UNIVERSIDADE D
COIMBRA

Dora Sofia Bernardes Lourenço

3D Navigation for Ground Robots in Forestry Environments

Master's Dissertation in MIEEC, supervised by Dr. David B. S. Portugal and presented to the Faculty of Science and Technology of the University of Coimbra

October 2020

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

3D Navigation for Ground Robots in Forestry Applications

Dora Sofia Bernardes Lourenço

October 2020

3D Navigation for Ground Robots in Forestry Applications

Supervisor:

Dr. David B. S. Portugal

Jury:

Prof. Dr. Jorge Manuel Miranda Dias

Prof. Dr. Paulo Jorge Carvalho Menezes

Dr. David Bina Siassipour Portugal

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

October 2020

Acknowledgements

In this short text, I would like to acknowledge the various people who have helped me, in one way or another, for the planning, development and conclusion of this work. These people include not only the academics and colleagues, but also family and friends who have assisted me in different ways during this last year.

First of all, I thank my supervisors, Dr. David Portugal and Dr. Gonçalo Martins, for the encouragement, for always keeping me on track, for continuously making me rethink and slowly refining my work through their interventions. I also would like to thank to Dr. João Filipe Ferreira, because despite not being my supervisor, offered his great motivation and crucial guidance since the beginning of this work, always helpful to answer any questions.

I would also like to thank Prof. Dr. Rui Rocha and Ingeniarius, Ltd. for the invaluable technical support for carrying out this work.

A special thanks to my closest family: my parents and my brother, who have unconditionally supported me through this chapter of my life. These are the people that made me who I am, and whom by their merely existence, allow me to go on, with their unconditional love and encouragement.

My appreciation goes, as well, to my closest friends for their support, friendship and listening skills, even when they had no idea what I was speaking about. To Quantunna - Tuna Mista da Faculdade de Diências e Tecnologias da Universidade de Coimbra, this family that was part of 4 years of my academic path. I bring great friends and stories to life in my memory.

Resumo

A gestão e limpeza florestal são tarefas cruciais na prevenção de incêndios florestais, uma causa comum de perdas humanas e económicas em países como Portugal. Além disso, a limpeza florestal pode ser laboriosa e perigosa, tornando-a uma forte candidata à automatização, sendo que, a forma que nós procuramos é aquela em que o robô limparia a floresta de forma totalmente autonoma. Uma tecnologia essencial para a limpeza autónoma das florestas é a navegação, através da qual um robô é capaz de se deslocar na floresta de forma segura e eficiente, o que constitui actualmente um desafio científico e tecnológico significativo.

Esta tese apresenta um método de planeamento local 3D em ambientes exteriores em aplicações florestais, de maneira a facilitar a navegação autónoma de um veículo terrestre não tripulado. O método proposto integra um módulo que estima o esforço mecânico envolvido na travessia do espaço circundante. Isto permite o sistema tenha em consideração o esforço mecânico ao planear os caminhos a percorrer, permitindo planear de forma mais eficiente.

Começamos por apresentar um levantamento em navegação e planeamento de caminhos, concluindo que a navegação em ambientes florestais 3D é muito desafiante devido a vários fatores, tais como o terreno acidentado e escorregadio que dificulta o movimento, a natureza não estruturada do ambiente que dificulta as técnicas de mapeamento, ou fatores dinâmicos como o vento ou a chuva. Existem alguns exemplos de sucesso nesta área, tendo alguns deles atingido o objetivo de navegar num ambiente florestal 3D, mas nenhum deles tem em conta o esforço mecânico do robô.

A nossa solução foi validada utilizando um simulador realista de ambiente florestal 3D, que inclui um robô e a informação recebida dos seus sensores. Os resultados mostram que alguns algoritmos existentes dão prioridade ao caminho e tempo mais curto ao escolher as trajetórias, ignorando a topologia do terreno. Também demonstrámos que a nossa abordagem pode navegar num ambiente 3D, contabilizando o custo do caminho, ou seja, o esforço mecânico do robô. Isto resulta numa técnica que faz com que o robô escolha os caminhos com menor esforço, dando prioridade a descidas e terrenos planos, quando possível.

Abstract

Forest management and clearing are crucial tasks in preventing wildfires, a common cause of human and economical loss in countries such as Portugal. Furthermore, forest clearing can be tedious, repetitive and dangerous, making it a strong candidate for automation Specifically, levels of automation in which the robot would clean the forest completely autonomously are preferred. A key enabling technology for autonomous forest clearing is navigation, whereby a robot is able to move through the forest safely and efficiently, which currently constitutes a significant scientific and technological challenge.

This thesis presents an innovative method for 3D local planning in outdoor environments in forestry applications to facilitate autonomous navigation of an Unmanned Ground Vehicle (UGV). The main practical output of the work is a module that estimates the mechanical effort involved in traversing the surrounding space, allowing the system to take mechanical effort into consideration when planning paths to traverse, enabling it to plan more efficiently.

We start by presenting a survey in navigation and path planning, concluding that navigating in 3D forestry environments is very challenging due to several factors, such as the rough and slippery terrain that interfere with the movement, the unstructured nature of the environment that hinders static mapping techniques, or dynamic factors such as wind or rain. There are some successful examples in this area, some of them having reached the goal of navigating in a 3D forest environment, but none of them take into account the mechanical effort of the robot.

Our solution was validated using a realistic 3D forest environment simulator, including a robot and the information received from its sensors. Results show that existing algorithms prioritise to the shortest path and the shortest time when choosing the trajectories, ignoring terrain topology. We also demonstrated that our approach can navigate in a 3D environment, accounting for the cost of the path, i.e., the mechanical effort of the robot. This results in a technique that makes the robot choose the paths with less effort, prioritising the downward slopes and level paths, when possible.

Contents

List of Acronyms

API Application Programming Interface

C-SLAM CSIRO SLAM

CSIRO Commonwealth Scientific and Industrial Research Organisation

DWA Dynamic Window Approach

EBand Elastic Band

GPS Global Positioning System

IMU Inertial Measurement Unit

JSON JavaScript Object Notation

LIDAR Light Detection And Ranging

NIR Near-Infrared

ROS Robot Operating System

RRT Rapidly Exploring Random Trees

RTK Real Time Kinematics

RTP Rollout Trajectory Planner

SEMFIRE Safety, Exploration and Maintenance of Forests with Ecological Robotics

SLAM Simultaneous Localization and Mapping

SSM Sensor Sharing Manager

TEB Timed Elastic Band

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

USV Unmanned Surface Vehicle

UUV Unmanned Underwater Vehicle

UWV Unmanned Water Vehicle

VFH Vector Field Histogram

List of Figures

List of Tables

1 Introduction

1.1 Context and Motivation

Uncontrolled fires have caused significant environmental and economic damages over the past few decades, especially in the Mediterranean region [39]. One of the most effective measures for forest fire prevention is fostering landscaping and forest management procedures [11], such as forest clearing, whereby unnecessary plant matter is reduced to small particulate in order to reduce its flammability. Clearing the forest is a harsh and dangerous task that routinely involves the usage of heavy machinery and specialised manpower, both of which command significant costs and are also increasingly difficult to source. These issues make forest clearing a very interesting task to automate, potentially reducing costs and increasing productivity. A key aspect to automating this task is enabling robots to navigate the forest safely and efficiently, so that they may perform their tasks in the forest autonomously.

Reliable autonomous outdoor navigation in forestry scenarios is still a significant scientific challenge. These environments can be very dynamic, with varying conditions such as weather, illumination, wind, etc, all of which contribute to hindering the robot's ability to navigate reliably [26]. Other issues include the unstructured environment that make it difficult perception and localisation [17], or the fact that estimating odometry may not be possible due to rough terrain and slippage [37]. Moreover, for a robot to navigate efficiently, it must possess information on its surroundings, which is hindered by the dynamic nature of the environment, thus it is necessary to find solutions to correctly perceive the surrounding environment.

The SEMFIRE R&D project[1] (Safety, Exploration, and Maintenance of Forests with the Integration of Ecological Robotics), to the scope of which this dissertation is inserted, using the Ranger (Fig. 1.1) as main actor, proposes the development of a robotic system to reduce fuel accumulation in forests. Its main goal will be to eliminate living flammable

[1] `http://semfire.ingeniarius.pt/`

Figure 1.1: The Ranger UGV being teleoperated in a realistic test in a forest environment. Its main sensors can be seen in its head, see Section 4.1.1 for more details.

material for wildfire prevention using a mulcher, thus assisting in landscape maintenance tasks, addressing in the process several challenges in forestry and field robotics [8]. The overall processing pipeline for decision-making of the Ranger platform can be seen in Fig. 1.2. Within the architecture shown in this figure, this dissertation will be contributing only to the **3D Safe Navigation** module.

1.2 Main Goals of this Dissertation

The main goal of this work is to propose navigation and local planning methods that allow a forestry robot to safely navigate to a target configuration while avoiding obstacles, taking into account our definition of mechanical effort. Specifically, we:

1. Conduct a state of the art study in navigation and path planning taking into account specific forest navigation problems;

2. Introduce a technique that uses a 3D point-cloud from the robot's sensors to estimate the cost of traversing each individual point in space, producing a costmap for navigation;

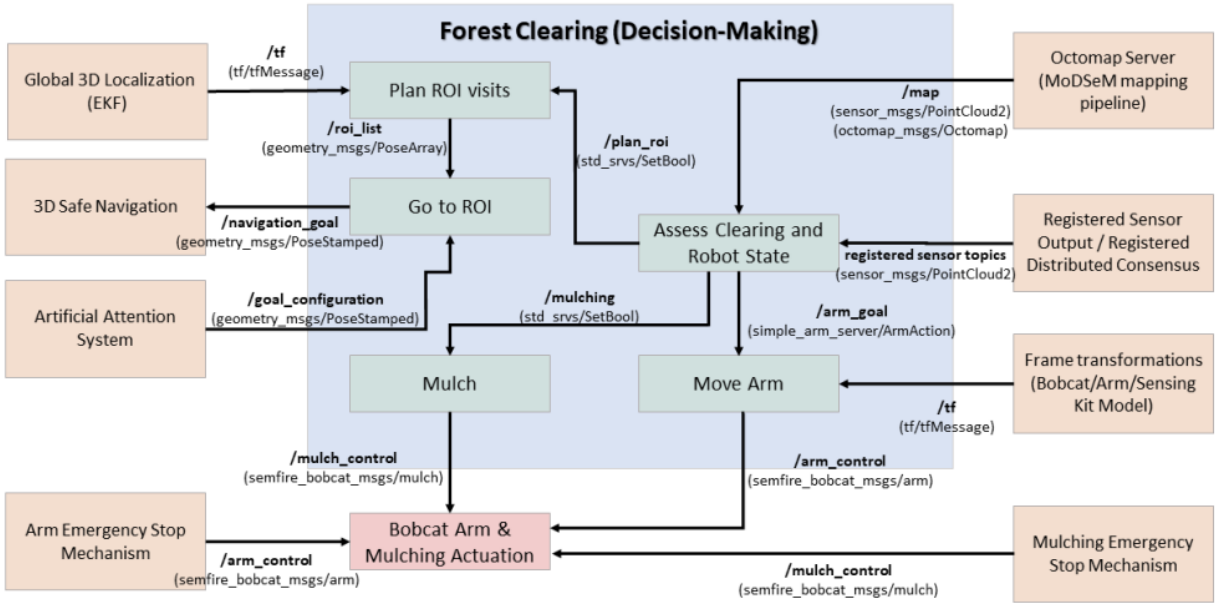


Figure 1.2: SEMFIRE decision-making pipeline for the Ranger, taken from [32].

3. Test the feasibility and applicability of existing local planning methods in forestry environments in light of the mechanical costs of traversal;

4. Demonstrate that existing techniques do not take mechanical cost into account, and can thus be refined to produce more economical trajectories;

5. Demonstrate that our approach chose the path with less mechanical effort.

Our tests were conducted on a forestry robot simulator, developed for the SEMFIRE project, that allows us to perform repetitive testing without the inherent costs of using the real large-scale rig, presented in Fig. 1.1.

1.3 Document Overview

This document is stuctured as follows. We start by performing a literature review in 3D navigation and path planning in Chapter 2. We then present our approach, explaining in detail the different steps to reach the final costmap that take into account the mechanical cost and the power consumption of the robot, in Chapter 3. Our approach is tested in Chapter 4, which presents the experimental setup and results. Lastly, in Chapter 5 we reflect upon the overall success of this work, proposing a number of potential lines of future work.

2 Background on 3D Navigation and Path Planning

2.1 Basic Concepts in Navigation

Autonomous navigation can be defined as a robot's ability to move in its environment without any external control by human operators, by planning and executing trajectories safely, both for itself, but also for humans and other entities that might be sharing its workspace [3]. Based on the kinematic constraints on movement that a robot is subject to, it can generally be classified as *holonomic or non-holonomic*. If a robot is holonomic with respect to N physical dimensions, it will be capable of moving in any direction in any of those dimensions, thereby approximating the ideal, unrestricted situation. On the other hand, if it is non-holonomic, it is restricted by its configuration in which directions it can move in. For example, a holonomic robot may be a drone (a quadcopter), and a non-holonomic may be a differential drive robot (like the Ranger).

Navigating safely and autonomously involves sensory and perceptual abilities for *obstacle detection and tracking*, enabling the system to circumvent entities in the environment that are hindering its navigation. As such, these systems can be equipped with sensing modalities such as time-of-flight cameras, laser-based 3D sensors [41], or 3D point cloud sensors, for instance to reconstruct a triangle mesh of the environment in real-time [33]. After the obstacle detection process, it is necessary to perform *path planning* that determines a collision-free path between start and goal positions in a workspace cluttered with obstacles. Path planning is usually a geometric matter because it aims to generate trajectories with no interest of the time of their execution [13]. In a complex environment, it becomes a dynamic planning process due to the flexibility and real-time nature of the conditions [55]. It is useful for the robot to keep a permanent recording of more than the immediate surroundings that its sensors can perceive at a specific moment in time - this information, drawing a human-like

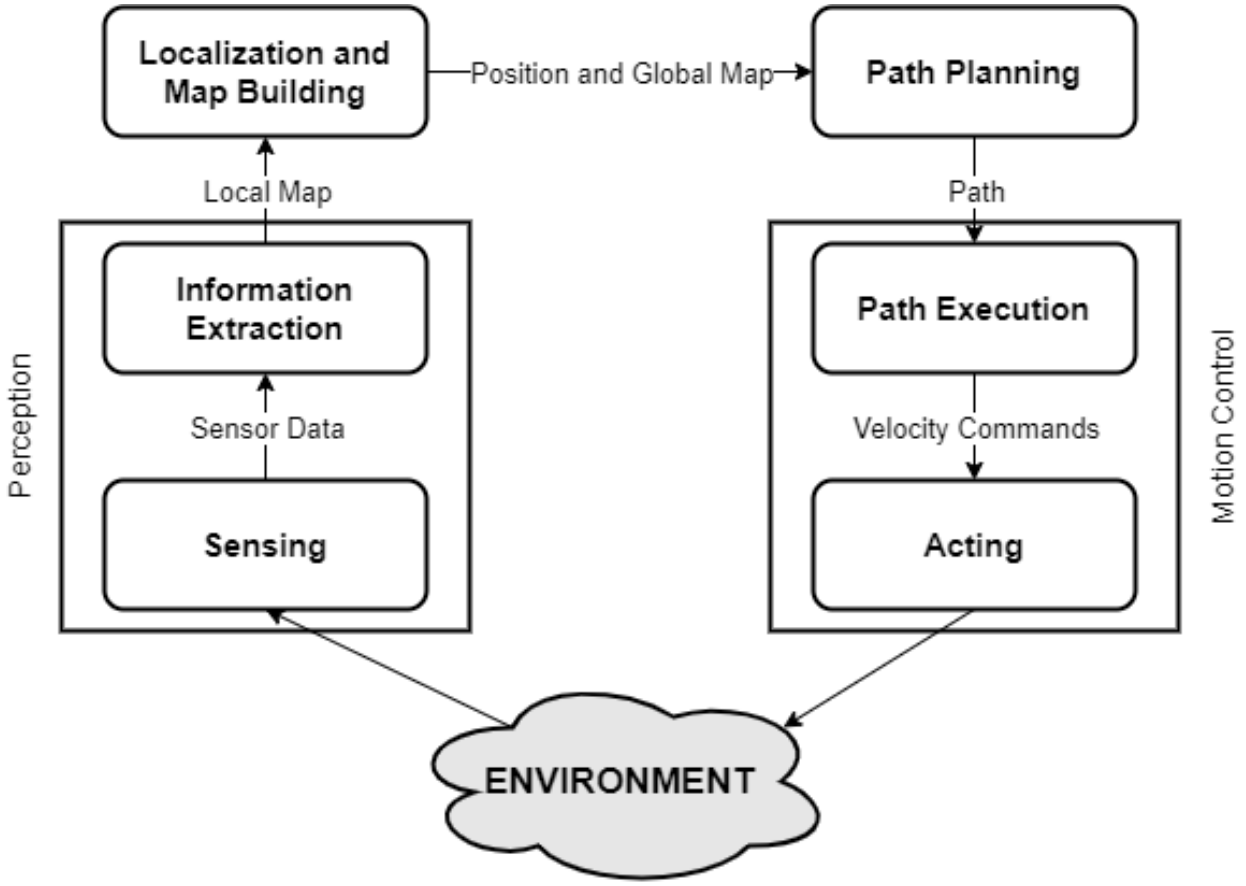


Figure 2.1: Flow diagram for a robot navigation. The sensors receive information from the environment that will be used to build a representation of the environment (map), which will be used by the robot control in order to decide its performance.

analogy, is said to compose a *map*. A map is a description of the environment which can be used for navigation and usually represents the static elements of the environment.

One approach to obtain a map is to perform *Simultaneous Localization and Mapping (SLAM)* [50] [22], an approach that allows the generation of a map of the robot's surroundings while estimating its location in that map at the same time. In [31], full 3D localization is performed by adapting an existing 3D SLAM algorithm, referred to as C-SLAM, to create the base map. In [26], authors considered the use of trunks as landmarks for localization using a sensor tilted upwards. However, one of the drawbacks of this technique was that the presence of people considerably increases the complexity of tree trunks extraction. Pose estimation performed by SLAM approach proposed by the authors can correct dead reckoning errors and track a single hypothesis of the vehicle pose, with little computational effort. Finally, in [46] the authors use two 2D LIDARs to develop a 2.5D SLAM based on the FastSLAM algorithm, where both LIDARs are used to build a joint occupancy grid map.

A robot may be semi-autonomous (i.e. remotely-controlled) or fully autonomous. There are several types of autonomous, unmanned vehicles, including:

- Unmanned Aerial Vehicle (UAV), (commonly known as a drone) [40] [18];
- Unmanned Water Vehicle (UWV), a type of vehicle can be further subdivided in:
  - Unmanned Surface Vehicle (USV) if it operates on the surface of the water;
  - Unmanned Underwater Vehicle (UUV) if it operates underwater;
- Unmanned Ground Vehicle (UGV) [31].

The type of terrain (air, water, ground) the vehicle operates in presents its own challenges. Underwater vehicles may have more favorable buoyancy and gravitational environments, at the expense of many problems with the communication. The dynamics associated with underwater motion are considered more complex and non-linear compared with other types. Aerial and, to a lesser extent the ground vehicles, must deal with gravity forces, but the physics of their interaction with the environment are much better understood. Prior work usually focuses on ground vehicles [42]. The robot used in the work described in this dissertation is a non-holonomic UGV (Unmanned Ground Vehicle) operating in a forestry application.

One of the key enabling features of an autonomous UGV is the ability to detect navigable areas, can reliably classify traversable ground in the environment. This topic is commonly referred to as *traversability analysis*. The quality of the traversabily analysis affects the free movement of the platform and how the robot is able to plan a path safe from collisions during the navigation. In 2D indoor environments traversability analysis is simply classified based on the observed obstacles. However in unstructured environments, where the ground is not flat and obstacles are not purely vertical, for traversability analysis specialized sensors are used to perceive a dense model of the world [49].

Historically, traversability estimation was addressed as a binary classification problem, distinguishing traversable from non-traversable areas. More recently, the need for a more accurate classification was recognized that assigned a continuous traversability capacity score or classified the terrain into the various classes that were commonly found in a specific application [29].

2.2 2D Navigation and Path Planning

Safe navigation and path planning is one of the most important functions of any mobile robot, being one of the most researched topics in Robotics nowadays. Global navigation

can be described as the ability to estimate the position of entities (e.g. objects, people, etc.) in the environment in function to a reference axis, and to steer to a previously decided goal; local navigation identifies the dynamic conditions of the environment and establishes positional relationships between various elements [30].

A generic path planning algorithm needs to take into account that [28]:

- the chosen path must have the lowest cost to prevent misdirection;
- the chosen path must be acquired quickly;
- the algorithm must be generic with respect to different maps, meaning that it should not be optimized for a specific map type.

There are several path planning algorithms, each adapted to different characteristics of the environment in which the robot is navigating in. Classical approaches [25] include the Visibility Graph, where lines are used to connect features to denote visibility; the Voronoi Diagram, which partitions space into cells, each of which consisting of points closer to one particular object than any other; the Cell Decomposition, with which free space is decomposed into a set of simple cells, and the adjacent relationships among the cells are computed; Artificial Potential Fields, wherein a robot is treated as a particle under the influence of an artificial potential field whose local variations reflect the “structure" of the free space.

The classic methods mentioned above suffer from a substantial number of disadvantages, such as time complexity in high dimensions, and getting trapped in local minima, which makes them inefficient in practice. For some reason, probabilistic approaches have been developed in order to improve the efficiency of classic methods, such as, for example, Probabilistic road maps and Rapidly-exploring Random Trees (RRT) [20].

Path planners can be divided into global and local approaches. Global path planners are based on a static map, creating a path from the start to a goal position. Local path planners, on the other hand, consider the global planner goals, and create local waypoints, taking into consideration the dynamic obstacles on the way and also vehicle constraints. Using local planning local map is reduced to the surroundings of the vehicle and is updated as the vehicle is navigating [9]. In Table 2.1, we can see the main differences between the two different approaches for path planners.

As mentioned above, the global planner requires a map of the environment to calculate the best route. Depending on the analysis of the map, some methods are based on road maps, such as Silhouette [5] or Voronoi [2]. Some approaches solve the problem by assigning

Table 2.1: Differences between Global and Local Path Planners (Based on [9]).

Global Path Planner	Local Path Planner
Map based	Sensor based system
Deliberative system	Reactive system
Relatively slower response	Fast response
Complete knowledge of the workspace area	Incomplete knowledge of the workspace area
Obtain a feasible path leading to goal	Follow path to the target while avoiding obstacles

a value to each region of the road map to find the path with the minimum cost, like Dijkstra [44] algorithm, Best First [10], and A* [14]. Other approaches use potential fields as described in [52], a classical technique developed decades ago, Rapidly Exploring Random Trees (RRT) [19], or the approach based on Neural Networks [54].

Local planning generates short-term control that tries to match the trajectory as much as possible to the waypoints provided by the global planner. There are some examples such as Dynamic Window Approach (DWA) proposed in [12], and Rollout Trajectory Planner (RTP) [15], which both discretely sample in the robot's control space and perform a forward simulation to evaluate each trajectory, scoring each one and discarding illegal ones. However, RTP samples from a set of achievable velocities over the entire forward simulation period, while DWA only samples for one simulation step.

Another example is Elastic Band (EBand) [35], which does not necessarily sample the robot's control space, instead it produces a collision-free path based on key features called "contraction force" and "repulsion force", where the former is responsible for minimising the length of path while the latter one pushes the robot away from possible obstacles. Another approach is the Timed Elastic Band (TEB) [38], an optimization of EBand that takes into account temporal aspects of motion in terms of dynamic constraints such as limited robot velocities and accelerations.

2.3 Navigation and Path Planning in 3D Environments

Planning feasible paths in three-dimensional environments is a challenging problem [37]. Existing algorithms typically use limited 3D representations that discard potentially useful information [47]. According to [53], 3D path planning algorithms can be divided into five categories distinguished from each other by their unique properties. The division proposed with a brief description of each method is illustrated in Figure 2.2.

Sample-based methods are widely used because of their effectiveness and low computational cost for high-dimensional spaces. They sample the environment as a set of nodes, cells,

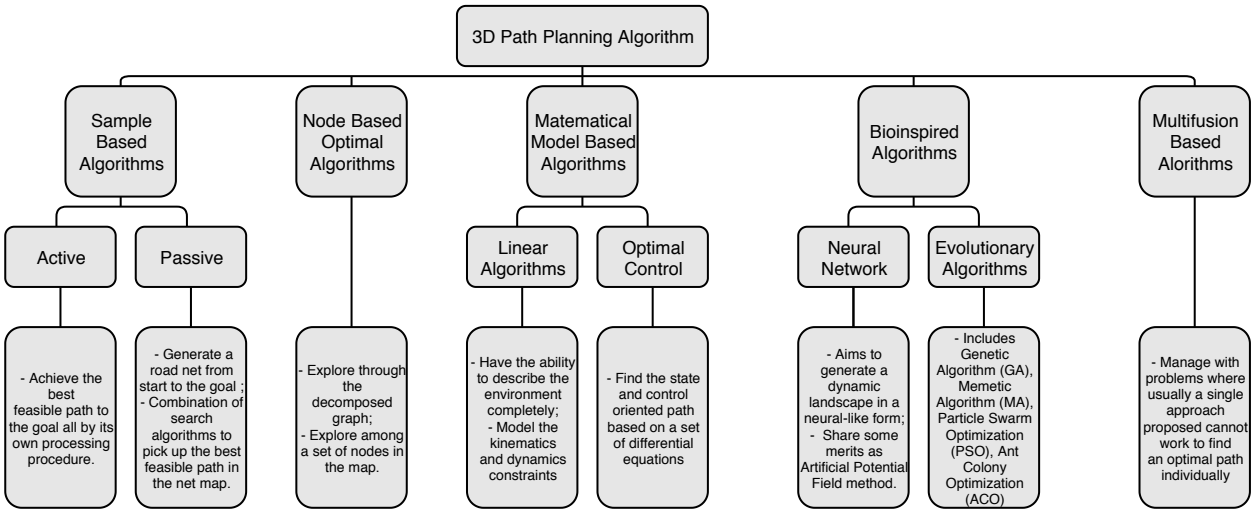


Figure 2.2: Categories of 3D path planning algorithms (Image based on [53]).

or other means of subdivision of space, and then map the environment or search randomly to achieve a feasible path. This type of algorithm can be divided into two categories: active, for which algorithms achieve the best feasible path to the goal through their own processing procedure, and passive, wherein the algorithm generates a road net from the starting point to the goal through a combination of search algorithms to pick up the best feasible path in net maps where many feasible paths exist.

The RRT (Rapidly-exploring Random Trees) [1] [53] are an example of an active sample-based method, and has been widely used for fast trajectory search because of its incremental nature, with several different versions of RRT developed to improve the cost of the solution path. Other examples include methods based on the Artificial Potential Field, i.e. forces of attraction and repulsion, like the Vector Field Histogram (VFH) that uses a statistical representation of the robot's environment through a histogram grid [26] being considered collision avoidance. Passive methods include PRM (Probabilistic Road Maps), that consists of taking random samples from the configuration space of the robot, testing them for whether they are in the free space, and use a local planner to attempt to connect these configurations to other nearby configurations [18].

Node based algorithms share the same property in that they explore among a set of nodes in the map where information sensing and processing procedures are already executed. This type of planers can find an optimal path according to the certain decomposition. Some examples are A* [41], D* [4], and Dijkstra's [31] [33], which are widely used for global planning, assuming that a global (or at least partial) representation of the environment exists.

Mathematics-models based technique include linear algorithms and optimal control. They

model the environment as well as the system and then bound the cost function with all the kinematic and dynamic constraints which are inequalities or equations to achieve an optimal solution. Finally, bioinspired algorithms originate from mimicking biological behavior to solve problems. One example would be neural networks [16].

In the literature, we can find several works on 3D navigation not just for ground robots, but also aerial robots [18] [40], using local planners like the B-spline approach, Elastic Band, respectively, and underwater robots [51], that uses the Dynamic Window Approach.

In [45] the authors represent the traversability map in a `2D Grid` where each cell gives the probability that the vehicle can successfully drive over. The authors of [33] use a 3D representation of a estimation of distances, height differences, and roughness, called `navigation_mesh`, to see if a area is traversable or not, testing this approach with a local planner that follows the approach of [48] to extract the vector field that directly correspond to the potential field. In [41], the authors present two techniques to detect and classify traversable areas in 3D environments: the first is a low-level mechanism aimed to detect obstacles and holes using a time-of-flight camera; the second is a high-level classification mechanism that detect traversable regions from 3D point clouds. They tested this work integrating their system with the Robot Operating Systems (ROS), using `base_local_planner` with the Rollout Trajectory Planner (RTP) and `move_base` as the low-level controller. In [7], the authors, from a given image that represent the height map of a terrain build a convolutional neural network to predict whether the robot will be able to traverse some path and in which direction. This particular methods take into consideration the fact that some points of the terrain might be traversable only in specific directions and by a specific robot, depending on the robot power.

In [27], the authors incorporate existing ROS-based algorithms for localization, mapping, traversability, navigation and exploration in unknown and unstructured environments. The authors used the `traversability_estimation` package that requires an elevation map to estimate the traversable spaces. To test their approach, they used the A* as a global planner in the traversability costmap and the Timed Elastic Band (TEB) as the local path planner. This is the current system used by the the SEMFIRE project for Ranger navigation.

There are also some works that take into consideration the energy spent by the robot, trying to minimize that energy in each path. The work presented in [24] performs a traversability analysis online from data sensed by the robot and assumes that the power consumed in navigation has two terms: a part which is a function of the path, the velocity, slope and traversability; and a part that results from the operation of on-board computers, sensors

and communication gear (the so-called “hotel load"). Performing several tests, using the D* algorithm [4] as path planner, the authors aim that the estimated map converges to the true map, optimistically assuming that the terrain has low traversability, however if the cost due to “hotel load" and the estimated travel cost does not provide a reduced energy cost the robot will not explore those particular areas.

Table 2.2 presents a comparison between some of the most used local path planners, taking into account the state of the art done.

2.4 Navigation in ROS

There are several software packages used for robot navigation, such as, Orocos[1] or works like [26] that use a software framework called Sensor Sharing Manager (SSM). As alternative, ROS, the Robot Operating System, is becoming increasingly popular and highly regarded. Because the SEMFIRE project use this framework and the packages we want to test are there, this will be the used framework.

ROS is a flexible framework for writing robot software. It includes a set of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms [6]. The key feature of ROS is the way the software runs and how it communicates, allowing the design of complex software while abstracting away from the hardware.

ROS provides a way to connect a network of processes (nodes), which can be run on multiples devices, allowing the exchange of messages [34]. It implements a standardised inter-process communication framework that leverages a uniform message passing system to provide abstraction from low-level communication details. It defines and implements two main message passing paradigms: publisher-subscriber, whereby nodes subscribe to topics that propagate information in a multicast fashion; and server-client, wherein a node requests a service from another and blocks until it is fulfilled.

This framework is organized in packages, which generally contain code, message types and other support files. It includes off-the-shelf packages that implement navigation and path planning.

[1] `https://orocos.org/`

Table 2.2: Comparison table of different local path planners

Path Planner	References	Operating Principles	ROS Package	Outdoor Environments	Tested in Forest Environments	Metrics Used
Vector Field Histogram (VFH)	[26]	Artificial forces of attraction and repulsion	No	Yes	No	- Success Rate - Distance from goal point
Dynamic Window Approach (DWA)	[51], [9]	Samples from the set of achievable velocities for just one simulation step given the acceleration limits of the robot	dwa_local_planner and base_local_planner	Yes	No	- Smoothness - Minimum distance from goal point - Maximum distance from goal point
Rollout Trajectory Planner (RTP)	[41]	Samples from the set of achievable velocities over the entire forward simulation period given the acceleration limits of the robot	base_local_planner	Yes	Yes	- Precision in traversability segmentation
Elastic Band (EBand)	[21], [9]	A collision-free path is generated using a “contraction force” and a “repulsion force”	eband_local_planner	Yes	No	- Smoothness - Minimum distance from goal point - Maximum distance from goal point
Timed Elastic Band (TEB)	[9], [27]	An optimization of EBand, considering the temporal aspects of the motion	teb_local_planner	Yes	Yes	- Success Rate - Planning Time - Smoothness

ROS provides a popular navigation software as a collection of nodes, which is used worldwide in distinct mobile robots, commonly known as `Navigation Stack`[2]. The role of this software is to process data from odometry, sensors and the environment map in order to find a safe path for the robot to execute, and give the robot the actuation commands [56]. The `move_base` package (Fig. 2.4) contains the high-level interface to the `Navigation Stack`, providing an implementation of an action that, given a goal, will attempt to reach it. The `move_base` node links both global and local planner, supporting different global planner algorithms such as the `carrot_planner`[3], `navfn`[4] and `global_planner`[5]. In the context of this work we focus on local planners, such as `base_local_planner`[6], the `teb_local_planner`[7], the `eband_local_planner`[8], and the `dwa_local_planner`[9]. The `Navigation Stack` software was initially created for 2D environments in order to allow a robot to complete a certain distances of autonomous navigation in a real office environment [23]. Since then it has been widely used and supported by the ROS community, configured to the characteristics of each robot and environments [36]. Some existing works, such as [56], in order to help to understand the `Navigation Stack` parametrization, present a guide for beginners, explaining how to best parametrize the global and local planners, as well as the costmaps.

Existing ROS packages also tackle the problem of traversability, and possibly navigate, such as `mesh_navigation`[10] or `traversability_estimation`[11] package, which uses elevation maps and traversability estimation filters to generate a traversability map. This packages has been integrated with the `Navigation Stack` in order to extend this software to 3D environments. As presented in the Section 2.3, [33] integrated the `mesh_navigation` with `Navigation Stack` in order to allow a robot to navigate in a 3D outdoor environment. The `traversability_estimation` have been used in the current SEMFIRE navigation system [27], allowing the robot to navigate in complex forestry environments.

In addition, ROS has tools to visualize the robot's world, `rviz` [12], a 3D visualization tool for ROS applications. It provides insight into the robot model, captures information from the robot sensors and reproduces the captured data. It can display cameras, lasers,

[2] `http://wiki.ros.org/navigation`
[3] `http://wiki.ros.org/carrot_planner`
[4] `http://wiki.ros.org/navfn`
[5] `http://wiki.ros.org/global_planner`
[6] `http://wiki.ros.org/base_local_planner`
[7] `http://wiki.ros.org/teb_local_planner`
[8] `http://wiki.ros.org/eband_local_planner`
[9] `http://wiki.ros.org/dwa_local_planner`
[10] `https://github.com/uos/mesh_navigation`
[11] `https://github.com/leggedrobotics/traversability_estimation`
[12] `http://wiki.ros.org/rviz`

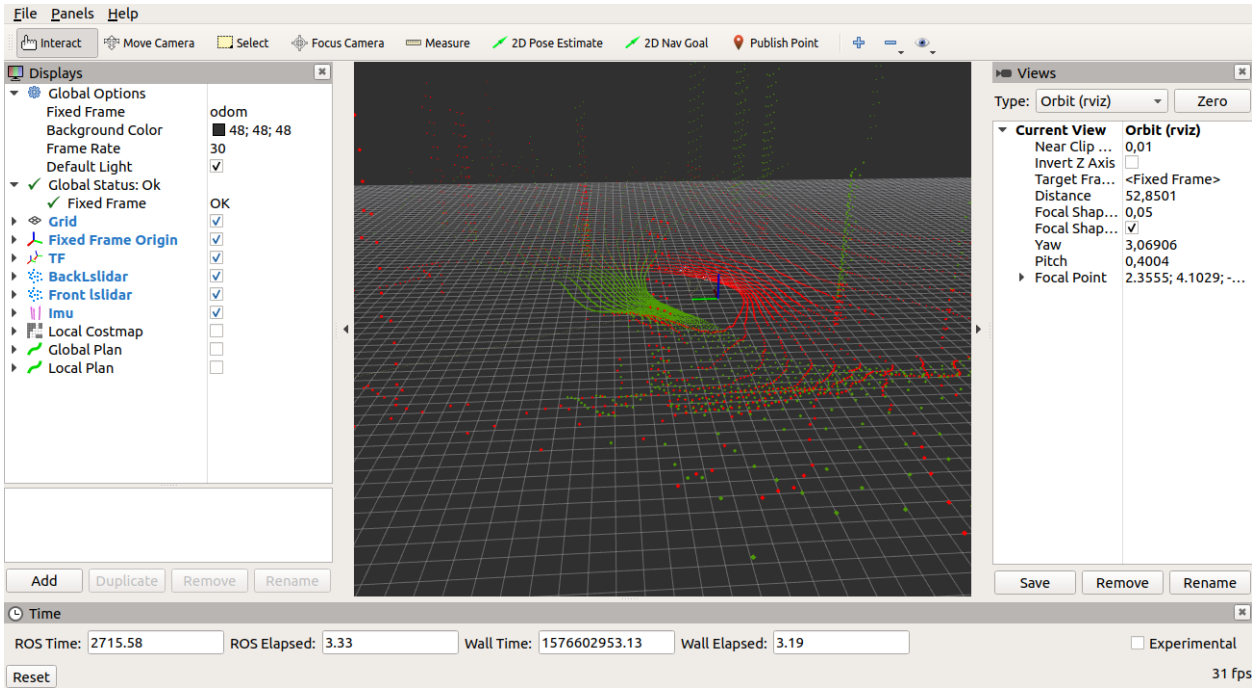


Figure 2.3: Ranger's LIDARs point cloud in simulation.

2D and 3D device data, including images and point clouds. In Figure 2.3 an example of visualization of the Ranger's back (red dots) and front (green dots) LIDARs, in the dense forest simulation scenario is illustrated.

2.5 Main Contributions

As mentioned in Chapter 1, we aim to allow a ground robot to safely navigate, focusing in local path planning, in a 3D forestry environment, who has many dynamic conditions already mentioned in Section 1.1.

Through Section 2.3 several techniques are discussed, which are able to successfully navigate in 3D scenarios, the state of the art shows that no technique seems to explicitly take the mechanical effort into account when planning the trajectories.

> ***Mechanical effort*** is the quantity that accounts the additional burden of the UGV when climbing hills in rough terrain environments.

We build on existing local planning methods to propose an algorithm that integrates the cost of the mechanical effort in a `2D Grid` to be considered for local planning, potentially allowing a robot to safely navigate in a 3D forestry environment. In Fig. 2.4, the blue blocks represent the focus of this work. Taking into account the goals defined in Section 1.2, specifically we contribute by:

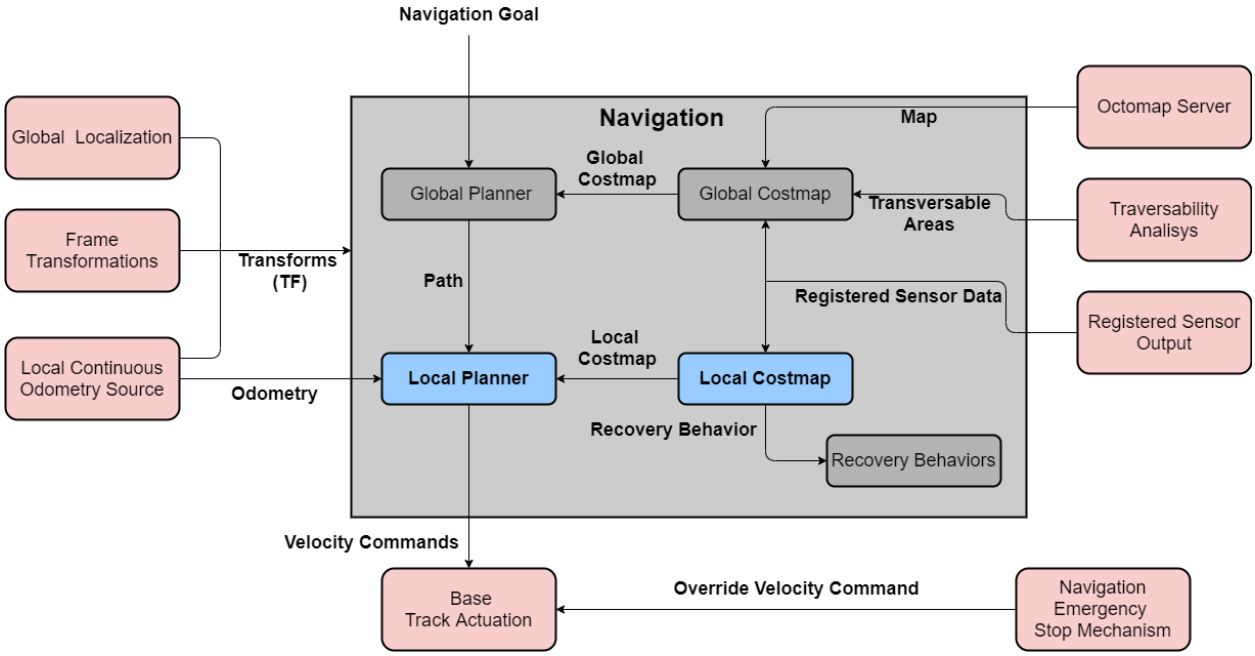


Figure 2.4: `move_base` operating diagram. Two planning levels are implemented: local planning using a local costmap, and global planning using a global costmap. These are informed by sensor data and a mapping approach, respectively. The main contribution of this work will be the blue blocks.

- Introducing a technique that uses a 3D point cloud from the robot's sensors to create a 2D grid based on the mechanical effort, producing a local costmap for navigation, an approach that no other work has taken, to the best of our knowledge;
- Testing and comparing our technique with existing ROS systems, to prove that it can produce more economical trajectories;
- Integrating the produced map with the `ROS Navigation Stack` so that the local planner takes into account the mechanical effort when choosing the paths, thus making our contributions widely available to the community.

3 Adding Mechanical Effort Awareness to the Navigation Stack

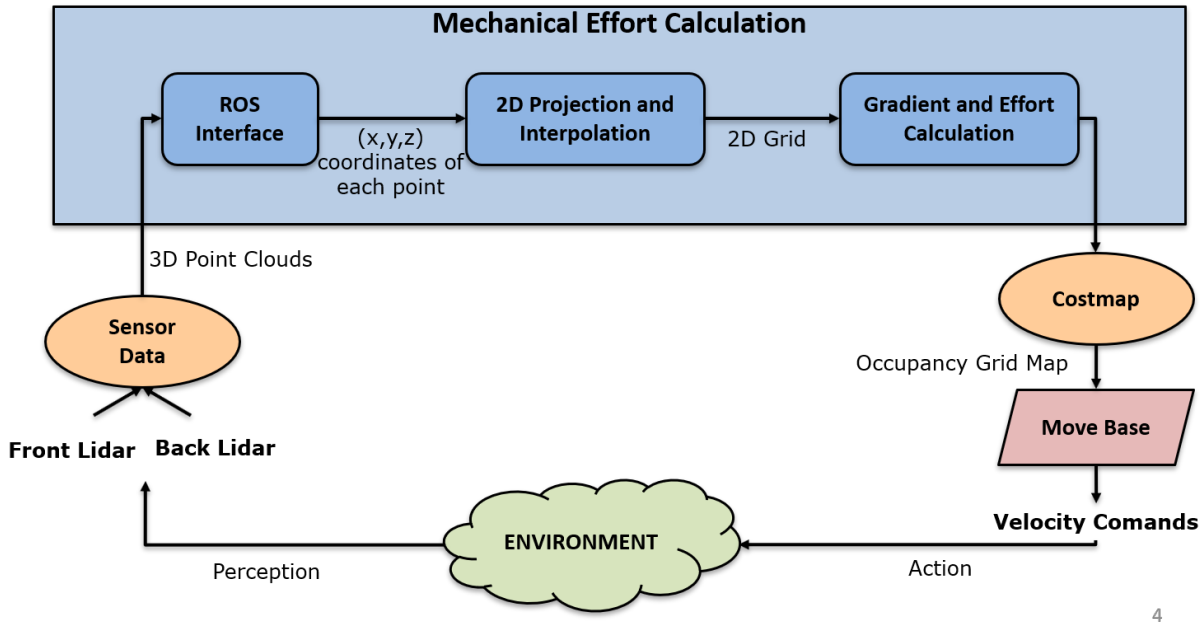


Figure 3.1: Flow diagram of the implemented algorithm to estimate the cost of traversing each individual point in space.

3.1 Overview

Most of the existing techniques do not take into account the effort that quantifies the additional burden of the UGV when climbing hills in rough terrain environments. Throughout this chapter, we will describe the solution to estimate the cost of traversing each individual point in space, producing a costmap for navigation (summarized in Fig. 3.1) and the necessary changes to integrate this map with the ROS Navigation Stack.

The focus of this works is the Fig. 3.1 blue box (Mechanical Effort Analysis), divided in three main blocks that uses 3D Point Clouds from the robot sensors to obtain a *Occupancy*

Grid Map, to use as a local costmap for robots navigation, where each cell give the mechanical effort necessary for the robot navigate. The mechanical effort of each cell is given with the relation of the gradient and the effort concept (explained in Section 3.2.4). The map is sent to a ROS topic, used by `move_base`. The local planner will subscribe that topic in order to, when its given a goal, the local path planner take into that cost.

This system is user-configurable in all relevant dimensions:

- Size of the map, in meters;
- Resolution of the map, in meters/cell;
- The option to choose if the user want a map based on gradient, or the effort concept, or in both, the last one ideal.

Our map contains information that `move_base` would not normally take into account, requiring that we extend it in a few specific ways. The default costmaps used by `move_base` are given by the package `costmap_2d`[1], despite the fact each cell in the costmap can have 255 different cost values, this package only represent three: free, occupied or unknown. Our resulted map uses all cost values to represent, in occupancy probabilities, <50 less effort (downward slopes) , =50 neutral effort (plane) and >50 bigger effort (upward slope), i.e., we want to prioritize descents and straights. So to integrate our map with the ROS navigation stack, we will:

- Subscribe to our map topic in the `move_base` package;
- Change the `base_local_planner` to take into account our map in the time to chose the trajectory;
- Create a weight to control how much the planner should take into account our cost.

3.2 Mechanical Effort Analysis Node

3.2.1 ROS Interface

The first sub-component of this node is the ROS Interface, this class takes data from the front and back 3D LIDAR Point Clouds available in the Ranger UGV. The final map will be built with the position that comes from odometry as center, so the points of each LIDAR

[1] `http://wiki.ros.org/costmap_2d`

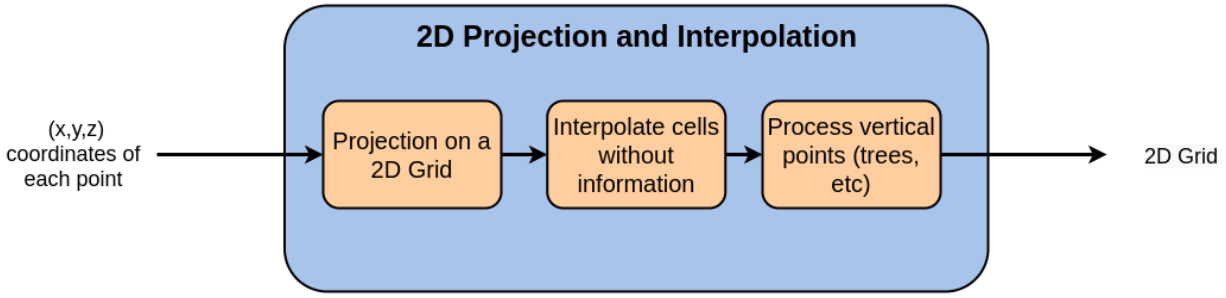


Figure 3.2: Flow diagram of the three main steps of the block 2D Projection and Interpolation of Fig. 3.1. Receiving the 3D data we do a 2D projection in a 2D grid, interpolating the cells without information.

are transformed from their respective frame into the odometry frame, converted in x, y, z coordinates and concatenated in a single matrix. The output matrix (P) is $N \times 3$, where each line represent one point of the LIDAR and the 3 colunms represent the x, y, z coordinates:

$$P = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix} \tag{3.1}$$

3.2.2 2D Projection

We used the ROS Navigation Stack that traditionally operates in 2D, which is a disadvantage when it is necessary to plan for a 3D environment. To account for this the points from P had to be 2D projected. This process is divided in three main steps, shown in Fig. 3.2. The main output of the technique is a spatial grid that contains the z information in that place.

The first step to computing the 2D costmap is to project the 3D LIDAR points into a 2D grid, so that information was represented by:

$$\Phi(x, y) = z, \tag{3.2}$$

where $\Phi(x, y)$ is one cell of the grid, x and y the index of the cell, and z the height information of the cell. In this step, we take into account the size and resolution of the map, and project each LIDAR point into its corresponding cell is in or if the point of the LIDAR is out of the map limits, not considering these points for the creation of the map.

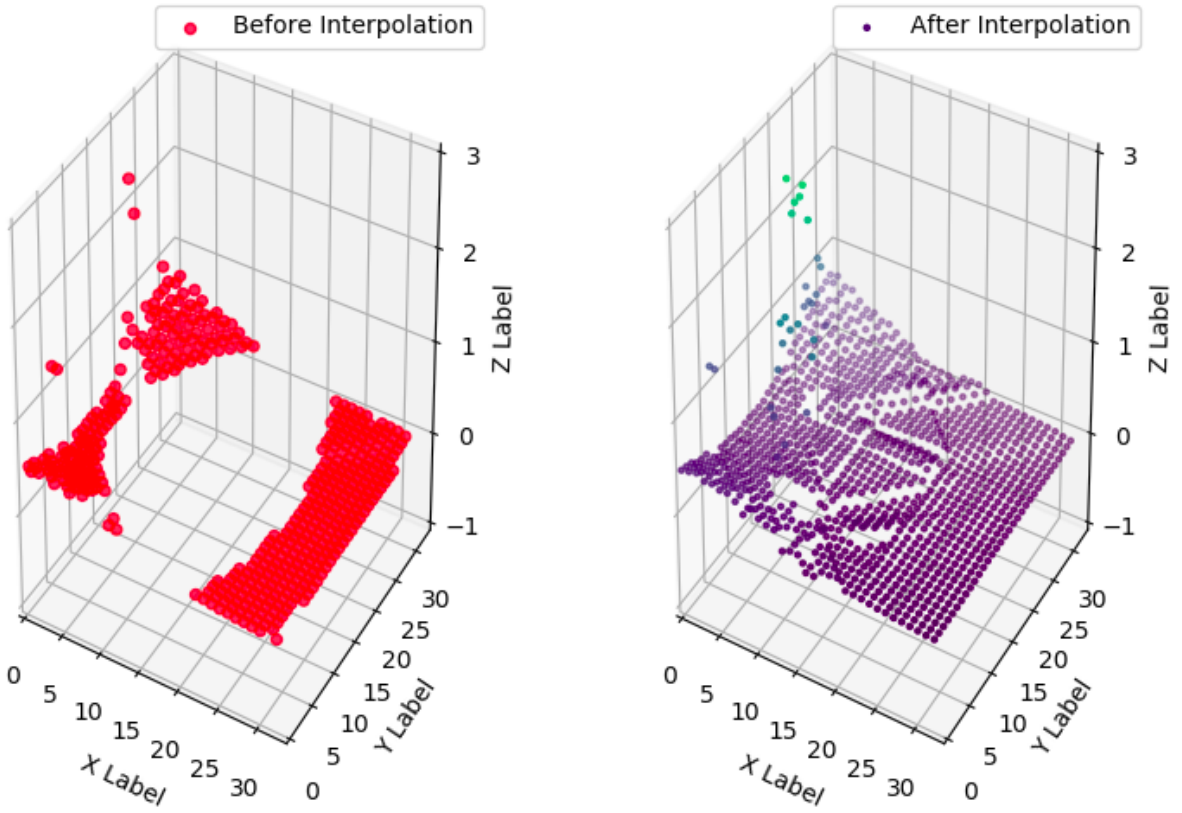


Figure 3.3: 3D representation of the LIDAR information projected in a 2D grid with dimension 10 meters and resolution 0,3 meters/cell before and after the interpolation.

Due to their discrete nature, each cell may contain more than one LIDAR point. In these cases, the median is used as a representative value for the whole cell. This results in a plot similar to the left picture of Fig. 3.3.

3.2.3 Interpolating Cells Without Information and Processing Vertical Points

The previous procedure results in many empty cells, like we see in the left graph of Fig. 3.3, so it is necessary to perform interpolation to obtain an estimation of a value for those empty cells. Our approach is to apply an adaptation of the natural neighbour interpolation [43], by analysing the neighbour cells in row, column and diagonals with information and apply the median, as before, to estimate the cell value. The algorithm deals with missing data, for instance in rows, by analysing if the cell next to it has no value, it runs through the entire row until it finds a cell with information.

Strong discontinuities in the resulting map cause undesirable artifacts when interpolated. These highly discontinuous points are part of the relief that we are not interested in (the

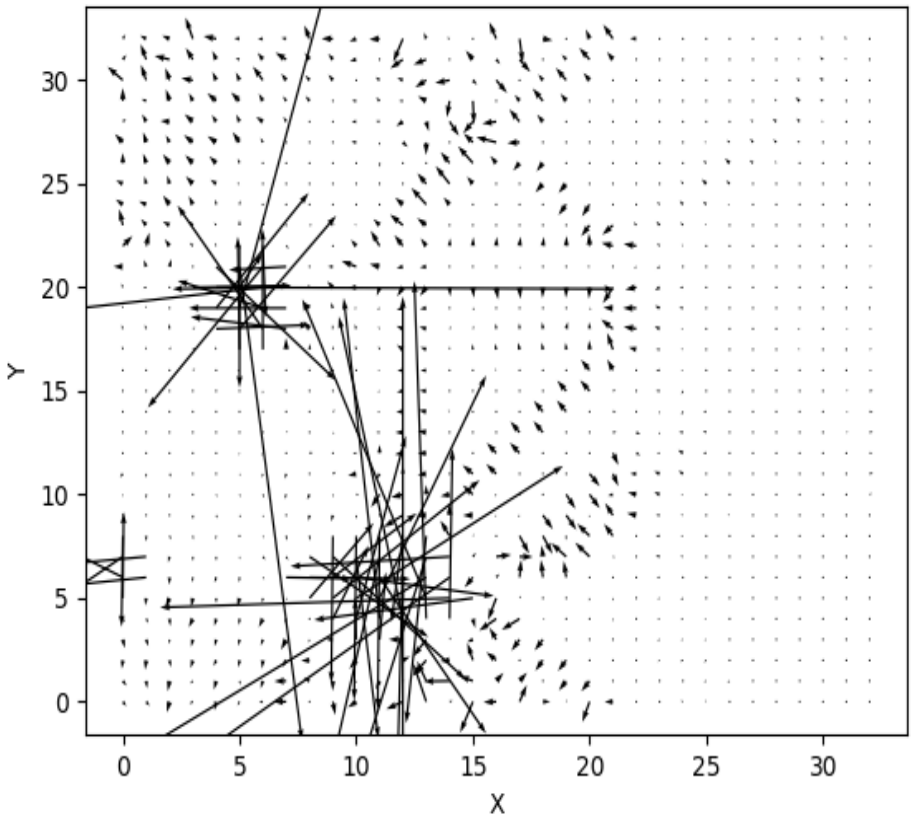


Figure 3.4: Gradient arrows of the environment. We can observe that large discontinuities in verticality generate uneven noise that is not representative of the true gradient of the terrain. These should instead be dealt with by an obstacle detection technique.

detection of obstacles takes care of them), and that spoil the smoothing that we want to remove from the interpolation. Thus, before applying the median of all neighbors, we have opted to filter out these points from the interpolation procedure according to:

$$|z_i - \overline{z}| < \rho, \tag{3.3}$$

where z_i is the height estimation of each of the neighbours of the current cell, $\overline{z}$ is the weighted average of all the neighbours, and the ρ threshold has been empirically adjusted to 2 m allowing for appropriate smoothing of the interpolation. The weighted average ($\overline{z}$) is computed as:

$$\overline{z} = \frac{\sum_i^n w_i z_i}{\sum_i^n w_i}, \tag{3.4}$$

being n the total number of neighbours and w_i the weights of each neighbour given by the inverse of the euclidean distance, because, like it was explained above, some neighbours are

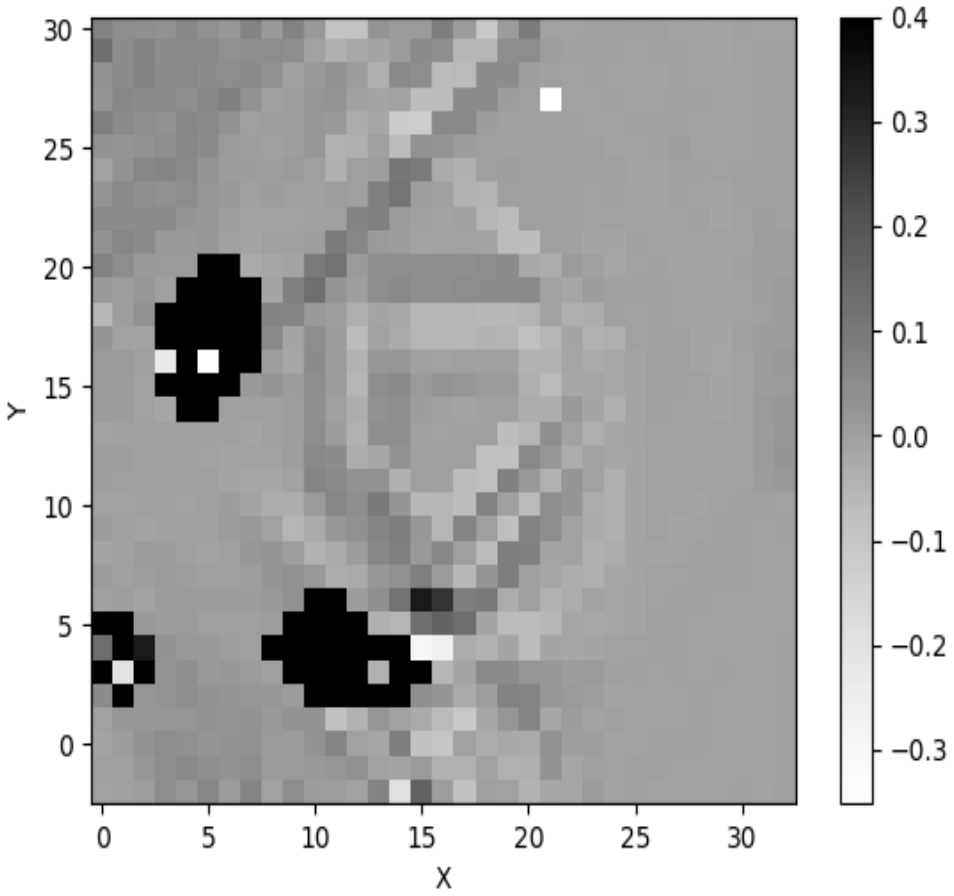


Figure 3.5: Resulting costmap based on the gradient and effort. We note that the discontinuities in Fig. 3.4 are considered in this graph as obstacles (black cells), the remaining grey graduations occur as explained in Section 3.1.

not the cell right after, so the further cell the less impact it has in the average. Only the neighbours of a certain cell that fulfill this condition are included in the calculation. An example of a result of the interpolation can be seen on the right of Fig. 3.3.

3.2.4 Gradient and Effort Calculation

With the map complete enough for our purposes, we calculated the gradient (Eq. 3.5) on each cell of the 2D grid to obtain the inclination in each direction, Fig. 3.4:

$$\nabla z(x,y) =< \frac{\partial z}{\partial x}(x,y), \frac{\partial z}{\partial y}(x,y) > \tag{3.5}$$

In order to build a costmap of the surrounding environment, all cells with gradient norm above a certain threshold, empirically chosen as 0.8, are automatically considered as non-traversable; to other cells, we apply the product between the gradient norm and the γ, i.e.,

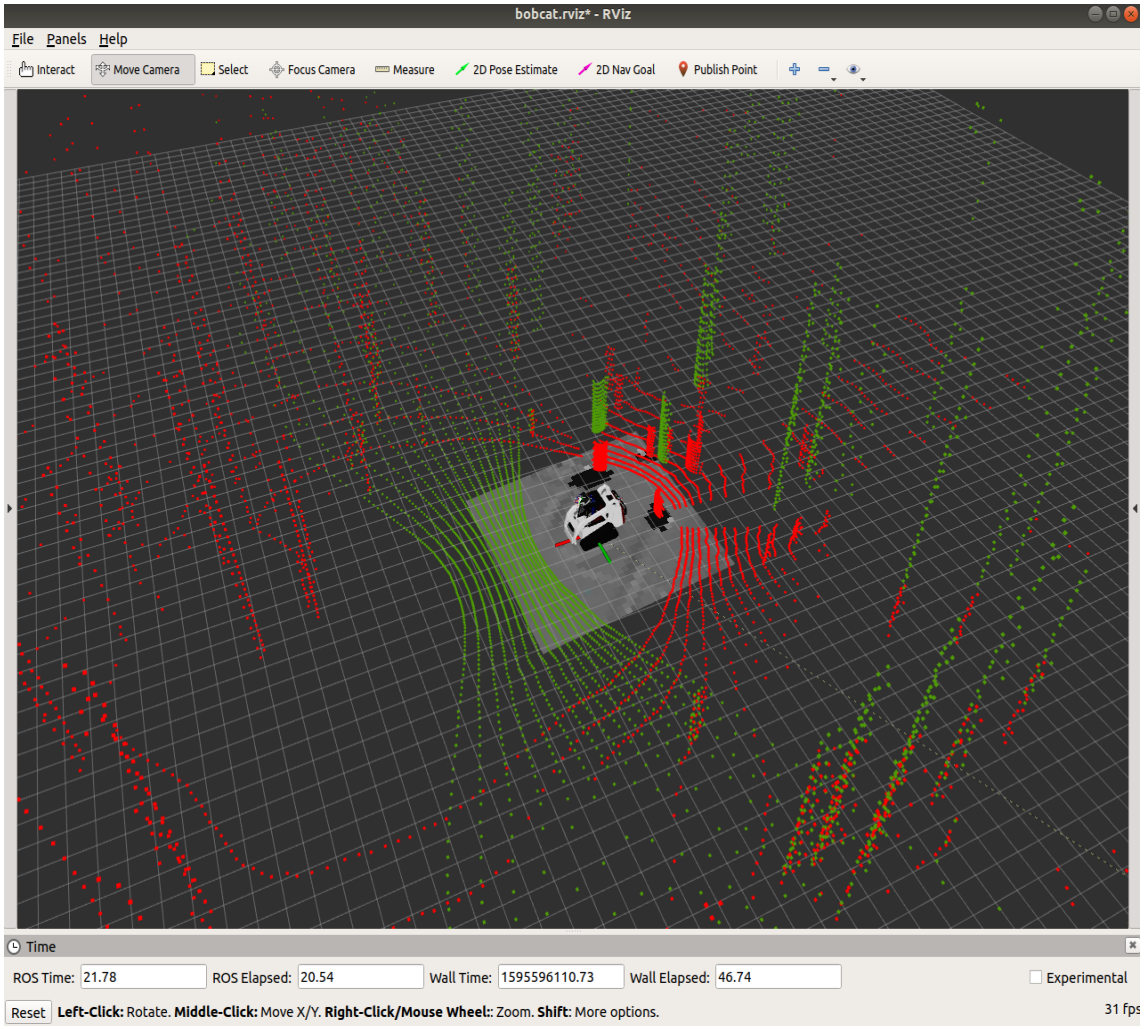


Figure 3.6: Simulated environment in `rviz`, with the information that the sensors perceive from robot surroundings and the resulting `Occupancy Grid` map

the effort (ξ):

$$\xi_{i,j} = \|\nabla z(x, y)\| \cdot \gamma, \tag{3.6}$$

where i and j are the x and y indexes, respectively, of a certain cell. We use γ to distinguish upward from downward slopes and is defined as:

$$\gamma = \cos(\theta), \tag{3.7}$$

where θ is the angle between the two vectors: the vector that connects the position of the robot with the position of the cell we are currently analyzing, and the gradient vector in that cell. Because the gradient direction is the direction in which the function increases most quickly, when the effort is:

- **Maximum** ($\gamma = 1$) we are facing an upward slope;

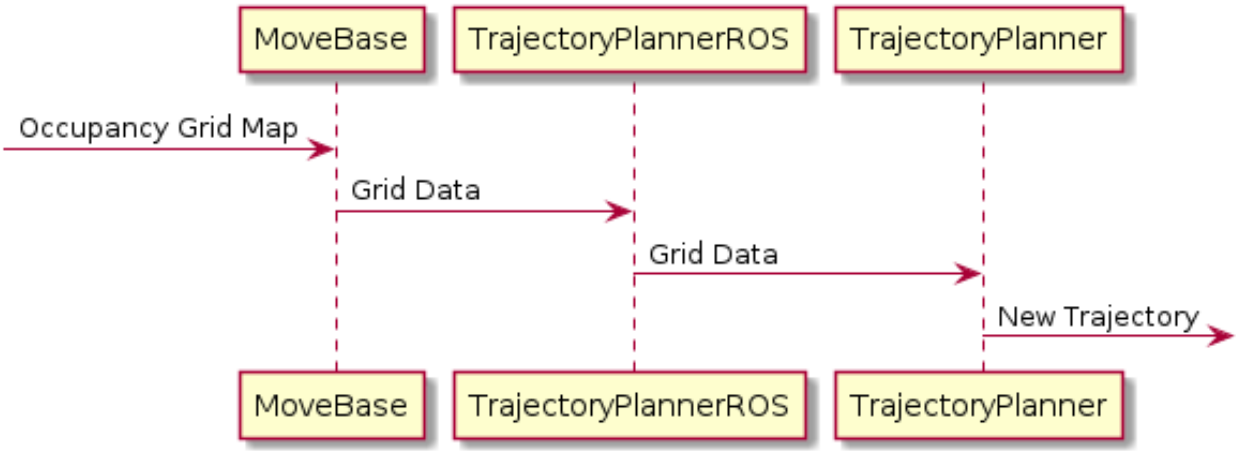


Figure 3.7: Sequence diagram of the changed classes for the integration of our method with `base_local_planner` of the Navigation Stack

- **Neutral** ($\gamma = 0$) we are on a flat terrain;
- **Minimum** ($\gamma = -1$) we are facing a downward slope.

Thus, we obtain a `2D Occupancy Grid Map` as in Fig. 3.5, where the black cells correspond to the non-traversable areas, and the shades of gray correspond to the probabilities explained in Section 3.1. This map is sent to a new ROS topic, to be used as a local cost map by with `move_base`, as seen in Fig. 3.6.

3.3 Integration with the Navigation Stack

To test this approach in navigation we integrated back into the Navigation Stack, taking only into account the local planning methods. For local planning we are using the ROS package `base_local_planner`, that provides implementations of the Trajectory Rollout and the Dynamic Window Approach. This planner uses costmaps to determine the optimal trajectory according to known costs between the target points, using a brute-force approach.

For the integration we have gone through several nodes, as shown in the sequence diagram of Fig. 3.7. We made the `MoveBase` node subscribe the our new map topic, the output of the Mechanical Effort Analysis node, Fig. 3.8, so that it could be included in the data structures of `base_local_planner`, initialized in the `TrajectoryPlannerROS` node. In this node, our map is sent to the `TrajectoryPlanner` node where is calculated the cost of traversing each cell, which it uses to score each possible trajectory, that behind this cost uses other characteristics. The score (T) is given by:

$$T = w_1 \cdot D + w_2 \cdot G + w_3 \cdot O + w_4 \cdot M, \tag{3.8}$$

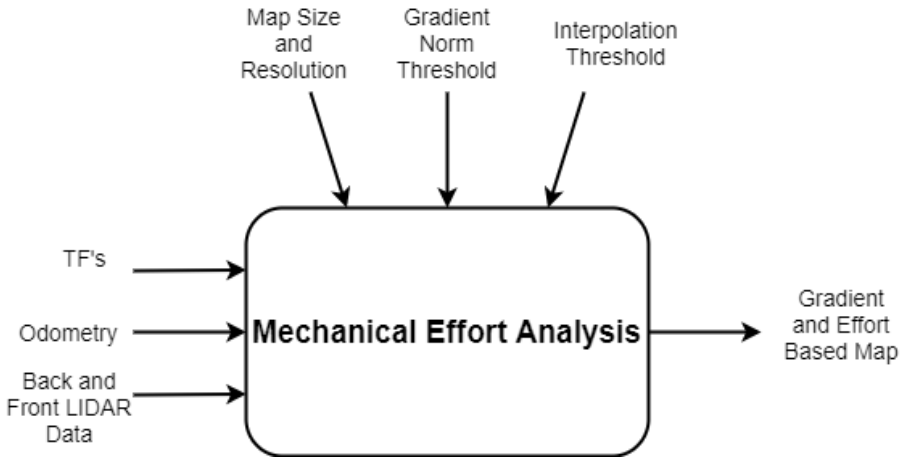


Figure 3.8: Resuming diagram of the mechanical effort analysis node.

where D is the distance to the goal, G distance to the local goals, O the obstacle cost given by the `costmap2D` (in this case zero because for the tests the observation sources were deactivated to only analyse our cost influence), and w_1, w_2, w_3 the respective weight of each parameter. The M parameter, corresponding to the mechanical cost obtained with our map and w_4 the respective weight were added by us.

Beyond that, in order to give priority flat and downward areas, we applied an offset to the M cost, with the average value that the map can take, 50:

$$\tau = \begin{cases} M - offset, & \text{if } M \geq offset \\ 0, & \text{otherwise,} \end{cases} \tag{3.9}$$

i.e., to all the flat and downward areas is given the value zero, it is safe to navigate.

All the trajectories are identified by a sampled velocity, and the trajectory with the smaller score is chosen, sending the corresponding velocity to the robot.

In addiction, the ROS Navigation Stack it is a powerful framework designed for, if possible, be used by a large number of robots, of different shapes and sizes. Optimizing the performance of the navigation stack requires tuning a number of parameters to ensure that it operates according to the requirements of our particular hardware. [56] presents a good explanation of some basics for parametrize the navigation stack. Beyond all the parameters needed for the robot configuration, the tolerance given in the goal, and the simulation, the main parameters that have a major impact in this strategy correspond to the trajectory scoring parameters, the w_1, w_2, w_3, w_4 of Equation 3.8, this parameters are shown in Table 3.1.

The **occdist_scale** is set to zero, because for this work the observation sources of the

Table 3.1: Main parameters needed to integrate our system with the ROS Navigation Stack, with the associated chosen values. (The name and description of existing parameters were taken from the ROS Wiki website[2]).

Parameter	Description	Value
pdist_scale (w_1)	The weighting for how much the controller should stay close to the path it was given	0.05
gdist_scale (w_2)	The weighting for how much the controller should attempt to reach its local goal, also controls speed	0.08
occdist_scale (w_3)	The weighting for how much the controller should attempt to avoid obstacles	0.0
mechanical_weight (w_4)	The weight for how much the controller should take into account the mechanical cost that come for our costmap (parameter created by us)	5.0

`move_base` are turned off, because in this tests we only want to test the applicability of the existing methods in light of our concept of mechanical effort. The **pdist_scale** and **gdist_scale** are set with low values and the **mechanical_weight** with a high value, this because if this difference did not happened, the planner did not take into account the cost of our map. Also, the **gdist_scale** is slightly higher than the **pdist_scale** because we want the controller to follow more the local paths than the global path, i.e., the paths that take into account the mechanical effort.

We have noticed that if we assign a low value to the **mechanical_weight**, the path planner ignores our costmap.. On the contrary, if we have a high value in weight, the robot get stuck, rotating on itself to find a safe path, but never succeeding.

In the following chapter, we will validate the technique experimentally, testing our approach and comparing it with existing techniques in light of the mechanical cost of traversal.

4 Experimental Validation

(a) Ranger platform without the mechanical mulcher attachment.

(b) Ranger in a simulated scenario.

Figure 4.1: Depictions of the real Ranger and in a simulated scenario. In the left pictures we observe the Ranger in a forest simulated environment, can see the similarities between the real and the simulated robot.

4.1 Experimental Goals, Setup and Metrics

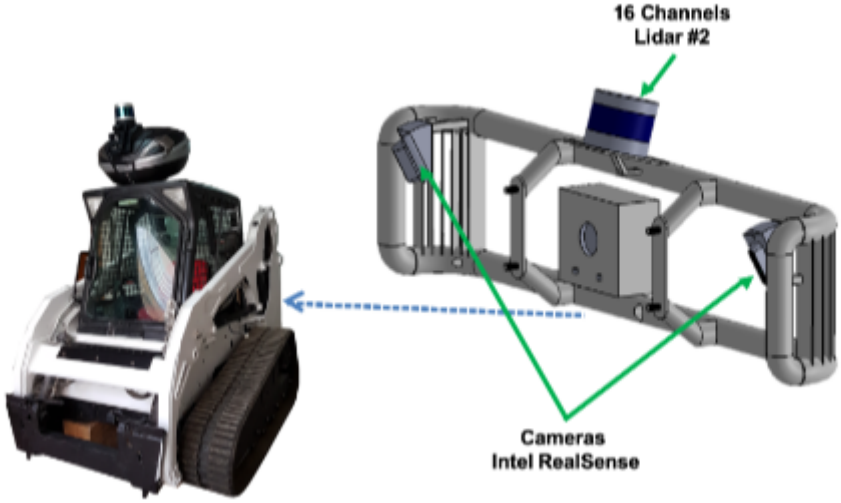
We have carried out several experiments in order to:

1. Demonstrate that existing techniques do not take mechanical cost into account;

2. Demonstrate that our approach is able to plan paths with lower costs than existing techniques.

These goals were defined in line with the overall goals of the thesis, and will be discussed with them in Chapter 5.

(a) Main sensor hub.



(b) Back sensor hub

Figure 4.2: Ranger sensor hardware framework.

4.1.1 Ranger

The Ranger (Fig. 4.1a) is a 4000kg robot, based on the Bobcat T190, equipped with several sensor modalities (Fig. 4.2), allowing the task of path planning and navigation in unknown environments:

- Two LeiShen C16 Laser Range Finders [1], which provide a 360º 3D view of the environment, contributing for instance for the estimation of distance to obstacles and occupancy;

[1] `http://en.leishen-lidar.com/product/leida/MX/15d44ea1-94f5-4b89-86eb-f5a781b04078.html`

- One FLIR AX8 thermal camera [2], which combines thermal imaging with a visual camera in one small package for continuous temperature monitoring;
- One Teledyne Dalsa Genie Nano C2420 multispectral camera [3], providing high speed, low noise, multispectral images, namely in the NIR range;
- Five Intel RealSense D435 RGB-D Cameras with five AAEON UP Board Atom for sensor acquisition [4], each consisting of a pair of depth sensors, RGB sensor, and infrared projector;
- GPS and RTK devices [5], used to enhance the precision of position data derived from satellite-based positioning systems;
- Inertial Measurement Unit [6], measures and reports the orientation of the body.

4.1.2 Simulator

In order to do sistematic navigation tests with a large forestry robot, we have used a simulator that realistically mimicks the Ranger platform and its sensors in a forest environment. This simulation was created with Unity by Ingeniarus Ltd. for the SEMFIRE project. It uses ROS#, allowing the simulator to communicate with ROS through rosbridge, a JSON API to ROS functionality for non-ROS programs. All scenarios, like the one shown in Figure 4.1b, were developed using the physics engine from Unity.

For our approach, from the large set of sensors that the robot presents, as observation source we only use the two 3D LIDARs.

4.1.3 Scenarios and Metrics

To test and compare the different algorithms we used three scenarios of the simulator:

1. One to validate the metrics, using a simple planar scenario with some obstacles (Fig. 4.3a);
2. In the second one, we used a forestry scenario, providing a minimally flat goal (Fig. 4.3b);
3. In the third goal, we intend to navigate through a more challenging forestry scenario, with large hills, to check whether the robot can overcome them efficiently (Fig. 4.3c).

[2] https://www.flir.com/products/ax8-automation/
[3] https://www.edmundoptics.eu/p/c2420-23-color-dalsa-genie-nano-poe-camera/4059/
[4] https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d415.html
[5] EmlidReachM+(https://emlid.com/reach/)
[6] https://www.pololu.com/product/2740

(a) First scenario. **(b)** Second scenario. **(c)** Third scenario.

Figure 4.3: Simulator scenarios where the comparison tests of the approaches were carried out.

We compare our modified version of the `base_local_planner` with the default version of `base_local_planner` and `dwa_local_planner` available in ROS, which uses RTP planning. We also compare our work with the approach proposed in [27], the current system used by the the SEMFIRE project for the Ranger navigation, that uses traversability analysis techniques and `teb_local_planner` for local planning (more detail in Section 2.3). We have opted for testing these specific techniques as they share their basis with state-of-the-art approaches, and are readily available as ROS-based packages. See Section 2.2 for more details on RTP, DWA and TEB.

We use different metrics that help us to evaluate and compare the different methods:

- Elapsed Time - Total time from a initial pose to the goal, in seconds;
- Traveled Distance - Every 0.1s we calculate the Euclidean distance between the current and the previous position of the robot, and use the sum as our measure of total distance traveled;
- Total Cost (T), defined as:

$$T = \sum_{i}^{N} (\nabla z_i), \tag{4.1}$$

  where ∇z_i corresponds to the z gradient value every 0.1s of the trajectory and T the sum of all the N z gradient values.

- Upward Cost (τ), defined as:

$$\tau = \begin{cases} \sum_{i}^{n} (\nabla z_i), & \text{if } \nabla z_i > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{4.2}$$

Table 4.1: Experimental results.

(a) Results for the first scenario

Approach	Traveled Distance	Time	T	τ
This Work	38.7505 ± 0.0175	46.5059 ± 0.1925	-0.0004 ± 0.0006	0.0037 ± 0.0009
`base_local_planner`	38.7692 ± 0.0200	46.4179 ± 0.1563	**-0.0006 ± 0.0004**	**0.0035 ± 0.0009**
`dwa_local_planner`	38.7693 ± 0.0839	77.4759 ± 0.1636	-0.0001 ± 0.0002	0.0032 ± 0.0002
Paper [27] system	39.0196 ± 0.0086	45.8759 ± 0.0592	-0.0017 ± 0.0002	0.0019 ± 0.0001

(b) Results for the second scenario

Approach	Traveled Distance	Time	T	τ
This Work	20.5997 ± 0.2443	35.9900 ± 3.1151	**-0.0038 ± 0.0033**	0.5653 ± 0.0225
`base_local_planner`	20.3395 ± 0.0579	25.6539 ± 3.4040	0.0011 ± 0.0015	0.5693 ± 0.0054
`dwa_local_planner`	20.5104 ± 0.1021	41.6579 ± 0.2079	0.0046 ± 0.0084	**0.5471 ± 0.0081**
Paper [27] system	20.5806 ± 0.0484	24.5139 ± 0.1974	0.0375 ± 0.0059	0.6902 ± 0.0209

(c) Results for the third scenario

Approach	Traveled Distance	Time	T	τ
This Work	18.6101 ± 0.5707	58.5279 ± 2.8600	**0.3206 ± 0.0304**	**0.4454 ± 0.0790**
`base_local_planner`	16.1218 ± 0.3155	49.7199 ± 2.7572	0.4078 ± 0.0594	0.7431 ± 0.0333
`dwa_local_planner`	15.9532 ± 0.0735	69.3179 ± 0.9581	0.4389 ± 0.0079	0.7373 ± 0.0054
Paper [27] system	17.8027 ± 1.3161	26.2499 ± 3.8105	0.3506 ± 0.0119	0.5639 ± 0.0769

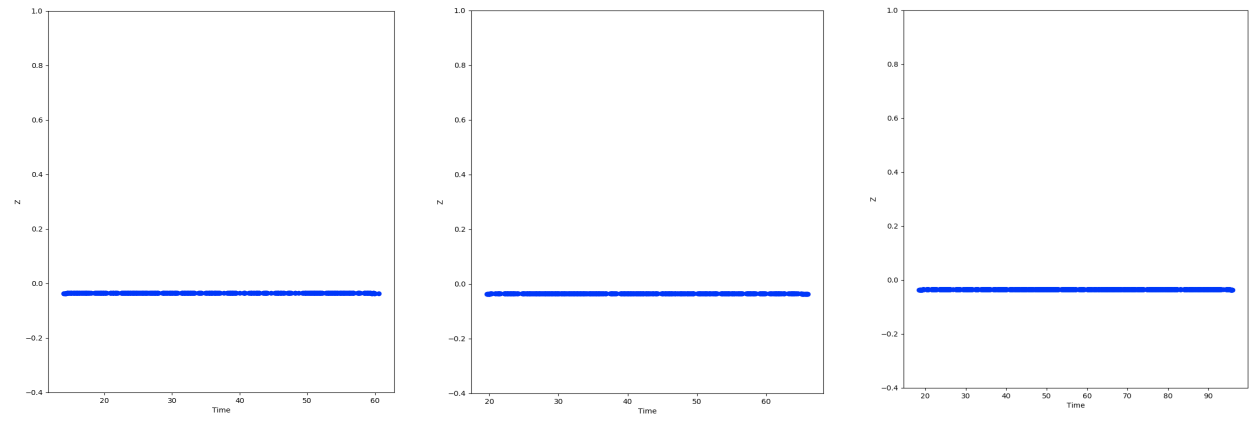
corresponding to the sum of the n positive z gradient values, which we use as a proxy for the mechanical effort for the same goals given to the robot.

In order to maximize the autonomy or minimize the energetic costs, the robot should mainly avoid steep climbs to reduce substantially the mechanical effort involved in the planned trajectory path. In this tests we aimed that our approach have a lower Total Cost (T) and Upward Cost (τ), with the possibility that, since we are not only taking into account the shortest path, the distance traveled is greater. We start by doing a simple test with simple planar scenario, in order to validate all the metrics.

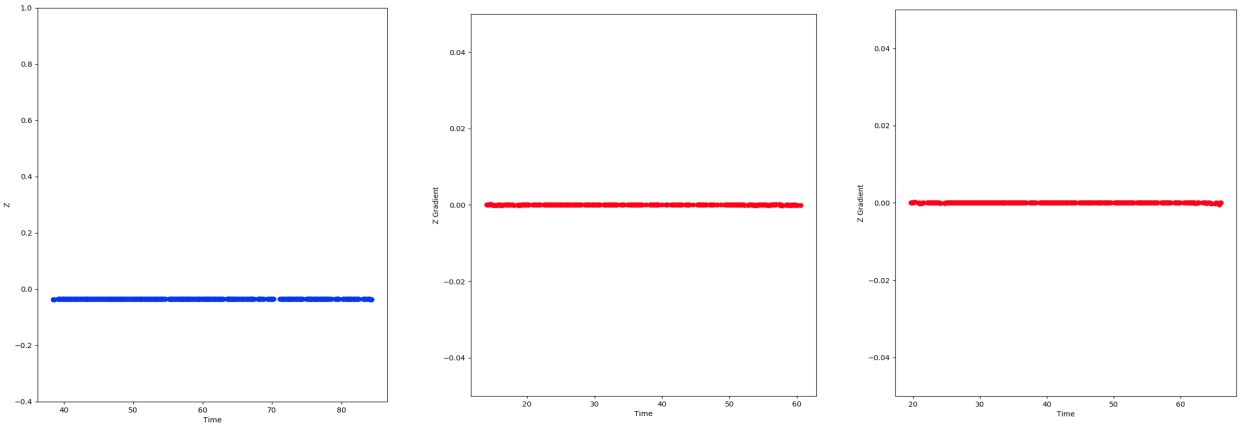
All the algorithms were tested 10 times on each scenario in order to obtain an average and standard deviation for each metric. To test our approach, we used a mechanical effort costmap with size 15 meters and resolution 0.5 meters/cell.

4.2 Results and Discussion

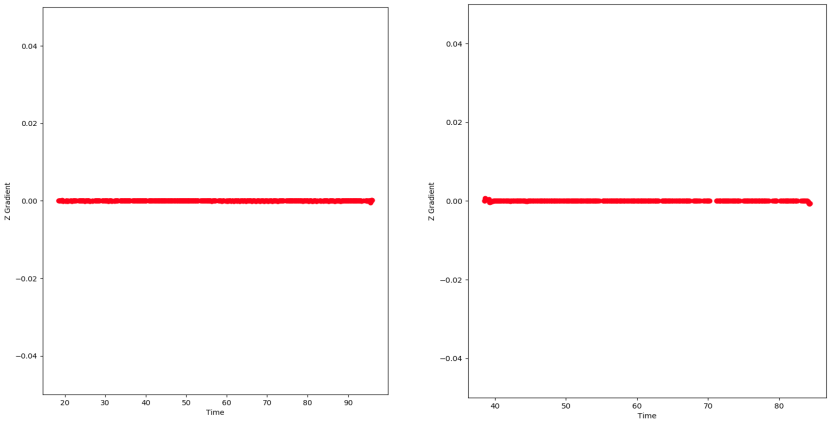
Given the results in the Table 4.1a, the planar scenario, we can observe that all the techniques make efficient use of mobility resources: they plan trajectories that take the robot from A to B in a quick, efficient manner. As expected all the algorithms chose the shortest path, travelling essentially the same distance, with a small variance that can easily be attributed to randomness both in the simulator and the control algorithms themselves. In this scenario, it is expected that all the approaches have the value of z nearly zero and constant, Fig. 4.4a-d,



(a) Modified `base_local_planner`. **(b)** `base_local_planner`. **(c)** `dwa_local_planner`.



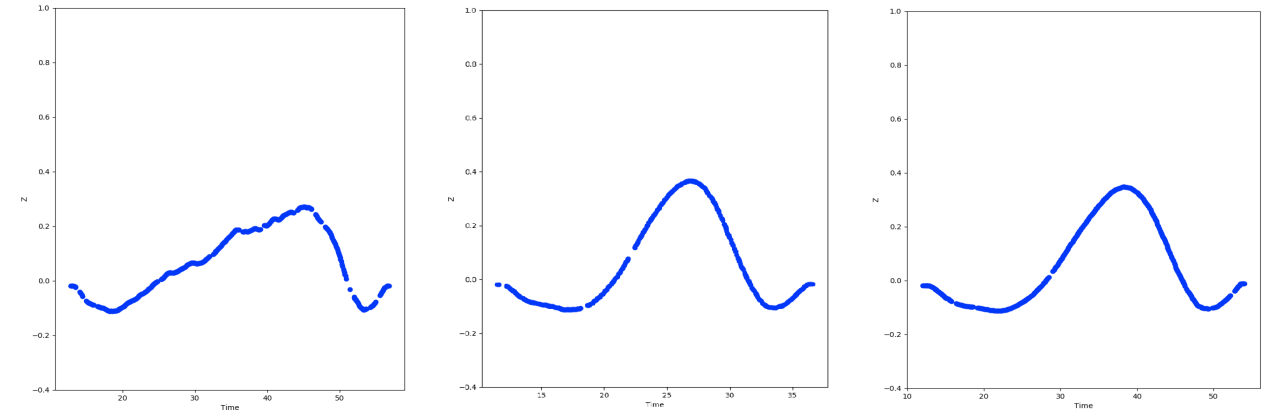
(d) `Paper [27] system`. **(e)** Modified `base_local_planner`. **(f)** `base_local_planner`.



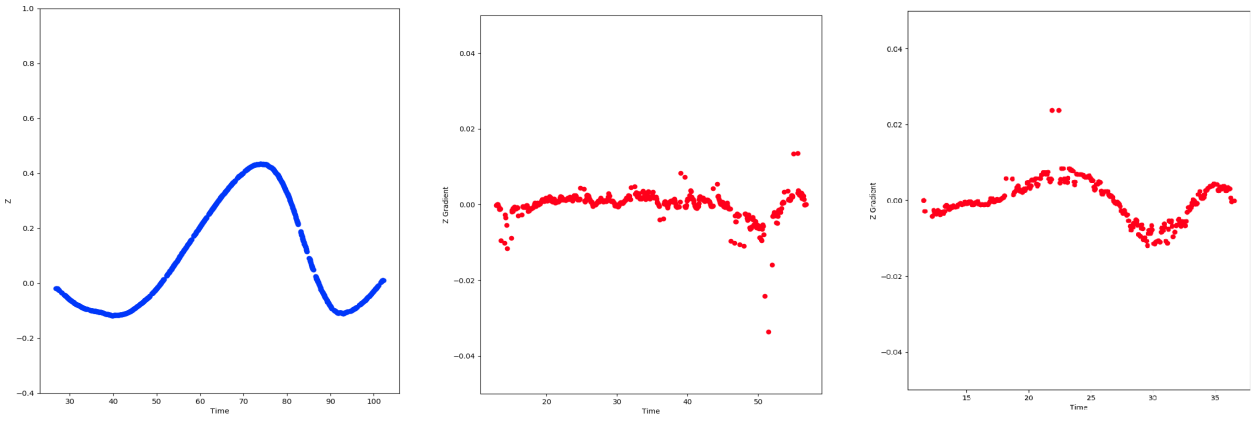
(g) `dwa_local_planner`. **(h)** `Paper [27] system`.

Figure 4.4: Plot of the z ((a), (b), (c) and (d)) during the route of the robot of one test in the first scenario and the corresponding gradient z values ((e), (f), (g), and (h)).

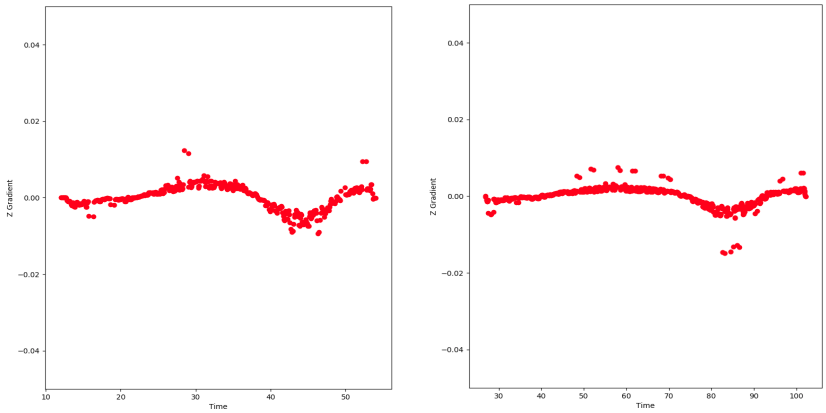
which result in a gradient of z almost zero, as it can be seen in Fig. Fig. 4.4e-h. Taking into account our mechanical effort definition, as well as, the definition of our metrics, it is expected that both Total Cost (T) and Upward Cost (τ) be nearly zero, which is confirmed according to the information in the table. It can be observed in the table that all the algorithms have a Total Cost (T) negative, meaning that the z value slightly went down more then up, but not significantly.



(a) Modified `base_local_planner`. **(b)** `base_local_planner`. **(c)** `dwa_local_planner`.



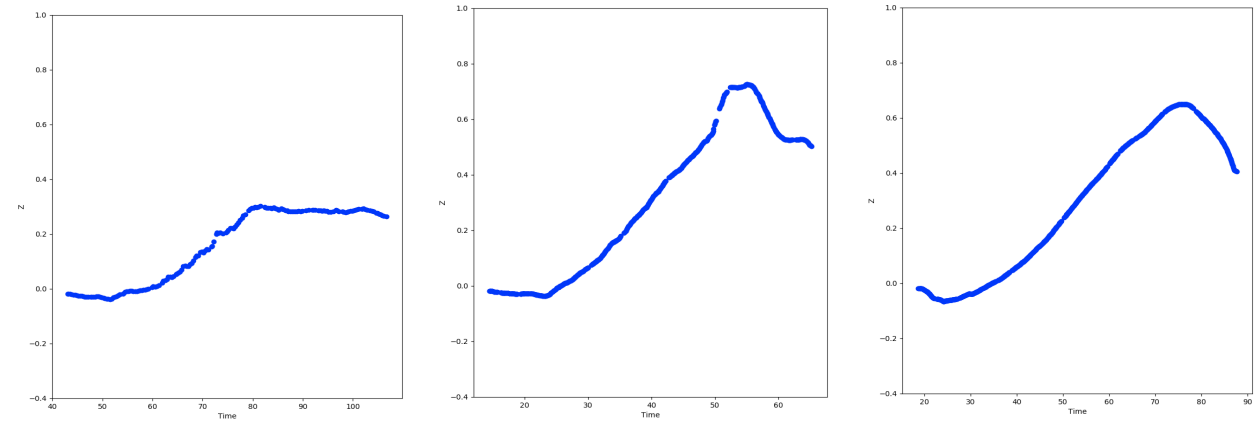
(d) Paper [27] system. **(e)** Modified `base_local_planner`. **(f)** `base_local_planner`.



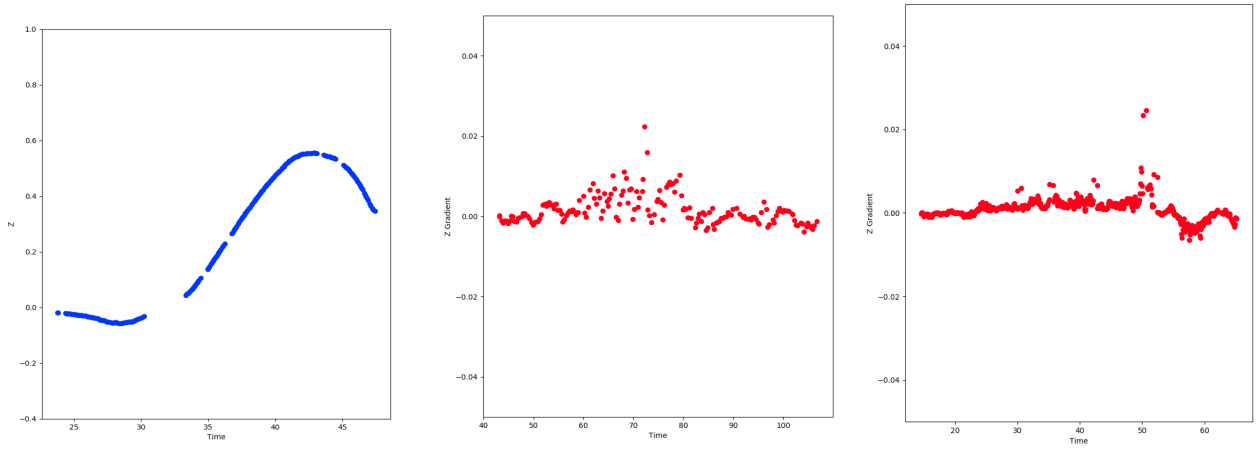
(g) `dwa_local_planner`. **(h)** Paper [27] system.

Figure 4.5: Plot of the z ((a), (b), (c) and (d)) during the route of the robot of one test in the second scenario and the corresponding gradient z values ((e), (f), (g), and (h)).

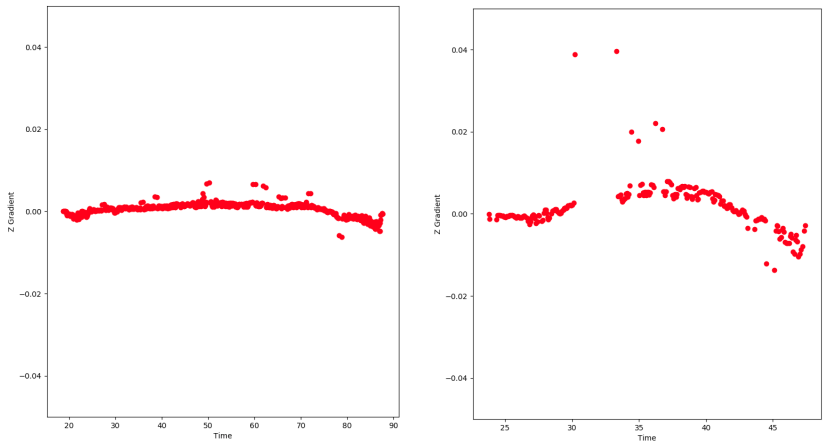
In this first scenario, with the more simple scenario where it is more likely to the robot have a constant speed, we calculated the average speed of all the techniques taking into account the traveled distance and the elapsed time. The `dwa_local_planner` tends to be slightly slower, taking more time to reach the goal, with only an average speed of 0.5004 ± 0.0043 m/s, while the standard `base_local_planner`, the modified `base_local_planner` and the approach presented in [27] obtained an average speed of $0.8352 \pm 0.0.0037$ m/s,



(a) Modified `base_local_planner`. **(b)** `base_local_planner`. **(c)** `dwa_local_planner`.



(d) Paper [27] system. **(e)** Modified `base_local_planner`. **(f)** `base_local_planner`.



(g) `dwa_local_planner`. **(h)** Paper [27] system.

Figure 4.6: Plot of the z ((a), (b), (c) and (d)) during the route of the robot of one test in the third scenario and the corresponding gradient z values ((e), (f), (g), and (h)).

$0.8332 \pm 0.0.0046$ m/s and 0.8505 ± 0.0015 m/s, respectively, very similar with one another and close to the maximum speed of the robot (0.9 m/s). This can be easily be attributed to a difference in the parameters used between approaches.

However, the existing techniques do not take mechanical cost into account in light of our definition of effort (Goal 1 of Section 4.1), the simple local planning algorithms tend to

choose the shortest path, not considering other costs. In the second and third scenarios we observe that both of the existing algorithms do not have problem to climb the hill instead of circumventing it, which can be seen in Fig. 4.5b-c and Fig. 4.6b-c. In this figure we observe that both algorithms tend go up and down instead of find a path that circumvent the hill, making it possible to obtain an improved Upward Cost (τ) and Total Cost (T). In the paper [27] the traversability analysis technique consider all the are traversable in this scenario, so the local planner just try to reach the shortest possible distance, as we can see in Fig. 4.5d the robot prefer go over the hill, presenting a higher Upward Cost (τ) and Total Cost (T) .

In the second scenario, we observe that our approach also does not avoid the hill, in Fig. 4.5a and observing the time that took our approach to reach the goal, Table 4.1b, we conclude that our approach had some indecision in the time to choose the trajectory but because the inclination of the hill was not so significant the robot ends to going over the hill. In the same table we observe that all the algorithms present a very similar Upward Cost (τ) and Total Cost (T), being the mean of the Upward Cost (τ) better in the `dwa_local_planner`, and the Total Cost (T), with the a negative value, our approach, meaning that the robot prefer to go more for downward slopes that upward, which is positive.

In the third scenario we can observe the main difference between our approach and the existing techniques, that our technique take into account our definition of mechanical effort (Goal 2 of Section 4.1). In this scenario we command the robot to navigate to a goal behind a larger hill and on a climb. Taking into account the plot on Fig. 4.6a and comparing with the Fig. 4.6b-c, we observe that our approach circumvent the big climb instead of attempting to traverse it directly, only climbing the necessary to reach the goal. In contrast, the existing approaches only choose the shortest path, not minimising the mechanical effort of climb the hill, preferring go up and down. This is further demonstrated by Table 4.1c, we can observe lower values for both Upward Cost (τ) and Total Cost (T). The work presented in [27] present a better Upward Cost (τ) and Total Cost (T) comparing with the existing local planning algorithms. The system tries to avoid some points that it does not consider traversable but largely favours steep slops to go up and down, similarly the other existing techniques, instead of circumvent the hill, Fig. 4.6d.

Generally, we can observe that our technique tends to pick longer paths, both in physical distance and in traversal time, but does take the overall mechanical effort into account. Thus, our technique avoids steeper slopes, which not only lead to higher consumption of fuel, but are also more dangerous to execute.

5 Conclusion

This dissertation presents a technique that seeks to enable large heavy-duty robots to navigate in challenging 3D forest environments. We start by surveying the state of the art in 3D local planning, concluding that there are a number of potentially applicable techniques, but that these do not taking mechanical effort into account (goal 1 of Section 1.2). Specifically, we propose a technique that estimates the cost of traversing each individual point in space according to terrain gradient, allowing the robot to prioritise paths that minimise its mechanical effort, and are thus safer and require less energy to execute (goal 2 of Section 1.2). Our method was integrated with the ROS Navigation Stack as a new factor in one of the default local path planners.

We have compared our work with existing techniques to determine how well they fared according to our metrics when navigating in a simulated forestry scenario. We determined that most techniques are applicable, making an effective use of mobility resources, with all techniques succeeding in guiding the robot from an initial pose to a goal point, both in flat and hilly environments. These results support goal 3 of Section 1.2.

We also observed that existing techniques do not take into account our definition of mechanical effort and power consumption. Instead, they consider only the distance traveled and ignore the terrain gradient, driving over steep slopes instead of trying to circumventing them. These techniques can thus be refined to produce more economical trajectories, which supports goal 4 of Section 1.2. In turn, our results show that the proposed technique favours paths with less mechanical effort for the robot, i.e. prioritizing level ground and downward slopes, potentially reducing the power consumption of the robot, which supports goal 5 of Section 1.2.

The proposed technique can now be extended and applied in a number of ways, described below.

5.1 Future Work

Map Creation Speed: In the creation of the map we have to analyse a significant amount of data that comes from the LIDARs point clouds, resulting in execution times that can take over a second on some machines. The step that lasts longer is projecting the pointcloud into the 2D grid, where points are looped over individually in order to obtain the grid cell to which each point belongs, and cells are looped over individually to compute the median of the height values; thus, map creation time increases with map resolution and pointcloud size. It would be important to further optimize the code, reducing execution time to facilitate real-time implementation.

Real Robot: It would be very interesting to transfer our tests to the real robot – the Ranger – to test our approach in real scenarios. This was one of the initial goals, but because of various technical constraints, as well as the pandemic context in which we live, it was not at all possible. As mentioned in Section 2.3 the proposed approach in [27] is the current system used by the the SEMFIRE project for the Ranger navigation, which already allows the robot to navigate in a 3D environment using elevation maps and traversability estimation. It would be interesting to integrate our approach with this system so that it also takes into account the mechanical effort and possibly the power consumption results from certain trajectories.

Local Planners: As seen in Section 2.4, the ROS Navigation Stack accepts different local planners. It would be interesting to integrate our approach with different local planners and determine whether they perform better with respect to our cost.

Interpolation Methods: After the 2D projection, to estimate the height value of the cells without information we perform an interpolation based on an adaptation of the natural neighbour interpolation, where we analyse all the neighbours and estimate the height value doing a median. It would be interesting to test different interpolation methods, potentially resulting in a better representation of the surface.

6 Bibliography

[1] Wilbert Aguilar and Stephanie Morales. 3d environment mapping using the kinect v2 and path planning based on rrt algorithms. *Electronics*, 5(4):70, 2016.

[2] Priyadarshi Bhattacharya and Marina L Gavrilova. Voronoi diagram in optimal path planning. In *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, pages 38–47. IEEE, 2007.

[3] Andrea Bonarini, Simone Ceriani, Giulio Fontana, and Matteo Matteucci. On the development of a multi-modal autonomous wheelchair. In *Handbook of Research on ICTs for Human-Centered Healthcare and Social Care Services*, pages 727–748. IGI Global, 2013.

[4] Abdeslam Boularias, Felix Duvallet, Jean Oh, and Anthony Stentz. Grounding spatial relations for outdoor robot navigation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1976–1982. IEEE, 2015.

[5] John Canny. A new algebraic method for robot motion planning and real geometry. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 39–48. IEEE, 1987.

[6] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. Rosplan: Planning in the robot operating system. In *Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015.

[7] R. Omar Chavez-Garcia, Jérôme Guzzi, Luca M. Gambardella, and Alessandro Giusti. Learning ground traversability from simulations. *IEEE Robotics and Automation Letters*, 3(3):1695–1702, 2018.

[8] Micael S. Couceiro, David Portugal, João F. Ferreira, and Rui P. Rocha. Semfire: Towards a new generation of forestry maintenance multi-robot systems. In *2019*

IEEE/SICE International Symposium on System Integration (SII), pages 270–276. IEEE, 2019.

[9] B. Cybulski, A. Wegierska, and Grzegorz Granosik. Accuracy comparison of navigation local planners on ros-based mobile robot. In *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, pages 104–111. IEEE, 2019.

[10] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

[11] Frank C. Dennis. Fire-resistant landscaping. *Service in action; no. 6.303*, 1999.

[12] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[13] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. Path planning and trajectory planning algorithms: A general overview. In *Motion and operation planning of robotic systems*, pages 3–27. Springer, 2015.

[14] David Gelperin. On the optimality of A*. *Artificial Intelligence*, 8(1):69–76, 1977.

[15] Brian P Gerkey and Kurt Konolige. Planning and control in unstructured terrain. In *ICRA Workshop on Path Planning on Costmaps*, 2008.

[16] Carlos Gordón, Patricio Encalada, Henry Lema, Diego León, Cristhian Castro, and Dennis Chicaiza. Autonomous robot navigation with signaling based on objects detection techniques and deep learning networks. In *Proceedings of SAI Intelligent Systems Conference*, pages 940–953. Springer, 2019.

[17] Maki K. Habib and Yvan Baudoin. Robot-assisted risky intervention, search, rescue and environmental surveillance. *International Journal of Advanced Robotic Systems*, 7(1):10, 2010.

[18] Emre Koyuncu and Gokhan Inalhan. A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3D environments. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 815–821. IEEE, 2008.

[19] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[20] Steven M LaValle and James J Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. 2000.

[21] Chi-Tai Lee and Ching-Chih Tsai. 3D Collision-Free Trajectory Generation Using Elastic Band Technique for an Autonomous Helicopter. In *FIRA RoboWorld Congress*, pages 34–41. Springer, 2011.

[22] Jurica Maltar, Ivan Marković, and Ivan Petrović. NOSeqSLAM: Not only sequential SLAM. In *Iberian Robotics conference*, pages 179–190. Springer, 2019.

[23] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *International Conference on Robotics and Automation*, 2010.

[24] Steven Martin and Peter Corke. Long-term exploration & tours for energy constrained robots with online proprioceptive traversability estimation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5778–5785. IEEE, 2014.

[25] Ellips Masehian and Davoud Sedighizadeh. Classic and heuristic approaches in robot motion planning-a chronological review. *World Academy of Science, Engineering and Technology*, 23(5):101–106, 2007.

[26] Yoichi Morales, Alexander Carballo, Eijiro Takeuchi, Atsushi Aburadani, and Takashi Tsubouchi. Autonomous robot navigation in outdoor cluttered pedestrian walkways. *Journal of Field Robotics*, 26(8):609–635, 2009.

[27] Ahmad Kamal Nasir, André G. Araújo, and Micael S. Couceiro. Localization and Navigation Assessment of a Heavy-Duty Field Robot. Ingeniarius, Lda, 2020.

[28] Christoph Niederberger, Dejan Radovic, and Markus Gross. Generic path planning for real-time applications. In *Proceedings Computer Graphics International, 2004.*, pages 299–306. IEEE, 2004.

[29] Panagiotis Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 26(4):1373–1385, 2013.

[30] BK Patle, Anish Pandey, DRK Parhi, A. Jagadeesh, et al. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 2019.

[31] Andreas Pfrunder, Paulo VK Borges, Adrian R. Romero, Gavin Catt, and Alberto Elfes. Real-Time Autonomous Ground Vehicle Navigation in Heterogeneous Environments Using a 3D LiDAR. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2601–2608. IEEE, 2017.

[32] David Portugal, João F. Ferreira, and Micael S. Couceiro. Requirements Specification and Integration Architecture for Perception in a Cooperative Team of Forestry Robots. In *In Proc. of Towards Autonomous Robotic Systems 2020 (TAROS 2020)*. University of Nottingham, Nottingham, UK, July 22-24 2020.

[33] Sebastian Pütz, Thomas Wiemann, Jochen Sprickerhof, and Joachim Hertzberg. 3D navigation mesh generation for path planning in uneven terrain. *IFAC-PapersOnLine*, 49(15):212–217, 2016.

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

[35] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807. IEEE, 1993.

[36] Robert Reid, Andrew Cann, Calum Meiklejohn, Liam Poli, Adrian Boeing, and Thomas Braunl. Cooperative multi-robot navigation, exploration, mapping and object detection with ros. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1083–1088. IEEE, 2013.

[37] Ricardo Reis, Filipe Neves dos Santos, and Luís Santos. Forest Robot and Datasets for Biomass Collection. In *Iberian Robotics conference*, pages 152–163. Springer, 2019.

[38] Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6. VDE, 2012.

[39] Jesús San-Miguel-Ayanz, Ernst Schulte, Guido Schmuck, Andrea Camia, Peter Strobl, Giorgio Liberta, Cristiano Giovando, Roberto Boca, Fernando Sedano, Pieter Kempeneers, et al. Comprehensive monitoring of wildfires in europe: the european forest

fire information system (effis). In *Approaches to Managing Disaster-Assessing Hazards, Emergencies and Disaster Impacts*. IntechOpen, 2012.

[40] Jose Luis Sanchez-Lopez, Min Wang, Miguel A. Olivares-Mendez, Martin Molina, and Holger Voos. A real-time 3D path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments. *Journal of Intelligent & Robotic Systems*, 93(1-2):33–53, 2019.

[41] Àngel Santamaria-Navarro, Ernesto H Teniente, Martí Morta, and Juan Andrade-Cetto. Terrain classification in complex three-dimensional outdoor environments. *Journal of Field Robotics*, 32(1):42–60, 2015.

[42] David A Schoenwald. Auvs: In space, air, water, and on the ground. *IEEE Control Systems Magazine*, 20(6):15–18, 2000.

[43] Robin Sibson. A brief description of natural neighbour interpolation. *Interpreting multivariate data*, 1981.

[44] Steven Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227, 1990.

[45] Juil Sock, Jun Kim, Jihong Min, and Kiho Kwak. Probabilistic traversability map generation using 3D-lidar and camera. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5631–5637. IEEE, 2016.

[46] Ilja Stasewitsch, Jan Schattenberg, and Ludger Frerichs. Cleaning robot for free stall dairy barns: Sequential control for cleaning and littering of cubicles. In *Iberian Robotics conference*, pages 115–126. Springer, 2019.

[47] Todor Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim J Lilienthal. Path planning in 3d environments using the normal distributions transform. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3263–3268. IEEE, 2010.

[48] Elena Stumm, Andreas Breitenmoser, François Pomerleau, Cédric Pradalier, and Roland Siegwart. Tensor-voting-based navigation for robotic inspection of 3D surfaces using lidar point clouds. *The International Journal of Robotics Research*, 31(12):1465–1488, 2012.

[49] Benjamin Suger, Bastian Steder, and Wolfram Burgard. Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3D-lidar data. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3941–3946. IEEE, 2015.

[50] Sebastian Thrun and Michael Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

[51] Inara Tusseyeva, Seong-Gon Kim, and Yong-Gi Kim. 3D global dynamic window approach for navigation of autonomous underwater vehicles. *International Journal of Fuzzy Logic and Intelligent Systems*, 13(2):91–99, 2013.

[52] Prahlad Vadakkepat, Kay Chen Tan, and Wang Ming-Liang. 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)*, volume 1, pages 256–263. IEEE, 2000.

[53] Liang Yang, Juntong Qi, Dalei Song, Jizhong Xiao, Jianda Han, and Yong Xia. Survey of robot 3d path planning algorithms. *Journal of Control Science and Engineering*, 2016:5, 2016.

[54] Simon X. Yang and Chaomin Luo. A neural network approach to complete coverage path planning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):718–724, 2004.

[55] Lingli Yu and Kaijun Zhou. A dynamic local path planning method for outdoor robot based on characteristics extraction of laser rangefinder and extended support vector machine. *International Journal of Pattern Recognition and Artificial Intelligence*, 30(02):1659004, 2016.

[56] Kaiyu Zheng. Ros navigation tuning guide. *arXiv preprint arXiv:1706.09068*, 2017.

Appendix A

Paper Accepted for Presentation in the WPPMFR 2020 at IROS 2020

3D Local Planning for a Forestry UGV based on Terrain Gradient and Mechanical Effort

Dora S. B. Lourenço, João F. Ferreira and David Portugal[1]

***Abstract*— Planning feasible paths in 3D environments is a challenging problem, mainly in forestry environments due to, for instance, the rough and slippery terrain or the challenges for perception, as those caused by the high amount of trees, wind, and general unstructured nature of the environment. This paper presents a work in progress to propose an innovative method for 3D local planning in outdoor environments, to facilitate autonomous navigation of a forestry Unmanned Ground Vehicle. The proposed method builds on the ROS navigation stack integrating a module that analyses the gradient of the terrain to quantify slopes on the robot's path, and taking into consideration its mechanical effort when planning paths to traverse.**

I. INTRODUCTION

Reliable outdoor autonomous navigation, particularly in forestry scenarios, is still a major challenge due to the dynamic conditions of outdoor environments such as weather, illumination and vegetation characteristics changes make it difficult to build a system that can robustly navigate all the time in all conditions [1]. The forest environment is unstructured making it difficult to perceive and to correctly localize in at all times [2]; the estimation of wheel odometry that is often a useful source for localization in most environments may not be usable in forests, because of the rough terrain conditions and slippage [3]. Forestry vehicles are also typically large heavy-duty machines, making their deployment process a hard and tedious task with safety concerns.

The main goal of this work is to propose local planning methods that allow a forestry robot to safely navigate from an initial to a target configuration while avoiding obstacles. Specifically, we:

1) Present a technique that uses a 3D pointcloud from the robot's sensors to estimate the cost of traversing each individual point in space, producing a costmap for navigation;
2) Test the feasibility and applicability of existing local planning methods in forestry environments in light of the mechanical costs of traversal;
3) Demonstrate that existing techniques do not take mechanical cost into account, and can thus be refined to produce more economical trajectories.

Fig. 1. The Ranger UGV.

We tested the approach in a forestry robot simulator, developed for the SEMFIRE R&D project[1], allowing us to perform repetitive testing without the related costs of using the real large-scale rig, presented in Fig. 1. This robot – the Ranger UGV – is the main actor of the SEMFIRE (Safety, Exploration, and Maintenance of Forests with the Integration of Ecological Robotics) project, which proposes the development of a robotic system to reduce fuel accumulation in forests, by eliminating flammable material for wildfire prevention, thus assisting in landscape maintenance tasks [4].

This paper is structured as follows: In Section II a literature review focused in 3D navigation is presented. In Section III is presented the approach proposed by this paper. In Section IV-A is presented experimental setup for the the tests and in IV-B the obtained results and their discussion.

II. RELATED WORK

Robot navigation is a common ability that enables the robot to reach the destination required by a certain job, planning and executing trajectories safely, both for itself, but also for humans and other entities that might be sharing its workspace [5].

Planning feasible paths in fully three-dimensional environments is a challenging problem [3]. Existing algorithms typically require the use of limited 3D representations that discard potentially useful information. In the literature, we can find several works on 3D navigation not just for ground

[1]D. Lourenço, J. F. Ferreira and D. Portugal are with the Institute of Systems and Robotics, University of Coimbra, 3030-290 Coimbra, Portugal; {dora.lourenco,jfilipe,davidbsp} at isr dot uc dot pt.

J. F. Ferreira is also with Nottingham Trent University, Computational Neuroscience and Cognitive Robotics Group, School of Science and Technology, Nottingham Trent University, Nottingham NG11 8NS, United Kingdom; joao.ferreira at ntu dot ac dot uk.

This work was supported by the Safety, Exploration and Maintenance of Forests with Ecological Robotics (SEMFIRE, ref. CENTRO-01-0247-FEDER-03269) research project co-funded by the "Agência Nacional de Inovação" within the Portugal2020 programme.

[1]http://semfire.ingeniarius.pt/, last accessed 2020/09/16.

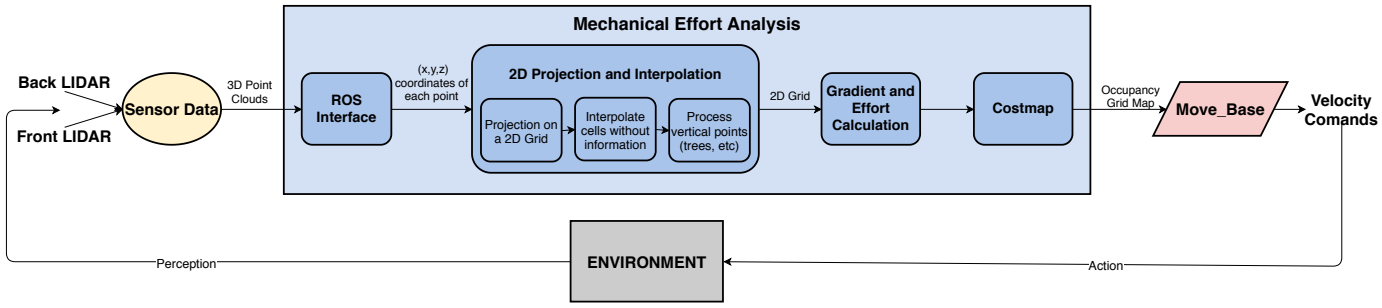


Fig. 2. Flow diagram of the implemented algorithm to estimate the cost of traversing each individual point in space.

robots, such as [6], [7], [8], but also aerial [9], [10], [11] and underwater robots [12].

As mentioned in Section I, we aim to allow a robot to safely navigate in a 3D forestry environment. Benefiting from existing local planning methods, we propose an algorithm that distinguishes traversable areas and integrates the cost of the mechanical effort of the robot in a `2D Grid`, to be considered for local planning. In a nutshell, we consider the mechanical effort as a quantity that accounts for the additional burden of the UGV when climbing hills in rough terrain environments. In [13], a similar approach to ours is followed. The authors represent the traversability map in a `2D Grid`, but instead of each cell representing the mechanical cost of that point they represent the probability that the vehicle can successfully drive over that cell.

Other seminal research works on traversability analysis methods are available in the literature [14], [15], and in this work, we make use of the Robot Operating System (ROS) to develop our own method to estimate the cost of traversing the environment. ROS includes some ready-to-use methods, such as `navigation_mesh`[2], adopted in [16]. The authors refer to this method as a 3D representation of estimation of distances, height differences, and roughness. If specific safety thresholds are violated, these areas will be marked as lethal obstacles. Similarly to [13], they only evaluate if the area is traversable, not taking into account the mechanical effort of the robot. In [17], the authors use a "perceive-decide" paradigm on an Elevation Map of the terrain to identify traversable areas, and based on this map the best trajectory is selected. There are other packages in ROS, such as `traversability_estimation`[3], which use the elevation map and traversability estimation filters to generate a traversability map.

For local planning, some of the works mentioned earlier use existing methods like [6], in which the authors integrate their system with ROS, using `base_local_planner`[4] with the Rollout Trajectory Planner (RTP) and `move_base`[5] as the low-level controller.

[2]`https://github.com/uos/mesh_navigation`
[3]`https://github.com/leggedrobotics/traversability_estimation`
[4]`http://wiki.ros.org/base_local_planner`
[5]`http://wiki.ros.org/move_base`
All URLs last accessed 2020/09/16.

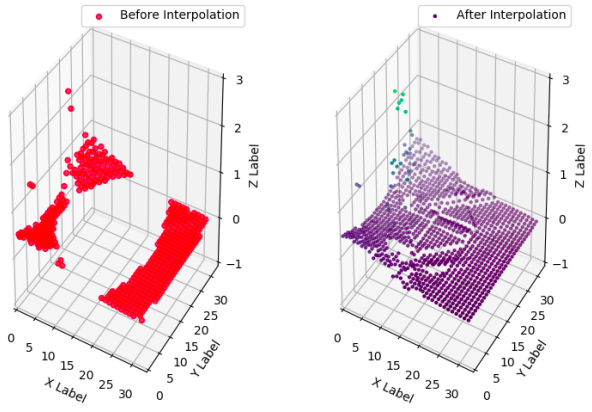


Fig. 3. 3D representation of the LIDAR information projected in a 2D grid with dimension 10 meters and resolution 0.3 meters/cell before and after the interpolation process.

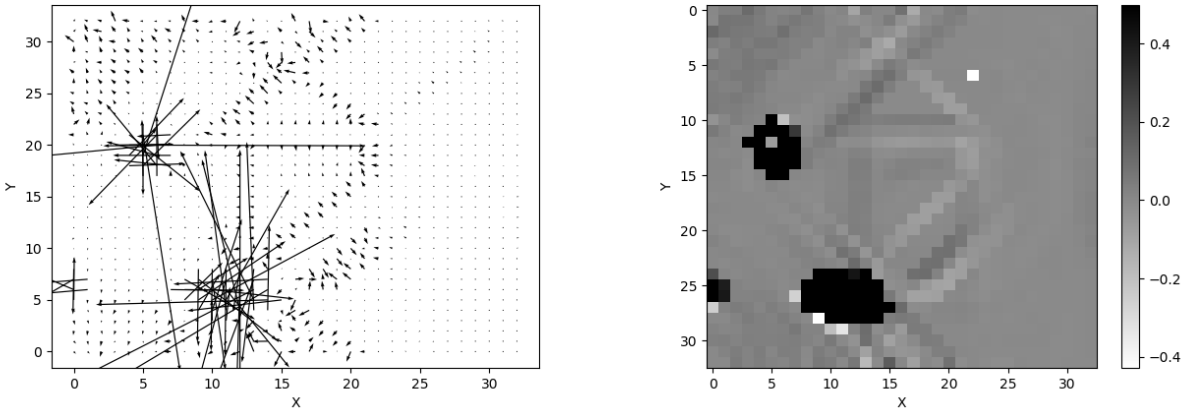
III. Proposed Approach

We have developed an algorithm that uses a 3D pointcloud from the robot's sensors to estimate the cost of traversing each individual point in space, producing a costmap for navigation. This allows us to quantify the mechanical effort of traversing each cell of a map, enabling the robot to decide to traverse paths that are sub-optimal according to traditional metrics, such as total distance or time, while optimizing its use of other resources, such as energy. An overview of this process is shown in Fig. 2.

The first step is a ROS Interface, in which we subscribe to the front and back 3D LIDAR Point Clouds available in the Ranger UGV. The points of each LIDAR are transformed from their respective frame into the robot base frame and concatenated in a single matrix.

We used the ROS Navigation Stack, a framework designed to generate minimal but complete autonomous navigation solutions, thus speeding-up the implementation processes required to obtain a robot navigation solution.

This framework traditionally operates in 2D, which is a disadvantage when it is necessary to plan for a 3D environment. To account for this, we can project the 3D LIDAR information into a 2D grid, so that information is represented by:



(a) Gradient arrows of the environment.

(b) Resulting costmap based on the gradient and effort.

Fig. 4. Graphics representing a costmap with dimension 10×10 meters and resolution 0.3 meters/cell. In (a) we can see some outliers, that are points with a very high gradient norm, and in (b) we can see the represented obstacles, such as trees in the environment.

$$\Phi(x, y) = z, \tag{1}$$

where $\Phi(x, y)$ is one cell of the grid, and z is the height of detected LIDAR obstacles in the corresponding discretized cell x, y.

The main output of the technique is a spatial grid that contains a specific cost per cell. The size and the resolution of the grid can be chosen by the user as a parameter of the approach. Due to their discrete nature, each cell may contain more than one LIDAR point. In these cases, the median is used as a representative value for the whole cell, as shown in the left picture of Fig. 3. Because this procedure results in many empty cells, it is necessary to perform interpolation to obtain an estimation of a value for those empty cells. Our approach is to apply Nearest Neighbour Interpolation by analysing the neighbour cells in row, column and diagonals with information and apply the median, as before, to estimate the cell value.

Strong discontinuities in the resulting map cause undesirable artifacts when interpolated. Thus, we have opted to filter out these points from the interpolation procedure according to:

$$|z_i - z_{average}| < \rho, \tag{2}$$

where z_i is the height estimation of each of the neighbours of the current cell, $z_{average}$ is the weighted average of all the neighbours, and ρ the threshold applied. Note that the weight of each neighbour is inversely proportional to the euclidean distance between the current cell and the neighbour cell, and the ρ threshold has been empirically adjusted to 2 m allowing for appropriate smoothing of the interpolation, resulting in the representation seen on the right of Fig. 3.

After the interpolation we calculate the gradient (Eq. 3) on each cell to obtain the inclination in each direction, Fig. 4(a):

$$\nabla z(x, y) =< \frac{\partial z}{\partial x}(x, y), \frac{\partial z}{\partial y}(x, y) > \tag{3}$$

We define effort ξ, as:

$$\xi = \cos(\theta), \tag{4}$$

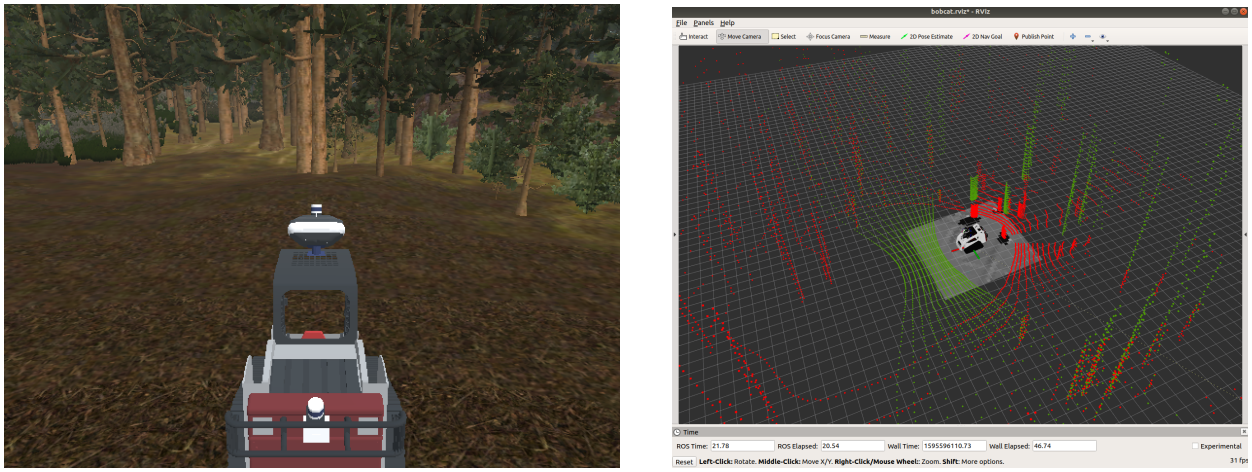
where θ is the angle between two vectors: the vector that connects the position of the robot with the position of the cell we are currently analyzing, and the gradient vector in that cell. Because the gradient direction is the direction in which the function increases more quickly, when the effort is maximum ($\xi = 1$) we are facing an upward slope, and when minimum ($\xi = -1$), a downward slope.

In order to build a costmap of the surrounding environment, all cells with gradient norm above a certain threshold are automatically considered as non-traversable; to other cells, we apply the product between the gradient norm and the so-called effort. Thus, we obtain a `2D Occupancy Grid Map` as in Fig. 4(b), where the black cells correspond to the non-traversable areas.

This approach was then integrated back into `move_base`. For local planning we are using the `base_local_planner` ROS package. This planner uses costmaps determine the optimal trajectory according to known costs between the target points, using a brute-force approach. To integrate our map with this package we made `move_base` subscribe the our new map topic, so that it could be included in the data structures that `base_local_planner` analyses to calculate the cost of traversing each cell, which it uses to score each possible trajectory.

IV. Experiments

We have carried out experiments to demonstrate that existing local planning methods in forestry environments do not take mechanical cost into account, and can thus be refined to produce more economical trajectories, in light of our definition of effort (see claims 2) and 3) in Section I).



(a) Typical simulator scene.

(b) Simulated environment in `rviz`.

Fig. 5. Images representing the same scenario, in (a) within the Unity simulator, and in (b) the information that the sensors perceive from robot surroundings, as shown in `rviz`.

(a) First scenario.

(b) Second scenario.

(c) Third scenario.

Fig. 6. Simulator scenarios where the comparison tests of the approaches were carried out.

A. *Experimental Setup*

As mentioned in Section I, we have used a Unity-based simulator that realistically simulates a 3D forest environment. In Fig. 5(a) we show an image of a typical scene in the simulator. The simulated Ranger UGV is equipped with two 3D LIDARs, that are used as source data for our algorithm. In Fig. 5(b), we can see the information collected by the front (green dots) and back (red dots) LIDARs, as well as the local `2D Occupancy Grid Map` created with our algorithm.

We compare the default version of `base_local_planner` and `dwa_local_planner` available in ROS, measuring distance traveled, the time it took to get to the target point, the total cost (T), defined as:

$$T = \sum_{i}^{N} (\nabla z_i), \tag{5}$$

where ∇z_i corresponds to the z gradient value every 0.1s of the trajectory and T the sum of all the N z gradient values. The Upward Cost (τ), defined as

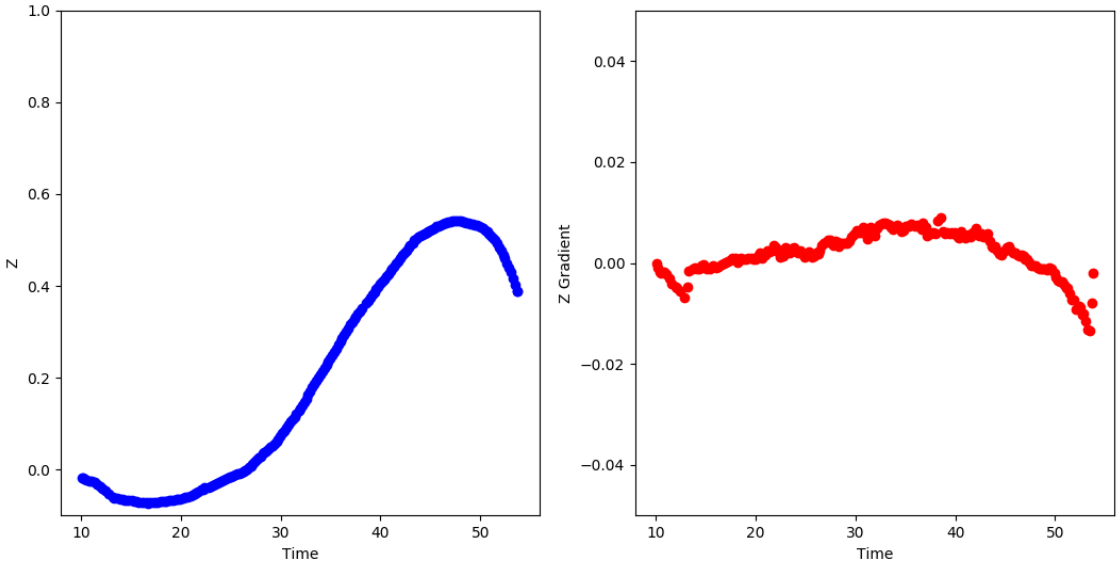
$$\tau = \begin{cases} \sum_{i}^{n} (\nabla z_i), & \text{if } \nabla z_i > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

corresponding to the sum of the n positive z gradient values, that is, the mechanical effort for the same goals given to the robot. We tested different scenarios (*cf.* Fig. 6):

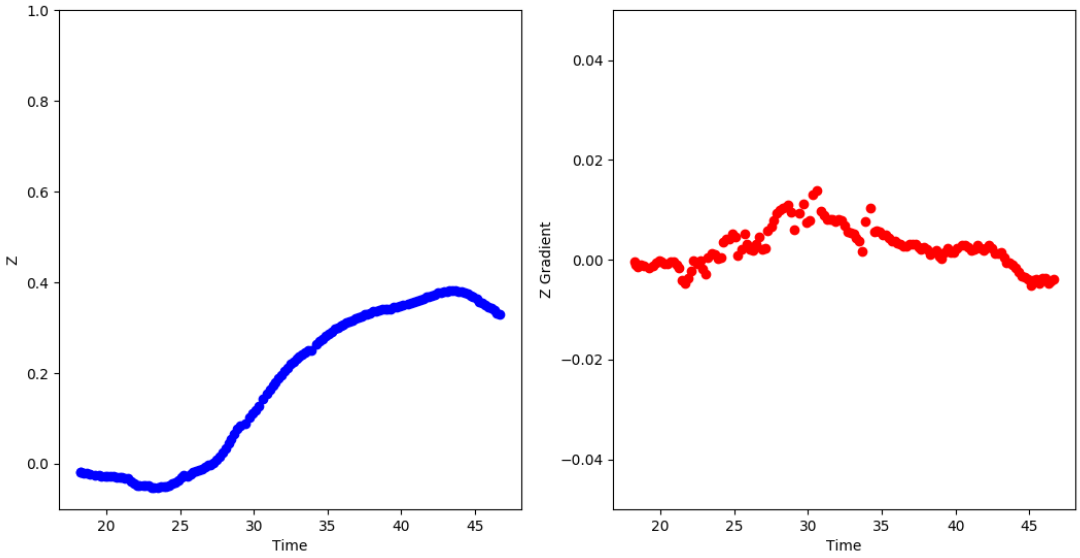
1) One to validate the metrics, using a simple planar scenario with some obstacles, Fig. 6(a);
2) In the second one, we used a forestry scenario, providing a minimally flat goal, with small map elevations, Fig. 6(b);
3) In the third goal, we intend the robot to go through a more challenging forestry scenario, with large hills, to check whether the robot can overcome them, Fig. 6(c).

B. *Results and Discussion*

Given the results in the Table I, we can observe that existing techniques make efficient use of mobility resources: they plan trajectories that take the robot from A to B in a quick, efficient manner. In the first test, the robot just had to go through a planar scenario, so as expected both algorithms chose the shortest path, and the traveled distance is the same. Being a planar scenario it was expected that both algorithms obtained a total cost very close to zero, with a small variance. The `dwa_local_planner` even obtained a very small negative total cost T, meaning that



(a) Resulting z (left graph) and z gradient (right graph) values as a function of time of `dwa_local_planner`.



(b) Resulting z (left graph) and z gradient (right graph) values as a function of time of `base_local_planner`.

Fig. 7. Graphs obtained from the z values and the z gradient, with the two algorithms along the third scenario.

there was a slightly greater negative variation of the gradient than positive along the path. The `dwa_local_planner` tends to be slightly slower, taking more time to reach the goal, with only an average speed of 0.459 m/s, while the `base_local_planner` obtained an average speed of 1.180 m/s, close to the maximum speed of the robot (1.5 m/s).

However, they do not take mechanical cost into account in light of our definition of effort. As said in the last paragraph, both algorithms tend to choose the shortest path, not considering other costs. In Fig. 7, and in Tables I-III we can see that both algorithms prefer to go through a big climb, presenting a high Upward Cost (τ), instead of circumventing it. In Fig. 5(a) we can see that some hills can be circumvented, preventing the high mechanical effort.

Therefore, these can be refined to produce more economical trajectories, avoiding slopes by preventing the robot to just choose the shortest path. In order to maximize the autonomy or minimize the energetic costs, the robot should mainly avoid steep climbs to reduce substantially the mechanical effort involved in the planned trajectory path.

V. Conclusion

This paper presents a work in progress on a technique to estimate the cost of traversing each individual point in space. The method is grounded on prioritizing paths that minimize the mechanical effort to the robot. We have defined

TABLE I

RESULTS FOR THE FIRST SCENARIO

Approach	Traveled Distance (m)	Time (s)	T	τ
base_local_planner	38.680	**32.779**	0.00029	0.006
dwa_local_planner	**38.647**	79.719	**-0.00026**	**0.001**

TABLE II

RESULTS FOR THE SECOND SCENARIO

Approach	Traveled Distance (m)	Time (s)	T	τ
base_local_planner	23.366	**25.559**	**0.2706**	**0.7899**
dwa_local_planner	**23.335**	48.36	0.276	0.842

TABLE III

RESULTS FOR THE THIRD SCENARIO

Approach	Traveled Distance (m)	Time (s)	T	τ
base_local_planner	17.670	**28.98**	**0.390**	**0.4646**
dwa_local_planner	**16.88**	43.659	0.403	0.6117

metrics and tested competing techniques to determine how well they fared according to our standards. We can conclude that there is room for improvement and, thus, this research will continue.

In the short term, besides testing additional interpolation methods, and to generally improve, document and make the proposed approach available to the community, we aim to obtain quantitative results comparing the travel time and energy spent when taking into consideration the mechanical effort. Also, we would like to propose metrics to answer questions such as: "Is circumventing a hill better than driving on it considering that a larger route will imply also additional power consumption to some extent?". Future work will tackle two main fonts: (1) we will compare our approach with the standard techniques; (2) when a stable approach is proposed, tests will be transferred to the real robot – the Ranger, a 4000 kg heavy-duty UGV, based on the Bobcat T190 (see Fig. 1).

ACKNOWLEDGMENT

The authors would like to thank Gonçalo S. Martins for the technical discussion and scientific guidance and Rui P. Rocha for invaluable help with technical support for carrying out this work.

REFERENCES

[1] Y. Morales, A. Carballo, E. Takeuchi, A. Aburadani, and T. Tsubouchi, "Autonomous robot navigation in outdoor cluttered pedestrian walkways," *Journal of Field Robotics*, vol. 26, no. 8, pp. 609–635, 2009.

[2] M. K. Habib and Y. Baudoin, "Robot-assisted risky intervention, search, rescue and environmental surveillance," *International Journal of Advanced Robotic Systems*, vol. 7, no. 1, p. 10, 2010.

[3] R. Reis, F. N. dos Santos, and L. Santos, "Forest robot and datasets for biomass collection," in *Iberian Robotics conference*, pp. 152–163, Springer, 2019.

[4] M. S. Couceiro, D. Portugal, J. F. Ferreira, and R. P. Rocha, "Semfire: Towards a new generation of forestry maintenance multi-robot systems," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, pp. 270–276, IEEE, 2019.

[5] A. Bonarini, S. Ceriani, G. Fontana, and M. Matteucci, "On the development of a multi-modal autonomous wheelchair," in *Handbook of Research on ICTs for Human-Centered Healthcare and Social Care Services*, pp. 727–748, IGI Global, 2013.

[6] À. Santamaria-Navarro, E. H. Teniente, M. Morta, and J. Andrade-Cetto, "Terrain classification in complex three-dimensional outdoor environments," *Journal of Field Robotics*, vol. 32, no. 1, pp. 42–60, 2015.

[7] A. Linz, A. Ruckelshausen, E. Wunder, and J. Hertzberg, "Autonomous service robots for orchards and vineyards: 3D simulation environment of multi sensor-based navigation and applications," in *Proceedings book of the 12th International Conference on Precision Agriculture (ICPA)*, pp. 2327–2334, 2014.

[8] A. Pfrunder, P. V. Borges, A. R. Romero, G. Catt, and A. Elfes, "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D lidar," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2601–2608, IEEE, 2017.

[9] E. Koyuncu and G. Inalhan, "A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3D environments," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 815–821, IEEE, 2008.

[10] J. L. Sanchez-Lopez, M. Wang, M. A. Olivares-Mendez, M. Molina, and H. Voos, "A real-time 3D path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1-2, pp. 33–53, 2019.

[11] C.-T. Lee and C.-C. Tsai, "3D collision-free trajectory generation using elastic band technique for an autonomous helicopter," in *FIRA RoboWorld Congress*, pp. 34–41, Springer, 2011.

[12] I. Tusseyeva, S.-G. Kim, and Y.-G. Kim, "3D global dynamic window approach for navigation of autonomous underwater vehicles," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 13, no. 2, pp. 91–99, 2013.

[13] J. Sock, J. Kim, J. Min, and K. Kwak, "Probabilistic traversability map generation using 3D-lidar and camera," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5631–5637, IEEE, 2016.

[14] S. Martin and P. Corke, "Long-term exploration & tours for energy constrained robots with online proprioceptive traversability estimation," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5778–5785, IEEE, 2014.

[15] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Learning ground traversability from simulations," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1695–1702, 2018.

[16] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg, "3d navigation mesh generation for path planning in uneven terrain," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 212–217, 2016.

[17] D. Bonnafous, S. Lacroix, and T. Siméon, "Motion generation for a rover on rough terrains," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 2, pp. 784–789, IEEE, 2001.