía estuvieron cercanos a los valores reales.

Contribución 4: Experimentos de campo empleando plataformas móviles experimentales.

El Capítulo 6 presenta dos experimentos de campo, uno para comprobar el funcionamiento de DyMu y otro para CAMIS. El primero de estos consistió en emplear una plataforma móvil experimental en forma de prototipo de rover. Esta dispone de seis ruedas, cuadro de ellas direccionables, y una configuración cinemática rocker-bogie. Este rover es HDPR, y llevaba equipado una cámara frontal estéreo Bumblebee. El software de navegación, basado en el sistema RoCK, es el encargado de

de manejar la información proveniente de las cámaras. También se encarga de enviar comandos a los motores. Una antena GNSS fue también instalada a bordo una estación fija externa proporcionaba la localización. La precisión de esta localización era RTK, de unos pocos centímetros o incluso menos. El escenario, un terreno rocoso que se asemeja al paisaje marciano, contiene cráteres más grandes que el rover. Además, réplicas de rocas fueron distribuidas por todo el lugar. Una publicación anterior (Gerdes et al., 2020), recoge información más detallada acerca de esta instalación. Con respecto al segundo experimento de campo, se empleó una plataforma móvil de cuatro ruedas llamada Cuádriga. Esta fue equipada con un módulo RTK GNSS, del modelo Emlid Reach M+, conectado a la red andaluza de posicionamiento RAP. De esta manera, la precisión en la localización también estaba al orden de centímetros. Un sensor de corriente eléctrica, la placa ACS754, fue instalada junto a una placa Arduino AtMEGA. Sirvieron para medir la corriente suministrada a los motores con la que posteriormente se construyó la función anisotrópica de coste. Con relación a la región de interés, esta contenía una cuesta en el medio. Este terreno fue modelado en base a imágenes geo-referenciadas que fueron procesadas mediante técnicas de fotogrametría. La configuración de ambos experimentos de campo sirvió para tener un mejor entendimiento de las soluciones de planificación de caminos propuestas en esta tesis. Esta configuración permitió llevar a DyMu y a CAMIS a aplicaciones reales.

Contribución 5: Clasificación general de los algoritmos de planificación de trayectorias más relevantes y empleados con robots móviles sobre tierra.

La última contribución de esta tesis sirve para complementar a las anteriores. Esta surge tras el tiempo invertido en la revisión de la literatura. Esta contribución consiste en el uso de un método para clasificar planificadores de trayectorias. Este método organiza los algoritmos de planificación que han sido encontrados por el autor de esta tesis. Esta clasificación divide estos algoritmos en cuatro clases: Computación Reactiva, Computación Suave, Búsqueda en el Espacio de Configuraciones y Control Óptimo. Los algoritmos que pertenecen a la Computación Reactiva suelen ser empleados para planificar caminos a escala local en escenarios con obstáculos. Los algoritmos de Computación Suave son configurables por el usuario a través de reglas difusas, redes neuronales o parámetros que definen técnicas de computación evolutiva. A continuación, los algoritmos de Búsqueda en el Espacio de Configuraciones extraen el camino de una malla. Esta malla puede ser dinámicamente creada o ser proporcionada al principio, siendo estática. Finalmente, los algoritmos de Control Óptimo general el camino óptimo a partir de un entorno conocido y modelado. La clasificación propuesta, presentada en el Capítulo 2, sirve como punto de partida para cualquiera interesado en elegir un planificador de caminos para una aplicación de navegación autónoma.