NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Path Planning for incline terrain using Embodied Artificial Intelligence

Georgios D. Kamaras

Supervisors: **Stasinos Konstantopoulos**, Research Associate
Panagiotis Stamatopoulos, Assistant Professor

ATHENS

June 2018

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πλοήγηση σε κεκλιμένο έδαφος μέσω Ενσώματης Τεχνητής Νοημοσύνης

Γεώργιος Δ. Καμάρας

Επιβλέποντες: **Στασινός Κωνσταντόπουλος**, Συνεργαζόμενος Ερευνητής
Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2018

BSc THESIS

Path Planning for incline terrain using Embodied Artificial Intelligence

Georgios D. Kamaras
S.N.: 1115201400058

Supervisors: **Stasinos Konstantopoulos**, Research Associate
Panagiotis Stamatopoulos, Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πλοήγηση σε κεκλιμένο έδαφος μέσω Ενσώματης Τεχνητής Νοημοσύνης

Γεώργιος Δ. Καμάρας
Α.Μ.: 1115201400058

Επιβλέποντες: **Στασινός Κωνσταντόπουλος**, Συνεργαζόμενος Ερευνητής
Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής

ABSTRACT

Embodied Artificial Intelligence aims to cover the need of a search problem's representation, as well as the representation of what constitutes a "good" solution to this problem in a smart machine. In this thesis' case, this smart machine is a robot. When we combine Artificial Intelligence and Robotics we can define experiments where the search space is the physical world and the results of each action constitute each solution's evaluation. In my thesis' context, I had the opportunity to experiment with the development of artificial intelligence algorithms that guide an unmanned ground vehicle to discover the solution of a tough outdoor navigation problem, like traversing a terrain region of steep incline. I attempted to face the problem with three different approaches. A Hill Climbing algorithm approach, a N-best search approach and an Evolutionary Algorithm approach, each one with its own strengths and weaknesses. In the end, I created and I evaluated demonstrations, both in simulated scenarios and in a real world scenario. The results of these demonstrations show a clear progress in the approach of the aforementioned problem, by the robotic platform.

SUBJECT AREA: Robotics

KEYWORDS: Path planning, Offline planning, Hill-climbing, N-best search, Evolutionary Algorithm, Unmanned Ground Vehicle, Bezier Curves, Robotic Platform Dynamics

ΠΕΡΙΛΗΨΗ

Η Ενσώματη Τεχνητή Νοημοσύνη στοχεύει στο να καλύψει την ανάγκη για την αναπαράσταση ενός προβλήματος αναζήτησης, καθώς και την αναπαράσταση του τι συνιστά “καλή” λύση για το πρόβλημα αυτό σε μια έξυπνη μηχανή. Στην περίπτωση της παρούσας πτυχιακής, αυτή η έξυπνη μηχανή είναι ένα ρομπότ. Συνδυάζοντας την Τεχνητή Νοημοσύνη και την Ρομποτική μπορούμε να ορίσουμε πειράματα των οποίων ο χώρος αναζήτησης είναι ο φυσικός κόσμος και τα αποτελέσματα κάθε πράξης συνιστούν την αξιολόγηση της κάθε λύσης. Στο πλαίσιο της πτυχιακής μου είχα την ευκαιρία να πειραματιστώ με την ανάπτυξη αλγορίθμων Τεχνητής Νοημοσύνης οι οποίοι καθοδηγούν ένα μη επανδρωμένο όχημα εδάφους στην ανακάλυψη μιας λύσης ενός δύσκολου προβλήματος πλοήγησης σε εξωτερικό χώρο, όπως η διάσχιση ενός εδάφους με απότομη κλίση. Επιχείρησα να αντιμετωπίσω το πρόβλημα αυτό με τρεις διαφορετικές προσεγγίσεις, μία με αλγόριθμο Hill Climbing, μία με N-best αναζήτηση και μία με Εξελικτικό Αλγόριθμο, καθεμία με τα δικά της προτερήματα και τις δικές της αδυναμίες. Τελικά, δημιούργησα και αξιολόγησα επίδειξεις, τόσο σε προσομοιωμένα σενάρια όσο και σε ένα σενάριο στον πραγματικό κόσμο. Τα αποτελέσματα αυτών των επιδείξεων δείχνουν μία σαφή πρόοδο στην προσέγγιση του προαναφερθέντος προβλήματος από μία ρομποτική πλαρφόρμα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ρομποτική

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Σχεδιασμός μονοπατιού, Σχεδιασμός εκτός σύνδεσης, Hill-climbing, Αναζήτηση Ν-καλύτερων, Εξελικτικός Αλγόριθμος, Μη Επανδρωμένο Όχημα Εδάφους, Καμπύλες Bezier, Δυναμική Ρομποτικής Πλατφόρμας

To my family.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Dr. Stasinos Konstantopoulos, from the Roboskel Group of the Software and Knowledge Engineering Lab of the Institute of Informatics and Telecommunications, NCSR “Demokritos”, for suggesting to me the topic of this thesis and giving me the chance to do my first serious work in the field of Robotics. I would like to thank Assistant Professor Panagiotis Stamatopoulos for encouraging me to contact the NCSR “Demokritos” Roboskel Group for a thesis that would combine Artificial Intelligence and Robotics, for his continuous support and for providing me with really useful input whenever I needed it.

I would also like thank the whole staff of the NCSR “Demokritos” Roboskel Group for their continuous help and support while writing this thesis. Especially, I would like to thank Dr. Stasinos Konstantopoulos and George Stavrinos for their continuous assistance, for sharing their great expertise and experience and for providing me with valuable ideas throughout the course of this project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This is an undergraduate thesis on path planning and, as such, its concepts lie in the intersection of the fields of Robotics and Artificial Intelligence. It was writen as part of the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens undergraduate studies degree requirements. It represents the results of its author's research in terrain traversability methods from Octomber of 2017 to June of 2018. It aims to propose a novel method for a robot to attempt climbing a terrain of steep incline.

1. INTRODUCTION

1.1 Motivation

Unmanned ground vehicles (UGVs) will play a crucial role to the future of the humanity. To put it in a simple context, they will be the mankind's first choice for a mission in areas where a manned mission is considered either too high-risk or not cost-effective enough. This opens a very broad field of potential applications, from first response to physical disasters and exploring remote planets, to cargo transportation and mining activities. One thing that all these potential applications have in common is that the unmanned ground vehicle will have to move efficiently in a challenging and unstructured terrain without expecting any human intervention. This will require the vehicle to solve many problems along the way, as they arise, by relying on some simple priors that describe its interaction with the physical world, no matter the world's structure. One such problem is the traversal of a terrain with steep incline.

1.2 Problem Statement

The problem that I will try to solve concerns the traverse of inclined terrain by an unmanned ground vehicle (UGV). The robot should be able to reach from an initial state to a goal state by traversing a terrain with steep incline. The terrain should be challenging enough for the robotic platform to need to plan for its own safety, like avoiding to tip over, to fall into a lethal obstacle or slide backwards, situations that may damage the platform or leave it in an unrecoverable state. The initial state and the goal state must be two states where the robot is in a neutral and balanced position towards the physical world.

1.3 Approach

The approach presented in this thesis attempts to solve the above problem at a path planning level. It draws inspiration from the path that a wheeled vehicle normally follows while trying to reach the top of a mountain. The car never goes straight up, but instead follows a path of smooth curves dictated by the roads that humans have created on the mountain. This is a luxury that a UGV cannot be expected to have, especially when trying to navigate an outdoor terrain. The UGV should be able to create this path of smooth curves by itself, using some basic knowledge regarding the terrain's structure and the boundaries inside which it has to operate, and this is what my approach attempts to achieve.

1.4 Outline

This thesis is organized in six main chapters. In *Chapter 2*, I provide a short review of the terrain traversability analysis for UGVs field, to prepare the reader for the concepts that

I am going to present in the rest of my thesis. In *Chapter 3*, I focus on the traversability of incline terrain, I present the problem that I attempted to solve and I present some work descriptive of the approaches that can be considered as the current state of the art. In *Chapter 4* I discuss the algorithms in which my implementation is going to be based. In *Chapter 5* I give a detailed explanation of my implementation of the problem's solution, based on the concepts that I have presented by then. In *Chapter 6*, I present the tests which I used to verify the efficiency of my approach in simulation and in *Chapter 7* I present the real-life scenario in which I demonstrated my methods. My conclusion is followed by two *Appendices* where I provide a presentation of the robotic platform's dynamics and a presentation of the Robot Operating System (ROS).

2. TERRAIN TRAVERSABILITY ANALYSIS FOR UGVS

Motion planning for unmanned ground vehicles constitutes a domain of research where several disciplines meet, ranging from artificial intelligence and machine learning to robot perception and computer vision. In view of the plurality of related applications such as planetary exploration, search and rescue, agriculture, mining and off-road exploration, the aim of the present survey is to review the field of 3D terrain traversability analysis that is employed at a preceding stage as a means to effectively and efficiently guide the task of motion planning. For this purpose, we are going to view the main streams of research in the field of terrain traversability analysis for UGVs. We are then going to talk in detail about the two main methodologies for sensory data processing, the proprioceptive and the exteroceptive, and some hybrid approaches that build on top of them. This will be followed by a discussion about the theories presented, their similarities and their differences, their impact and we are going to see some examples of real world breakthroughs in the field of terrain traversability analysis for UGVs. The survey will be concluded with a glimpse into the future of this field.

2.1 About

Terrain traversability's main goal is the navigation of a robotic *("unmanned")* ground vehicle within environments of varying complexity. At the same time, it wants to ensure that the robot is safe in terms of collisions or reaching unrecoverable states, while trying to achieve a goal in an optimal mode of operation. By paraphrasing Papadakis [54], the main components of the *traversability problem* can be captured with the following definition:

> Find a way for a ground vehicle to reside over a terrain region under an admissible state, by taking into account the vehicle's initial state, a terrain model, the robotic vehicle model, the kinematic constraints of the vehicle and a set of criteria based on which the optimality of an admissible state can be assessed.

Historically, terrain traversability estimation was initially addressed as a binary classification problem, where a terrain region was classified as either being traversable or non-traversable. Later on, the continuously growing trend of deploying robots into environments of increasing complexity created the need for a finer classification, which assigned a continuous traversability score or classified the terrain into the various classes that were commonly encountered within a particular application. However, binary terrain classification should not be viewed as redundant or trivial in the light of this trend. The computational complexity of the continuous traversability analysis increases together with the terrain complexity, hence, binary classification can assist in performing a more efficient analysis in the portions of a terrain that are most important for an application.

2.1.1 The main streams of research

Initially, the dominant approach for estimating traversability was the analysis of 2D digital elevation maps, like we see in Kweon and Kanade [39]. This trend was based in the utilization of occupancy grid maps for the terrain representation, which allowed the employment of efficient graph search search algorithms for the purpose of planning. As the years go by, the focus shifts towards the use of 3D point clouds, which can be acquired from LIDAR sensors. LIDAR sensors measure distance to a target by illuminating the target with pulsed laser light and measuring the reflected pulses with a sensor, thus using the laser light's travel time to calculate the distance to the target. Even when using point clouds, the grid maps remain the preferred choice for terrain representation. Simultaneously, other approaches emerge, utilizing passive vision sensors, inertia management units (IMUs), wheel slip sensors, collision bumpers or other domain-specific sensors, however the 3D world representation is in the center of almost every approach.

Based on the type of sensory data processing that is employed for the traversability estimation of a certain terrain region, the methodologies that are followed can be distinguished in three main streams of research:

- Proprioceptive sensory data processing
- Exteroceptive sensory data processing
- Hybrid approaches

The *Exteroceptive sensory data processing* can be further distinguished in two subfields; *Geometry-based* and *Appearance-based*. The majority of the employed methods projects in the first two categories, Proprioceptive and Exteroceptive analysis. The Hybrid approaches mainly lie in the simultaneous utilization of proprioceptive and exteroceptive sensors during the analysis.

In the next sections, we review the field of 3D terrain traversability analysis for the purpose of UGV motion planning. A short review of all the methods stated above is provided, with an aim to be kept as close to what is currently considered as the state-of-the-art as possible, without omitting to mention works that are seminal for any stream of research.

2.2 Proprioceptive traversability analysis

In the proprioceptive traversability analysis, the vehicle learns the best model that describes the difficulty in traversing a given terrain by analyzing various sensory inputs which come from interacting with it, such as vibrations, wheel slips, bumper hits, etc. Some of the most representative recent works in this category are Bermudez et al. [9], Hoffmann et al. [27] and Walas [78]. Bermudez et al. [9] show us how a robot which recognizes its terrain can adapt its control policy to traverse that terrain most effectively. Hoffmann et al. [27] work concerns the terrain classification based on its properties by a robot and they demonstrate a data-driven approach, which relies on a variety of sensors and pays

special attention to the active generation of the sensory data. What's impressive is that their approach results in an almost perfect classification performance.

Practically, proprioceptive sensing is often combined with data collected from the robot's exteroceptive sensors. This way, the always local proprioceptive traversability assessments can be propagated to distant regions, captured only by the platform's exteroceptive sensors. Some characteristic examples are Howard et al. [29] and Shneier et al. [61]. The former's work has influenced Peynot et al. [55] whose work focuses on hazard detection and avoidance and Santamaria-Navarro et al. [58] whose work deal with 3D terrain classification in outdoor environments, while both Howard et al. and Shneier et al. work is mentioned in the work of Suger et al. [68], that presents a semi-supervised learning approach, where the robot learns its traversability capabilities from a human operating it.

However, such approaches often leave the robot vulnerable to failures, like tipping over or crashing. For this reason, a highly tolerant and dispensable robotic platform is required whenever they are employed. Because of this, as we will see in this survey, the attention has been shifted towards methods that favor the exteroceptive sensing, so the traversability of a terrain region has been assessed before actually driving into it.

Before concluding our view of proprioceptive traversability analysis, we should mention the work of Jonschkowski and Brock [32, 33]. Their work focuses on the robot learning state representations of its physical world, which will allow it to successfully perform tasks like navigation with distracting moving objects. The key idea behind their work is the utilization of a set of *robotic priors*, which is essentially the robot's prior knowledge (observation) about its interactions with the physical world.

2.3 Exteroceptive traversability analysis

As was stated early in this chapter, the *Exteroceptive traversability analysis* can be distinguished in two subfields; *Geometry-based traversability analysis* and *Appearance-based traversability analysis*. In the following sections we are going to take a look on these two subfields. Like we did in the previous chapter, for the Proprioceptive traversability analysis, we are going to observe how the seminal ideas from each subfield evolved to what is currently considered as the state of the art.

2.3.1 Geometry-based traversability analysis

The geometry-based traversability analysis methodologies are the most commonly followed ones in the field of Terrain Traversability Analysis. The basis of their methodologies is building a terrain model and deriving a set of features from a given terrain according to this model. On top of this model, more sophisticated processing can be carried out, by adding a robot model and various kinematic and stability constraints to the problem's representation. There are three common approaches in building a terrain model; (i) *signal processing* of the terrain signal, (ii) the terrain signal's *convolution with a kernel* simulating the underlying vehicle and (iii) *statistical processing* and extraction of moment-based fea-

tures and certainty assessments. We will also discuss another approach, (iv) the utilization of *robot-dependent variables*.

2.3.1.1 Signal processing

Signal processing techniques are not as popular as the other approaches. They use signal processing methods to determine a terrain roughness and classify the terrain based on its roughness. They can be decomposed into two major branches; the *single-scale space* and the *multi-scale space* signal analysis. One of the first approaches in the single-scale space branch is the work of Hoffman and Krotkov [26]. This work is still influential towards modern approaches like the works of Castelnovi et al. [11] and Jun et al. [34]. Similarly for the branch of multi-scale space analysis, the work of Pai and Reissell [53] is cited by many modern works like the ones of Pivtoraiko et al. [56], Stelzer et al. [66] and Hauer et al. [24].

2.3.1.2 Convolution with kernel

A growing trend in terrain traversability analysis is the simulation of the vehicle as a fixed-size 2D kernel which convolves with the 2D terrain map. This is where the convolution with kernel approach proves really useful. The terrain map is iteratively processed by superimposition of a window, which can be either radial or rectangular and is centered at the total set of discretized positions and potentially at different orientations, so that it simulates the overlay of the vehicle on top of the terrain and computes the terrain's features at these positions. [54]

2.3.1.3 Statistical processing

The most popular and extensively studied approach for terrain traversability analysis evolves around the statistical processing of the 3D information that can be collected from a terrain surface. The main idea behind geometry-based traversability analysis based on statistic processing is constructing 2D traversability grid maps by computing various statistical properties of terrain regions using the data of a 3D representation and comparing these properties with vehicle-dependent thresholds. The works of Langer et al. [40] and Gennery [21] are great demonstrations of the above idea. Langer et al. [40] constructed a 2D traversability grid map by computing elevation statistic from the set of 3D points residing within each grid cell. Gennery [21], described terrain traversability using a cost function that takes into account the terrain region's elevation, slope, roughness and data point accuracy, as well as, the total distance of the path traveled and the probability of traversability. These two works are cited in many papers describing modern methods. Langer et al. [40] is mentioned in the work of Kelly et al. [36]. A really interesting aspect of the Kelly et al. work is that during the path planning they do not just take into consideration obstacles that can be pushed aside, like bushes, but they also take into account *support*

surfaces, meaning surfaces that are judged to be dense enough for a specific robot to move over them.

The works of Balch and Arkin [6] and Saska et al. [59, 60] are also worth mentioning. They focus on the UGV-formation navigation utilizing a *leader-followers* approach, with the leader being one of the formation's robots Balch and Arkin [6], a UAV Saska et al. [59] or a virtual entity Saska et al. [60]. The work of Gennery [21] is mentioned in the works of Angelova et al. [3], Angelova et al. [4] and Tarokh [70]. The research presented by Angelova and her team is focused on adding the UGV's wheel-slip learning and prediction on the path planning process. The main idea is learning the mechanical slip output from visual information, collected by the robot's vision sensors while the robot is still away from the terrain region of interest. The robot may create a wheel-slippage estimation by classifying the terrain's image based on the dataset with which it has been trained. While the robot drives over a terrain region a slip measurement is taken from its on-board sensors, so that the robot's training in predicting wheel slippage is continuous. Tarokh's work focuses on the development of a hybrid intelligent path planner which includes an adaptation procedure for adjusting the probabilities of the genetic operators, which are the operators used in genetic algorithms to guide the algorithm towards a solution to a given problem Dulay [16]. The two most common approaches for deriving the traversability of the terrain based on statistic processing are *by modeling uncertainty and error* and *by extracting shape features*.

2.3.1.3.1 Modeling uncertainty and error

This is a relatively modern field, created to deal with the weakness of the deterministic systems at dealing with complex and uncertain environments. According to Papadakis [54], the works of Kavraki et al. [35] and Gennery [21] set the foundations of this field, which focuses on modeling the uncertainty in terrain perception probabilistically. This uncertainty is created by the error that may have been induced in existing estimations regarding a terrain's features, like its elevation, that can be attributed to inaccurate data gathered from the vehicle's sensors. A notable work in this field is this of Thrun et al. [72, 73] regarding the terrain analysis module of the vehicle "Stanley", which won the 2005 DARPA Grand Challenge by becoming the first robot to complete the 132 mile DARPA Grand Challenge course in California's Mohave Desert. We talk in some detail about "Stanley" and its accomplishment later in this survey.

Thrun's team work on "Stanley" can be considered monumental for the field of Terrain Traversability Analysis for UGVs and, as a result, both of his papers are often cited in newer works. For example, Thrun et al. [72] is mentioned by Martín et al. [45] and Oliveira et al. [50]. Martín et al. [45] work on detecting when a robot is navigating around a previously visited place to check the global error and to minimize it to give consistency to the global map. It achieves that by extracting the most important features from two different 3D laser scans in order to obtain an indicator that is used as a threshold to detect when the robot is visiting a known place. Oliveira and his team deal with the uncertainty of a mapping created by the presence of obstacles. To solve this problem they propose a method which fuses laser data with vision data, so that the mapping is not computed in

the regions where obstacles are present.

2.3.1.3.2 Extracting shape features

This approach employs 3D scene understanding algorithms and maps the difficulty of a vehicle in traversing a terrain region into distinct levels. Some approaches focus on locating 3D shape primitives like a line, a planar patch or a cluster of scattered 3D points in the 3D scene that represents a terrain region [76]. Other approaches focus on perceiving negative obstacle shapes in a terrain's 3D representation [25]. Such negative obstacles may be gaps, downward inclined planes or descending steps and may be non-traversable by a certain UGV. However, both the 3D shape primitives and the negative obstacles approaches are prone to errors in case of missing sensor data. They are also very limited towards the detail of their traversability assessments. These two disadvantages often lead to predictions based on a worst-case-scenario, thus categorizing as non-traversable terrain regions that may be passable under certain conditions, depending on the mobility capabilities of the UGV, as we understand from the Papadakis [54] review.

From this review, it is also evident that the field is still in its early stages, however, works have started to emerge that go beyond just simulating the vehicles as a symmetric kernel which convolves with the terrain. First, the robotic vehicle's configurations and shape started to be taken into account. Then, works emerged that focus into a robot's exact points of contact with the ground. Factors like wheel slippage and terrain inclination came into play, leading to the development of motion planners that consider alternate routes and weight them based on factors like energy consumption and time duration expected in order for a path to be traversed.

2.3.1.4 Incorporating robot dependent variables

By the previous subsections, we arrive at the result that when the traversability analysis is performed using solely characteristics of the terrain its results can be deceptive. The scientific community realized that and soon enough, the development of more detailed and case-specific models that take as input vehicle dependent variables started to be explored. However, the early researchers where limited by the computing capabilities of their time. As a result, their ideas where sidelined until recently, when the dedicated software for running models, such as simulation environments and physics engines, was proliferated and the necessary hardware became more affordable.

2.3.2 Appearance-based traversability analysis

As this approach's name suggests, in the appearance-based traversability analysis, a terrain region's traversability is judged based on its appearance. For example, there are approaches that measure a terrain region's traversability based on its roughness, slope, discontinuity and hardness [28]. These features are determined by computing the size, concentration and average separation distance of rocky regions within the observed area

that were identified by comparing their visual signature against that of the ground. Such approaches were followed by more refined ones, which also use color statistics to classify the terrain into soil, sand, gravel, asphalt grass and woodchips [4].

2.4 Hybrid approaches for traversability analysis

From the approaches, fields and methods presented so far, it is obvious that none of them is clearly superior. Actually, most modern approaches are what we call *"hybrid"*, meaning that they combine at least two main categories of sensor data, like LIDAR and vision sensors, to perform terrain traversability analysis.

2.4.1 Fusion of geometry-based and appearance-based features

The fusion of geometry-based and appearance-based features is the main stream of research in hybrid traversability analysis methodologies. Instead of using a single representation of fused features, practitioners most often assign distinct roles to each sensory data type for assessing different kinds of terrain traversability. An example is the work by Bellutta et al. [7], where terrain perception is based on the combination of geometric and visual features through a rule-based system. Terrain was geometrically classified into negative or positive obstacles by inspection of the height profile of elevation data while the terrain support was statistically learned through Expectation Maximization within the color space.

Manduchi et al. [44] builds on top of Belluta's Obstacle Detection algorithm and presents an algorithm that works directly with measured 3-D point clouds. Obstacles are detected by measuring slope and height of visible surface patches, an operation that can be carried out directly in the range image domain. For terrain classification, a combination of data coming from color cameras and a single-axis ladar is used, as well as statistic processing in the background, with Gaussian Mixture Models.

One year later, we find Kelly et al. [36], an interesting work that combines many things we have mentioned so far. We see terrain mapping and navigation methods not constraint in the domain of UGVs but also applicable for UAVs. Discrete obstacle and support surfaces detection are among them. Again, for the environment's perception a combination of image and geometrical data is utilized. In another work of 2006, Kim et al. [38], again using image and ladar data, infer terrain traversability based on some key assumptions. One of them is that visual features derived from stereo vision and color imagery are sufficient to discriminate between terrain regions from the standpoint of the traversability affordance. Specifically, they assume that two terrain locations which produce similar feature values will have similar properties of traversability, but, they point out that this does not imply that all traversable terrain locations look similar, as there is no simple description of traversable regions in terms of image appearance or geometry.

In a much more modern work, we see Nissimov et al. [49] to be using color and depth information provided by a Kinect sensor to distinguish obstacles from non-obstacles for

UGV suited for agricultural applications. They improve the quality of their results and deal with alignment errors between color and depth images using various configurations and two classifiers, a color and a texture classifier.

2.4.2 Heterogeneous sensor fusion

The idea of heterogeneous sensor fusion comes to build on top of the aforementioned ideas regarding the fusion of appearance-based and geometry-based features. The depth information and the various visual cues are fused with data deriving from sensing the environment's temperature, humidity and precipitation.

Many creative approaches have been published over the years. Dima et al. [14] combined the appearance and geometrical data with (near)-infrared imagery. This way, a system that does feature and classifier fusion for obstacle detection and terrain traversability was presented. In Silver et al. [62] we see the typically collected appearance and geometry data being combined with data originated by the robot observing and analyzing its teleoperation by an *expert*. All these three factors are taken into account for the formulation of global/local planning. In the work of Kelly et al. [37], we see the presentation of a system that aims to aid the teleoperation of an unmanned ground vehicle. By fusing data from a lidar sensor, stereo cameras and a FLIR (Forward Looking Infrared Sensor), a virtualized reality is created, which enables the operator to control the robot by having a complete view of the terrain region in which it moves (forward, left, right, backward), not limited by a single sensor's field of view.

Anand et al. [2] uses imagery data coming from RGB-D sensors to spot commonly found objects in the 3D point cloud of indoor scenes. In the paper of Sullivan et al. [69] we see a terrain classification architecture which relies in data taken from 3D point clouds combined with data from RGB images. What's interesting is that they also use probability distribution in some of their more sophisticated classification algorithms. A year later, Milella et al. [46] propose a self-learning scheme for ground classification, using the platform's camera and radar sensors. This is implemented by their self-supervised classification framework, in which the camera's visual classifier is trained under the supervision of the radar's classifier that in turn is self-taught.

2.4.3 Comparative studies and surveys

Over the years, various studies and surveys have been published. One worth mentioning here is Halatci et al. [23], which is a study of fusion methodologies. It includes an in-depth evaluation of features from different sensing modalities, fusion strategies and classification methodologies, and the application domain is planetary rovers and terrain classification into "rocky", "sandy" and "mixed". Again, we see appearance-based features, in the form of color data, been combined with geometry-based features, in the form of wavelet-based signatures, while statistical tools like Gaussian Mixtures and Bayes Fusion assist the whole process by helping the combination of classifiers data.

2.5 Discussion

Now that we have talked about the main streams of research in the field of terrain traversability analysis for UGVs, we can compare them and we can take a look at the theory's impact to real world applications.

2.5.1 A comparison

As it has already been stated many times, there is no best approach in terrain traversability analysis. However, in practice, certain methods have been definitely proved more successful than others, at least in the current state of the art. For example, the research community realized early on that solely *proprioceptive terrain traversability analysis* is not a viable approach for most robotic platforms, because it requires them to be robust to collisions and tipping over and can often leave the robot vulnerable to reaching an unrecoverable state. However, the proprioceptive-based methods were not tossed aside, as we see them making appearances in more modern works, trying to deal with issues like slipping prediction during motion planning, in order to complement a generally exteroceptive-based motion planning module.

The field of the *exteroceptive terrain traversability analysis*, by itself, is very big. We see a wide variety of methods and approaches, most of them inspired by concepts of either Signals or Probabilities theory. Their goal is commonly to make a traversability estimation from a safe distance using visual or radar data and adjust the UGV's planned path accordingly. As the years go by and the LIDAR and radar technologies become more affordable and open to everyone more and more research groups incorporate them in their projects, because of these sensor abilities to capture range and angle data simultaneously, in contrast to the classic camera sensors, where additional processing is required. Still, despite that the purely exteroceptive-based approaches produce better results than the proprioceptive-based ones, false estimations is a big issue, because many times possible routes are disqualified based on some worst-case scenario, whereas they may be traversable under certain conditions in regards to the environment or the robotic platform.

And so, we arrive in the current state of the things, where the most popular and successful approaches are *hybrid*. Initially, the camera (appearance) and lidar (geometry) data started being fused for the formulation of traversability estimations. Then, more sensors started being used, trying to boost the traversability estimation's accuracy by taking into account factors like the environment's temperature. Ultimately, hybrid approaches were incorporated for the navigation of historic vehicles like NASA's "Spirit" and "Opportunity" [47] and Stanford's "Stanley" [71], that pushed the boundaries of the terrain traversability analysis field and set some high standards for future research.

2.5.2 Real world approaches

We conclude our discussion with a short presentation of *Boston Dynamics* and *iRobot*, two companies with continuously growing and, already, great impact in the Robotics Design in-

dustry. We will also see the cases of NASA's *Spirit* and *Opportunity* rovers and Stanford's *"Stanley"*, three vehicles that really pushed the boundaries of the Terrain Traversability Analysis field and set some high standards for the future's great Unmanned Ground Vehicles.

2.5.2.1 Boston Dynamics

Boston Dynamics is an American engineering and robotics design company. It is considered a pioneer in the field of robotics and it is one of the most advanced in its domain, combining the principles of dynamic control and balance with sophisticated mechanical designs, cutting-edge electronics, and software for perception, navigation, and intelligence. It began at the Massachusetts Institute of Technology, where it developed the first robots that ran and maneuvered like animals. Boston Dynamics [10] Marc Raibert, company's president and project manager, spun the company off from the Massachusetts Institute of Technology in 1992. Boston Dynamics is best known for the development of BigDog, a quadruped robot designed for the U.S. military with funding from Defense Advanced Research Projects Agency (DARPA), and DI-Guy, software for realistic human simulation. On 13 December 2013, the company was acquired by Google X (later X, a subsidiary of Alphabet Inc.), where it was managed by the Android OS creator, Andy Rubin, until his departure from Google in 2014. Immediately before the acquisition, Boston Dynamics transferred their DI-Guy software product line to VT MÄK, a simulation software vendor based in Cambridge, Massachusetts. On 8 June 2017, Alphabet Inc. announced the sale of the company to Japan's SoftBank Group. [43]

2.5.2.2 iRobot

iRobot Corporation is an American advanced technology company founded in 1990 by three MIT graduates who designed robots for space exploration and military defense. Incorporated in Delaware, the company designs and builds consumer robots for inside and outside of the home, including a range of autonomous home vacuum cleaners (Roomba), floor moppers (Braava), and other autonomous cleaning solutions. It is considered to be the leading global consumer robot company, with a vision to design and build robots that empower people to do more both inside and outside of the home. [30]

2.5.2.3 NASA, the Spirit and Opportunity rovers for Mars exploration

NASA's *Mars Exploration Rover (MER)* mission is an ongoing robotic space mission involving two Mars rovers, Spirit and Opportunity, exploring the planet Mars. It began in 2003 with the launch of the two rovers: MER-A Spirit and MER-B Opportunity to explore the Martian surface and geology. Both rovers landed on Mars at separate locations in January 2004 and, since then, they have far outlived their planned missions of 90 Martian solar days. MER-A Spirit became stuck in soft soil in late 2009, and its last communication with Earth was sent on March 22, 2010. MER-B Opportunity is still active in 2018,

however, at the time of writing this thesis, due to dust storms it has entered hibernation mode which is expected to last for several weeks Good [22]. Opportunity holds the record for the longest distance driven by any off-Earth wheeled vehicle. Its mission highlights include the initial 90 sol mission, finding extramartian meteorites such as Heat Shield Rock (Meridiani Planum meteorite), and over two years studying Victoria crater. It survived duststorms and reached “Endeavour” crater in 2011, which has been described as a “second landing site”. Fitzpatrick [19] With data from the rovers, mission scientists have reconstructed an ancient past when Mars was awash in water. Spirit and Opportunity each found evidence for past wet conditions that possibly could have supported microbial life. Opportunity’s study of “Eagle” and “Endurance” craters revealed evidence for past interdune playa lakes that evaporated to form sulfate-rich sands. The sands were reworked by water and wind, solidified into rock, and soaked by groundwater. [48]

2.5.2.4 Stanford’s “Stanley”

“Stanley” is an autonomous car created by Stanford University’s Stanford Racing Team in cooperation with the Volkswagen Electronics Research Laboratory (ERL). It won the 2005 DARPA Grand Challenge, earning the Stanford Racing Team the 2 million dollar prize. Thrun et al. [73] The challenge consisted of building a robot capable of navigating 175 miles through desert terrain in less than 10 hours, with no human intervention. On October 9, 2005, “Stanley” became the first robot to complete the 132 mile DARPA Grand Challenge course in California’s Mohave Desert, finishing in just under 6 hours 54 minutes and averaging over 19 mph on the course. [64]

2.5.2.4.1 About the robot

“Stanley” is based on a stock, Diesel-powered Volkswagen Touareg R5, modified with full body skid plates and a reinforced front bumper. Stanley is actuated via a drive-by-wire system developed by Volkswagen of America’s Electronic Research Lab. All processing takes place on seven Pentium M computers, powered by a battery-backed, electronically-controlled power system. The vehicle incorporates measurements from GPS, a 6DOF inertial measurement unit, and wheel speed for pose estimation. While the vehicle is in motion, the environment is perceived through four laser range finders, a radar system, a stereo camera pair, and a monocular vision system. All sensors acquire environment data at rates between 10 and 100 Hertz. Map and pose information are incorporated at 10 Hz, enabling Stanley to avoid collisions with obstacles in real-time while advancing along the 2005 DARPA Grand Challenge route. The development of Stanley began in July 2004. At the time of the initial team application, the vehicle is largely functional and has logged dozens of autonomous miles along the 2004 DARPA Grand Challenge course. [65]

2.5.2.4.2 Modeling uncertainty and error

“Stanley’s” terrain analysis module is a great real-world example of the utilization of statis-

tic processing to model uncertainty and error. The traversability is formulated as a probabilistic feature wherein locally spaced elevation differences in the acquired 3D point cloud were verified or discarded, through an error probability estimate that designated whether a position corresponded to an obstacle, drivable space, or unknown. The probabilistic terrain analysis took into account errors occurring in the robot's state estimation and the noise induced from the laser sensors and accumulated over time. [54]

2.6 A glimpse into the future

Now that we have seen what is the current state-of-the-art in terrain traversability analysis, we conclude this survey by taking a brief look to where things are going. In general, we see that most research teams conclude their papers by pointing out either the need to validate the results that they produced in simulation or in "controlled" environments in real-world scenarios, or the need to further expand and test their method for more challenging use cases. From the works presented so far, Castelnovi et al. [11], Jun et al. [34] and Stelzer et al. [66] fall in the former category, while Kelly et al. [36], Kelly et al. [37], Sullivan et al. [69], Peynot et al. [55] and Jonschkowski and Brock [32] fall in the later category.

Some interesting ideas about the future can be derived from the work of Hoffmann et al. [27], where they say that they plan to test their methods on a variety of UGV platforms, both legged and wheeled with a similar sensory set and they also plan on investigating how modifications in a robot's morphology, like the weight distribution or the stiffness of the springs in the knee joints, can affect the terrain discrimination. Jonschkowski and Brock [33] say that they plan to investigate state representation learning across related tasks, something similar to real-life's "transferable knowledge" for humans. Another work worth mentioning at this point is Bengler et al. [8]. It is a review of the three decades of driver assistance systems, mentioning many things we saw above, like Thrun's team work on "Stanley". It points out the need for further training the robotic driving systems by having them observe a human operating their vehicle, to gain a mapping and an understanding of driving as close as possible to that of an experienced human.

3. INCLINE TERRAIN TRAVERSABILITY FOR UGVS

In this chapter we are going to talk in greater detail about the problem of incline terrain traversability specifically. First of all, we will go through the main considerations while discussing a robotic platform's capability of traversing a terrain of certain incline. Then, we will talk about the *"Traversability Estimation"* software developed by the ETH Zurich's Autonomous Systems Lab. This will lead us in examining possible weaknesses of the current software and ways that it can be used, as it is, to improve the outdoor terrain navigation capabilities of an existing robotic platform. We will also see the *"Adaptive Feedforward Controller"* software developed by TU Darmstadt's "Cooperative, Adaptive and Responsive Monitoring in Mixed Mode Environments" Research Training Group and we will discuss about its applications potential regarding the problem of incline outdoor terrain traversal, as well as its limitations.

3.1 Main considerations for incline terrain traversability

We will talk in some detail about how incline terrain traversability is affected by factors concerning the robotic platform's and the terrain's physical properties.

The incline terrain traversability is perhaps one of the most important fields of problems regarding unmanned ground vehicles. Its advancement is going to be crucial for the UGVs to reach the full potential of their applications as it will enable them to move safely and autonomously in unstructured outdoor terrains. Naturally, there is not only one true solution to the outdoor terrain traversability problem, as the type of the terrain (step, roughness, slipperiness, etc.) the type of the UGV (biped, wheeled, etc.), the platform's dynamics (center of mass, wheel/foot width, etc.), even the UGV's mission requirements may dictate certain constraints or may give certain degrees of freedom for one to design an effective approach to this problem.

In this thesis we are going to focus on how a wheeled UGV of relatively small size (0.3m x 0.275m) may climb a terrain of steep incline. Factors like the terrain's step, roughness and slipperiness will be left to be addressed in future work and we are going to focus on the terrain's slope and how we can compensate for it by adjusting our motion's path planning appropriately.

3.2 The ETH Zurich's "Traversability Estimation" framework

We find a great summary of the ETH Zurich's work at the Wermelinger et al. [79]. This paper presents a framework for traversability estimation and path planning, especially suited for legged robot navigation. The framework's parameters can be tunned to fit a certain robotic platform's specific characteristics, while the traversability of a terrain region is estimated based on data deriving from onboard sensing. These data are gathered in a discrete 2.5D elevation map, from which features regarding the terrain's slope, roughness

and steps can be extracted by applying different filters. A hierarchical path planning architecture is proposed, which utilizes a local traversability map to build and update a global planning map. A global path planner attempts to find a complete global solution path from the current robot's state to the desired goal state using this global planning map. Then, a short range planner tries to produce a valid local plan for the robot to execute, based on each segment of the global path planner. During this process, the validity of each segment is decided. Each valid segment is directly executed, however, for each invalid segment the global path is replanned.

3.2.1 The "Traversability Estimation's" software capabilities

The most up-to-date published component of the above framework can be found in the respective ETH Zurich's ASL Github repository [5]. There we can find a ROS package for traversability estimation, which utilizes the methods of the framework presented at Wermelinger et al. [79]. The package's software is highly configurable, so it can produce results that fit the specific needs and characteristics of any robot. It utilizes an elevation map to build the traversability map of a terrain region of specified dimensions around the robot. Such an elevation map can be found at ETH Zurich's ASL Github repository for Elevation Mapping [17], where a ROS package, deriving from the work presented at Fankhauser et al. [18], is stored. This traversability map can be further configured with filters regarding the terrain's slope, roughness and steps, like we talked in the short framework presentation above. The traversability map's information can be accessed at any time, either by listening to its respective ROS topics, like "/traversability_estimation/traversability_map" or by using one of the many ROS services that it includes.

3.2.2 Expected weaknesses of the "Traversability Estimation" framework

The "Traversability Estimation" framework is really well-designed and well-developed, it can be definitely considered as state-of-the-art. However it can display all the weaknesses expected by a software that derives an image of a situation on-the-fly, depended on the sensors of the platform that is using it. The traversability image created can have either a low or a high resolution.

In case of a lower resolution, updating this image will have a low processing cost, which is really important for a small platform that wants to be in motion for an extended period of time. However, in real world scenarios misclassification of terrain regions is going to be an issue which is going to lead in wrong path planning decisions. This is especially true when the resolution lowers to the point that the area of a cell of the traversability map's grid tends to approach the area of the robotic platform as viewed from above. In case of a higher resolution, updating the map is going to require much more computational power. In UGV, which is a small robotic platform that wants to move autonomously for an extended period of time, the higher-res traversability map is going to be more accurate, but the robot will have to choose between delaying its motion until the map is updated or planning ahead and moving based on an outdated map, which has a high risk in outdoor terrains, which are generally unpredictable.

As mentioned before the traversability map's creation process is highly depended on the robotic platform's sensors. This has the advantages of being realistic and minimizing the risk of misinterpreting the terrain's image, it does however limit the robot's path planning to the range of its sensors and leaves it vulnerable to possible misses or other technical issues of its sensors.

As a conclusion, the "Traversability Estimation" framework, as advanced as it is, it cannot be considered as a viable option for outdoor path planning, at the time of writing this thesis.

3.3 The TU Darmstadt's "Adaptive Feedforward Controller" software

The *"Adaptive Feedforward Controller"* (AFFW control) project, described in the works of Ommer [51], Ommer et al. [52] implements a framework for an adaptive feedforward controller for mobile platforms. The controller works on top of existing motor models and learns a compensation model during runtime, without any prior knowledge, to counteract non-modeled disturbances such as slippage or hardware malfunctions.

3.3.1 The "Adaptive Feedforward Controller's" capabilities

The AFFW controller's main goal is to reduce the error between target velocity and actual measured velocity. This way, the controller is able to prevent motion errors a priori and is well suited for real hardware due to high adaptation rate. It can be used in conjunction with any motion model as only motion errors are compensated. Different online learning methods have been implemented in the controller's software package, including Locally Weighted Projection Regression (LWPR), Sparse Online Gaussian Process (SOGP) and Recursive Least Squares (RLS). As we can see in the paper in which Ommer et al. present the controller's framework [52], during its development, the framework was applied to two very different robot systems across the RoboCup Leagues. Although those robots were underlying distinct mobility challenges, the controller was able to improve locomotion performance for each system. The project's code can be found on its Github repository [74].

3.3.2 Expected weaknesses of the "Adaptive Feedforward Controller"

The AFFW controller is a great and original idea, its software package represents a well-written concept, however at the time of writing this thesis it is not well-documented enough and it hasn't been updated for over two years. So, as impressively as it may have performed in the Team Hector robots for which it was developed, it can't be expected to perform similarly on every other UGV. Also, unlike the ETHZ's ASL "Traversability Estimation" framework, it is difficult to be adapted to a specific platform and it is highly doubtful that the weaknesses and flaws that it is naturally going to display in real-world usage will ever be patched or corrected. The AFFW controller comes to assist the robot's motion during runtime. This is useful as an extra layer of safety for a UGV, however only by itself

it is not enough to guarantee the vehicle's safety in an unstructured and unpredictable outdoor terrain region, as it does not interact with the robot's path planning system.

As a conclusion, the AFFW controller, is a great concept however it is not mature enough yet and it can't guarantee the robot's long-term safety when operating standalone.

4. PATH GENERATION ALGORITHMS

In this chapter, we are going to take a look on the basic principles of the *Hill Climbing, N-Best and Evolutionary Algorithms* and discuss possible ways in which each one of them may benefit an implementation which will try to face the incline terrain traversability problem.

4.1 Why we need the Hill Climbing, N-Best and Evolutionary Algorithms

As we will see later on, the incline terrain traversability problem can be approached at an offline path planning level, by trying to produce Bezier paths, which will provide the robot with a smooth curved path to its goal, at the top of the incline. We are going to use the Hill Climbing, N-Best and Evolutionary Algorithms in order to produce these Bezier paths in three different ways, by taking into account both the production's efficiency and the paths' safety, meaning that they do not go through lethal obstacles.

The general idea is that we view the segment of the problem's field between the start and the goal positions as a grid. Each one of this grid's cells can be considered as the node of a graph that spans between the start and goal positions. Our task can be reduced into finding the lowest-cost path from the start to the goal. The cost of each edge of the graph is assigned in a way that leads our search into producing smooth curved paths.

4.2 Hill Climbing algorithms

Hill Climbing is a mathematical optimization technique and is a local search method. During solving a problem, Hill Climbing is guaranteed to find local optima solutions, which in the end constitute a global "good enough" solution. It is a relatively simple algorithm and this makes him a very popular choice for creating an initial approach to an optimization problem, in order to test a new idea or a new method. In the field of Artificial Intelligence it is widely used to reach a goal state from an initial state [67].

In the context of this thesis we are going to see a variant that can be categorized as a *Steepest Ascend Hill Climbing*. Steepest Ascend Hill Climbing algorithms is a subset of the Hill Climbing algorithms which operate in a best-first manner. While, at a locally optimal solution, they search all the successor states, then they select the best successor state and continue solving the problem from this state.

Generally, such an approach has the benefit of reducing the number of local searches as much as possible, while making sure that the end result will be as good as possible. In our problem, this is very useful, as finding an optimal path from start to goal with a greedy algorithm that does an exhaustive global search, will be a very slow and computationally expensive process, whereas we can find a "good enough" path with a greedy algorithm that does many consecutive exhaustive local searches that lead to locally optimal solutions.

However, Hill Climbing algorithms have a very important shortcoming. While the algo-

rithm is efficient, as it delivers in a very short time a solution whose quality is relative to the freedom that it was given to search the problem's space, it is by no way safe. If one of its locally optimal solutions coincides with a lethal obstacle, which makes the solution essentially prohibited, it has no ability to come up with a relatively close, but safe, alternative. So, in such a case, the planning process is going to fail. This gives us the motivation to improve the Hill Climbing algorithm, by keeping its core logic and its core functionality, but providing him with the structures and the logic that will help him come up with an alternative solution in case that a plan fails. And this is where the *N-Best algorithms* come into play.

4.3 N-Best algorithms

It is obvious now that in order for the Hill Climbing algorithm implementation to be applicable to a substantial range of real world scenarios, the capability to keep some alternative local plans which are as close to optimal as possible is necessary to be added. For this purpose, almost intuitively, an *N-Best* algorithmic approach seems to be the way to go. So, to enhance the functionality of the Steepest Ascend Hill Climbing algorithm derivative which we discussed above, in a way that will enable him to be considered an N-Best algorithm, we need to draw inspiration from the work of Chow and Schwartz [12]. In their work, Chow and Schwartz use an N-Best algorithmic approach to generate, in a clean and efficient way, a series of alternative hypotheses, more specifically N alternative hypotheses, regarding the meaning of a spoken natural language sentence. What we need is a clean and efficient way to generate N alternative paths between two given positions. In each step of the planning, we are going to take these paths in a most-optimal-first order and in case our choice fails in the next step of the planning, we can take a step back, choose the next available path and see where this leads us. We can do this as many as N times.

So, the N-Best approach that we are going to see later in this thesis is basically a greedy, best-first approach, that keeps a list of alternatives in each step of the planning, in which it can fall back in case the next step of the planning fails. This can be vital if we take into account lethal obstacles that may exist in an incline terrain region, which are obstacles that have the potential to leave our robot in a damaged or irrecoverable stage once it tries to pass from them. Such lethal obstacles may be rocks which will flip our UGV over, tree trunks or holes in which one of the robotic platform's wheels may stuck, or worse, the whole UGV may fall into the hole and be unable to climb out. In case such an obstacle's coordinates are chosen as part of the generated path, the Hill Climbing algorithm presented in the previous section will have to stop its execution and report a failure. But, if we use an N-Best approach, we will have N-1 alternatives in our disposal, which we will be able to try in a best-first order and see if they get us through the lethal obstacle.

That being said, there is still a way for the N-Best algorithm to fail and terminate its execution without having produced a viable path from the start position to the goal position. Admittedly, one of the N-Best algorithms' most important disadvantages, which kind of jumps out of the page when we talk about them, is that they keep just the *N-Best* alternatives. So, in case of a complex lethal obstacle, like a rock formation or a clump of bushes, it is highly possible, depending on our problem's search space parameters, that in the end

the search is going to fail, because none of the N-Best possible choices will be able to get us safely around the lethal obstacle. It would be really useful if, instead of being limited to just the N best choices, we could keep searching until we had a viable path from start to goal, without sacrificing much of our need for efficiency. Fortunately, this can be achieved by using an *Evolutionary Algorithm*.

4.4 Evolutionary Algorithms

Evolutionary Algorithms are a product of the field of *Evolutionary Computation*. This field was created by the inspiration to use biological evolution theories and methods to solve highly complex problems. For this to happen, the subfields of Artificial Intelligence and Soft Computing that study biological evolution combined their ideas. The field of Evolutionary Computation utilizes population-based trial and error algorithms and carefully thought-out heuristic functions to solve problems that seem impossible to be solved optimally, or at all, with a more conventional way of thinking. During this repetitive trial and error process many operations inspired by the Evolutionary Biology, like mutation and crossover, are applied to a potential solution to the given problem. These operations drive our search for an optimal solution, which is essentially a process in which random, commonly suboptimal, solutions are evolved into optimal, or "optimal-enough", solutions depending on the termination criteria that we have given our algorithm [16], [77].

So how can life's evolution mechanisms be applied to a path planning problem like the one that we study in this thesis? What we hope to achieve by using an Evolutionary Algorithm is to efficiently arrive in an as-good-as-possible solution to a complex problem, like planning a path through an incline terrain region filled with complex obstacle formations. Like a common evolutionary process, we are going to start our search from a first generation, which is an initial set of suboptimal individuals. These individuals will be our initial paths and each one of their chromosomes, which will be their waypoints, will have been chosen randomly. We will then enter a repetitive process in which the best individuals of each generation are chosen to participate in crossovers, thus becoming the parents of the next generation, which is hopefully going to bring us closer to a satisfactory solution to our problem. Some of the children that are derived from this process may also be subjected to mutations, in case we want to ensure a relatively high degree of biodiversity. However, this is recommended to be done in caution, as it has the potential to lead into good but unrealistic paths. Such paths may be going against the robotic vehicle's dynamics and their detection and elimination may add unwanted complexity to our algorithm. Theoretically, like the human evolution, the evolutionary process described so far is never going to end, so we have to define some termination criteria. Such criteria may be reaching to a generation that contains an individual which is "good-enough" for our purposes or observing a performance stagnation for a number of consecutive generations, meaning that the evolution between them is not as radical as we want it to be [16], [77].

From the above we understand that the Evolutionary Algorithms are a great tool that can help us face tough problems, with vast search spaces, efficiently. We start with a random, "not-good-enough" approach of a solution and we gradually evolve this approach into an admissible solution. It is really as simple and intuitive as it sounds and it can prove as

powerful of a mechanism as the human evolution. Their only shortcoming is that their efficiency is greatly depended in the complexity of the function that we use to judge the fitness of each individual of a generation, thus deciding whether he can act as a parent to the next generation or not. This shortcoming is normally faced with the application of fitness approximation techniques, however, since the evaluation functions that we use in this thesis rely into simple criteria and have low complexity, we are not going to expand our discussion on this subject.

5. CLIMBING A HILL WITH A WHEELED UGV

In this chapter I am going to present my approach to solving the problem of a UGV traversing a terrain of steep incline. As it has already been stated, this problem is open to many approaches, each one with its own strengths and weaknesses. After studying the current state of the art in the field of Terrain Traversability Analysis, I decided to develop an approach that tries to deal with the problem in a path planning level. Our path's planning is going to be concluded *offline* meaning that our path is going to be complete and final before the robot starts following it. This means that before the robot starts moving we would have already decided the whole motion with which it will try to approach its goal from its start position, having taken into consideration all the available information relevant to our problem.

My approach evolves around determining a list of waypoints which constitute a path created by stitching many smooth curves together, and sending these waypoints to the robot's move_base so that it will execute them as desired. The move_base is one of our middleware's components and is the implementation of one of the most basic actions that a moving robot can perform. This action is that given a goal in the world, under a specific frame of coordinates, our middleware will attempt to reach it with our robot and, in the process, will take care of the most basic technical details, like moving a UGV's wheels the proper way. This approach has the potential to be extremely useful in an unstructured but nevertheless stable outdoor environment as it is not directly depended on the robotic platform's sensors and processing power and it produces only one, definite plan and lets the robot deal with its execution. The approach is very clear, it can be easily adapted to any UGV and it can benefit other smart systems applied on the path that it produces to guarantee its safe execution.

5.1 Problem set-up

I developed my approach in a simplified version of inclined terrain. We do not take into account the terrain's step, roughness or slipperiness, so that we can focus on how to deal with its relatively large slope ($> 40°$). The robot's start position is in the beginning of the incline terrain region, where the robot is in a neutral and balanced state, facing straight upwards, which in a real world scenario will be the moment when the robot realizes that faces the challenge of a terrain with steep incline and triggers the mechanism that utilizes my approach. The robot's goal position is normally in the end of the incline terrain, far enough from the start position so that we can clearly see my approach in action and understand its usefulness.

It is considered that the robot already knows some vital facts about the task that it has to perform, like the degree of the terrain's slope, the initial position, which is its position, the goal position and the boundaries inside which it is allowed to search for solutions and move. After a certain stage, lethal obstacles are added into the robot's realm of knowledge, from which the robot has to survive while trying to reach the goal. As the lethal obstacle formations get more complex, I present more sophisticated versions of my

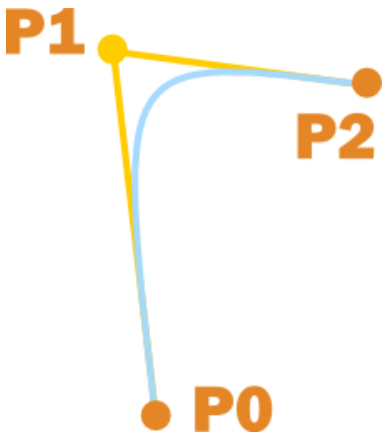


Figure 1: A sample of a quadratic Bezier curve [75]

approach, which aim to help the robot survive these obstacle formations. The acquisition of this data may vary depending on the robotic platform and, as it is admittedly a big topic of robotics by itself, it is considered as being outside the scope of this thesis. It is also important to note that in the context of this thesis the UGV uses only its odometry to localize itself, which in simple terms is "how much its wheels have turned". This is the minimal knowledge that a wheeled robot can have relatively to its state in the *world* in which it operates.

5.2 Path generation criteria

We want to create a path which can be described as a series of smooth curves stitched together. This way the robot will be able to ascend the incline terrain without tipping over or sliding backwards, as its platform pitch will be always kept as low as possible. The smooth curves that make up the path are quadratic Bezier curves, at times sub-optimal, depending on the selected search step between their control points. Each path can be described by a total cost which is the sum of the cost of its control points. This cost is created by taking into account the estimated deviation from the straight-line course, the estimated roll and pitch of the robotic platform and the estimated arc relative to the terrain slope between three consecutive control points, with the control point at question being in the edge. To these estimations certain weights, which have been determined experimentally, are assigned.

We treat the path generation problem as an optimization problem, and more specifically a minimization of the path's total cost problem. However, creating a globally optimal path will be very expensive computationally, especially for a simple UGV, without a GPU to assist it with parallel processing, which still has to be able to operate autonomously. That's why we focus on creating a "good-enough" path by taking a series of locally optimal control points decisions and stitching them together. It is worth noting that not all of the criteria mentioned in the above paragraph affect the vehicle's effort to climb the incline terrain the same way. More specifically, we expect the pitch to toughen the effort as it increases, but in contrast we expect the roll to ease the effort as it increases. We also do not want the

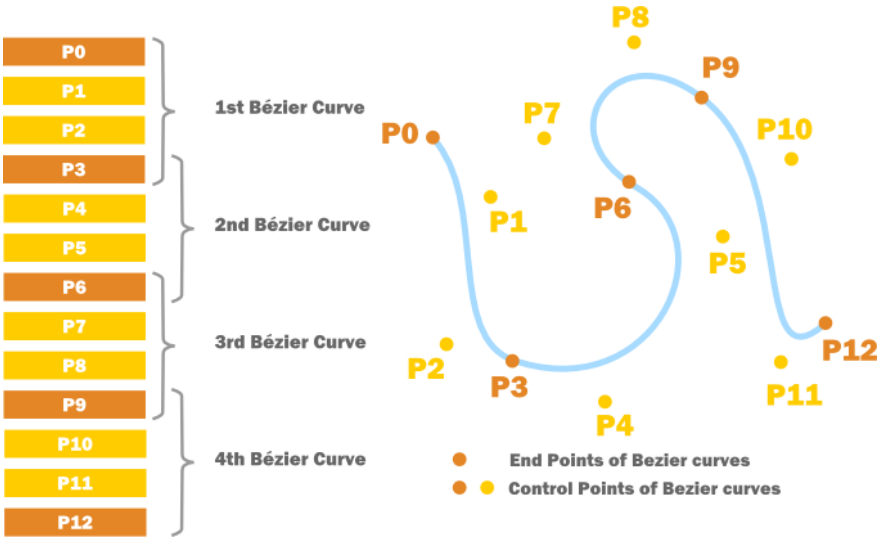


Figure 2: A sample of a stitching Bezier curves to form a Bezier path [75]

vehicle to deviate a lot from the straight-line course, as it may create a route that is longer and less time efficient than what is necessary, and we do not want the arc between three consecutive control points to be small in a steep incline, as it is possible for the UGV to slide backwards and lose a part of its progress and its localization, which will endanger the whole effort. For this reason, the roll estimation has a negative weight, while the other criteria have a positive weight.

The formula that evaluates a path is the following:

$$cost = 3.5 * (45.0/terrain\_slope) * \sum norm\_deviation \; + \; 10.0 * \sum pitch \; - \; 10.0 * \sum roll \; + \; (terrain\_slope/10.0) * \sum arc$$

It is worth noting that we sum the deviation normalized relatively to the straight-line distance between the start and the goal position.

5.3 Path generators

The first path generator which is going to be presented is a greedy but very efficient Hill Climbing approach. The second is a much safer and still efficient enough N-best search approach. The third is an Evolutionary Algorithm approach, the most involved and safe for real-life operations of all the presented approaches.

5.3.1 Hill Climbing generator

The Hill Climbing generator is not only the simplest implementation of the presented waypoints method included in the **ugv_navigation_goals** [1] package, but it is also the basis on top of which the rest of the implementations build. We treat the optimum path generation problem as the problem of traversing a graph by finding the minimum cost path between two certain nodes. This graph is created by fitting a grid in the area between the start and the finish positions. Each cell of the grid is a node of the graph. The grid is described by a search step which can be modified according to the user's needs, however it is recommended to be kept as close to the robot's length as possible. This is because, each waypoint will take a certain time to be transmitted to the robot's move_base and to be reached by it and if we have waypoints very close to each other we risk at missing some of them because of differences in the transmission rate of our node and the move_base or localization error. This localization error should always be taken into account, however small it may be, especially if we are designing for a vehicle that uses its odometry to localize itself.

The p_0 of the first local Bezier curve is the start position. We search the incline area's grid by taking two rows of it at each step. Greedily, we try to place the p_1 control point, which is not part of the local path, in the first row and the p_2 control point, which is the end of the local path at the second row. First, we calculate each local Bezier curve's total cost. Then, we examine all the possible local Bezier curves, hence the *Hill Climbing*, and we select the optimal local Bezier curve and we add it to the global Bezier path. In the next step we assign the p_2 of the latest local Bezier curve to be the p_0 of the next local Bezier curve and we repeat the same process. When we reach the last row of the grid we assign the p_2 to the goal position. This way we efficiently create a globally "good-enough" Bezier path, by combining locally optimal decisions.

We expect this generator to be the most time efficient of the three generators that we are going to examine. However, we also expect him to be the most naive of them all. This generator is only capable of moving forward, one step at a time, and choosing the best viable local solution. It has absolutely no capability of handling the presence of a lethal obstacle in one of these locally optimal solutions. That is why we expect him to fail even in the presence of some simple lethal obstacles inside the "good-enough" path that we expect him to select. The Hill Climbing generator will just have no way to get around them.

5.3.2 N-best generator

The N-best generator builds on top of the Hill Climbing generator. It operates on a greedy, best-first way and it produces a "good-enough" global solution by taking many locally optimal, or at least "as-optimal-as-possible", decisions. Its implementation basically consists of a couple of additions to the Hill Climbing implementation. This time, at each step of the search, while examining all of our local choices, we keep a collection of the N best choices. We take the best of these choices and we move to the next step of the search.

[1] Complete source code at `https://github.com/yorgosk/ugv_navigation_goals`

If our search fails in this step of the search, because of a lethal obstacle, we return to the previous step and we take the next-best choice. We do this until either we find a choice that enables us to move to a next step where we do not fail because of lethal obstacle, or we run out of choices. If we run out of choices, we consider that our search has failed. In this case, our N-Best generator reports this fail and then terminates.

We expect this generator's time efficiency to be close to the Hill Climbing generator's. However, the N-Best generator is substantially less naive, as it can successfully handle simple lethal obstacles by quickly figuring out a way to go around them. Where we expect this generator to fail is in the presence of complex lethal obstacles, meaning lethal obstacles formations that at some point get into the way of all of the N best local choices. If this happens, the generator will simply have no way to go around them.

5.3.3 Evolutionary Algorithm generator

The Evolutionary Algorithm generator views the problem's field the same way as the previous two generators. It fits a grid between the start and the goal position and considers each cell of the grid to be the node of a graph that spans from the start position to the goal position. So, its task can be reduced to finding the least cost path between these two positions, while avoiding any prohibited areas that it may encounter. An area is prohibited because it is crowded with lethal obstacles.

The Evolutionary Algorithm that our generator uses, operates in the same philosophy as the one presented on the *Chapter 4*. Its termination criteria are our search reaching a designated threshold of examined generation, or our search achieving its object, which is producing an admissible path from start to goal position, or a designated threshold of consecutive stagnated generations being reached during our search. A valid smooth curved path from start to goal is considered as an individual and each path's waypoints are considered as being the individual's chromosomes.

We start our search, which is our evolutionary process, with a specific number of random individuals. We sort those individuals based on their fitness and we enter our generator's main loop. In the beginning of each step of this loop, we select the best-fit individuals for reproduction. We take every possible pair of these individuals and we subject it to crossover, in order to create the individuals of the next generation. We then evaluate the fitness of our new individuals, we take them and their parents, we sort them all based on their fitness and we proceed to check whether they fulfill the termination criteria. If they do not fulfill the termination criteria we proceed to the next generation. If they fulfill these criteria, we select the best individual of the current generation as a solution to our problem and proceed to send him to the move base.

Something that is also worth commenting is the absence of the mutation operation during the reproduction phase of our evolutionary process. Its use was examined, however, experimentally it was proved as being inefficient, because, while it lead to the production of some good paths, those paths were often going against the robot dynamics or were too close to the margin of our problem's field. When our robot ended up following them, it risked being lead to an undesired position because of the natural localization error that

exists, as we mentioned above.

Our Evolutionary Algorithm generator is the most advanced path production technique that we examine in this thesis and we naturally have high expectations on its performance. We do not just expect him to find a “good-enough” path from start to goal under simple conditions, like no obstacles or occasional lethal obstacles, we also expect him to perform commendably in cases where the objects are complex and exist in formations. This will make an Evolutionary Algorithm generator the most highly recommended option to face the incline terrain navigation problem.

5.4 From idea to implementation

My waypoints approach is implemented as a ROS node named *ugv_navigation_goals*. This node can be compiled with the parameters that the user sees to fit his needs better. Before running this node the user should have already activated the robot’s move_base node, as well as any other node that is desired or necessary for his robot’s navigation. Once the ugv_navigation_goals node starts running, it loads the problem’s set-up, it initializes the move_base’s initial position to be the problem’s start position and it proceeds to create a path between the start and the goal positions. When an admissible path has been created it is transmitted to the robot’s move_base.

5.4.1 Sending path plan to move_base

One of the most challenging and time-consuming parts of the implementation that demonstrates the work presented in this thesis, was finding the appropriate way to send the path’s waypoints to the robot’s move_base node. It is obvious that having the right path is completely useless if we can’t make sure that our robot is going to follow it with enough precision while moving. That is why, in this section, we are going to talk with some detail about how this is done.

The most important measure that we take in order to make sure that we have a plan which, in practice, can be executed successfully by our robotic platform is to make sure that, between each triplet of control points that constitute a path’s quadratic Bezier curve, there are enough “minor” waypoints for our robot to follow. These waypoints are generated by the formula that gives us a quadratic Bezier curve [75]:

$$[x, y] \; = \; (1-t)^2 P_0 \; + \; 2(1-t)tP_1 \; + \; t^2 P_2 \; , \;\; t \in [0, 1]$$

t is a parameter ranging from 0 to 1 and controls the position of the produced curve point relative to the position of the curve’s endpoints. In a Bezier curve, the first and the last control points of the curve are its *endpoints*. So, if $t = 0$ then $t \equiv P_0$ and if $t = 1$ then $t \equiv P_2$. We use these “minor” (non-control-points) waypoints to make sure that the robot follows a smooth curved path, while moving from P_0 to P_2 in a local curve of our Bezier path. We also use a simple clean-up function to make sure that the path that we end up with has no waypoints-outliers. Such waypoints may be parts of the curve that make a robot go backwards in the *x’x* axis, so it essentially loses a part of its progress.

Some other minor measures have been taken that have to do with the rate in which our path is transmitted to the robot's move_base. However, as these measures heavily depend on our platform's hardware and its middleware, it is not worth any analysis, because they are essentially part of the ugv_navigation_goals package calibration to the user's specific UGV.

6. SIMULATION RESULTS

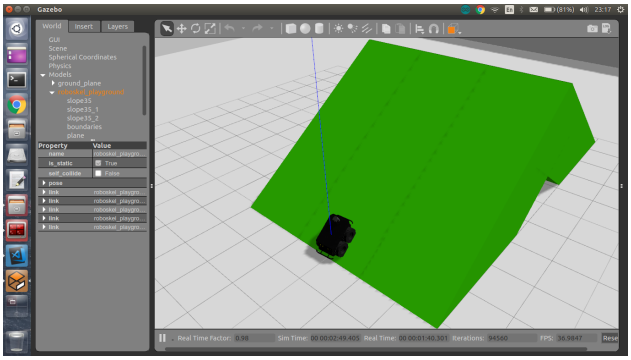
The concepts and implementations described in the previous chapters have been tested in simulation using the Gazebo environment. As a testing platform a laptop with an Intel® Core™ i7-4710MQ CPU @ 2.50GHz × 8 CPU, a GeForce GT 740M/PCIe/SSE2 GPU, 5,7 GiB RAM and 64-bit Ubuntu 16.04 LTS was used.

6.1 Simulation Environments

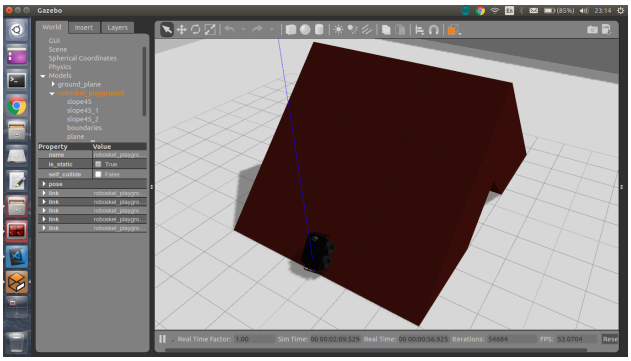
For the purpose of my thesis experimental validation with a UGV with attributes and dynamics like the Clearpath's Husky [57] and the Dr Robot's Jaguar [15], three simulation environments were created [1]. A 35° slope environment, a 45° slope environment and a 43° slope environment, but with 45 meters of straight-line distance between the start and goal positions, meaning a 45 meters long hypotenuse for the slope's triangle. The 35° slope has a substantially reduced (1) friction coefficient and, as such, represents slippery terrain conditions, whereas the 43° and 45° slopes have attributes that in real world can be considered ideal, like a friction coefficient 1. This is like driving in a brand new highway, with brand new wheels, only that this highway has an almost 100% inclination. The 35° slope was chosen because it helps us see in a simulated environment whether our method, in its current version, has the potential to help the UGV in a relatively challenging real-world scenario to go further than it would have gone in case it was climbing "straight up". The 45° slope was determined experimentally as being ideal for the testing of my methods, because it is steep enough for the Husky and Jaguar to not be able to traverse it in a straight line path without tipping over, while it is not too steep so that the UGV won't be able to perform manoeuvres which are still in an experimental stage on it. The 43° slope environment, with 45 meters of straight-line distance between the start and goal positions, was created by having in mind the maximum range of the localization sensors available in the market at the time of writing this thesis, which is about 25 meters. This way it is demonstrated that the smooth curves method is viable even in cases where the robot can't see as far as the goal, but somehow has a precise knowledge of its existence and of the challenges that it is going to face in order to reach it, perhaps by another system or software package.

For each one of these three slopes, three separate scenarios have been created. In the first one, the robot will have to reach the goal without having any lethal obstacle along the way. In this scenario, the *Hill Climbing* path generation approach is expected to be the most efficient one. In the second scenario, the robot will have to reach the goal while having certain non-complex and small in area obstacles along the way. In this scenario, the *N-Best* path generation approach is expected to be the most effective and efficient. Finally, there is a third scenario, perhaps the most realistic one for outdoor incline terrain navigation, where the robot has to overcome certain relatively complex and large in area lethal obstacles along the way. In this scenario we are going to see the usefulness of the

[1] All simulation environments can be found at the address `https://github.com/yorgosk/husky_simulator/tree/indigo-devel/husky_gazebo`



(a) 35° slope set-up



(b) 45° slope set-up

Figure 3: 35° and 45° slopes set-up

more sophisticated *Evolutionary Algorithm* path generation approach.

Before we continue, it has to be noted that the search's *resolution* or *search step* was kept the same for all the simulation scenarios, with it being equal to 1.5 meters. This number derived from experimentation, regarding which resolution was best for our generators to search for a solution in our simulation scenarios, which are characterized by the narrowness of the area in which our robot can move. This is a relatively small resolution if we consider that our simulations' UGV has a length of *0.3 meters*.

For the simulations, the **husky_gazebo** ROS package was used, with the **husky_roboskel_playground_world.launch** launch file. The robot was placed in the exact initial position using the **teleop_twist_keyboard** ROS package and the its move_base was initialized using the **husky_navigation** ROS package, with the launch file **move_base_mapless.launch**. The ugv_navigation_goals package was used by calling the **rulah_navigation_goals** ROS node with the **rulah_navigation_goals.launch** launch file. Here, the "rulah" instead of the "ugv" comes to honor "Rulah", the Roboskel Group's Dr Robot's Jaguar UGV that was used for the real life demonstration that we are going to see in *Chapter 7* and whose replica we are going to see in this chapter's simulation videos.

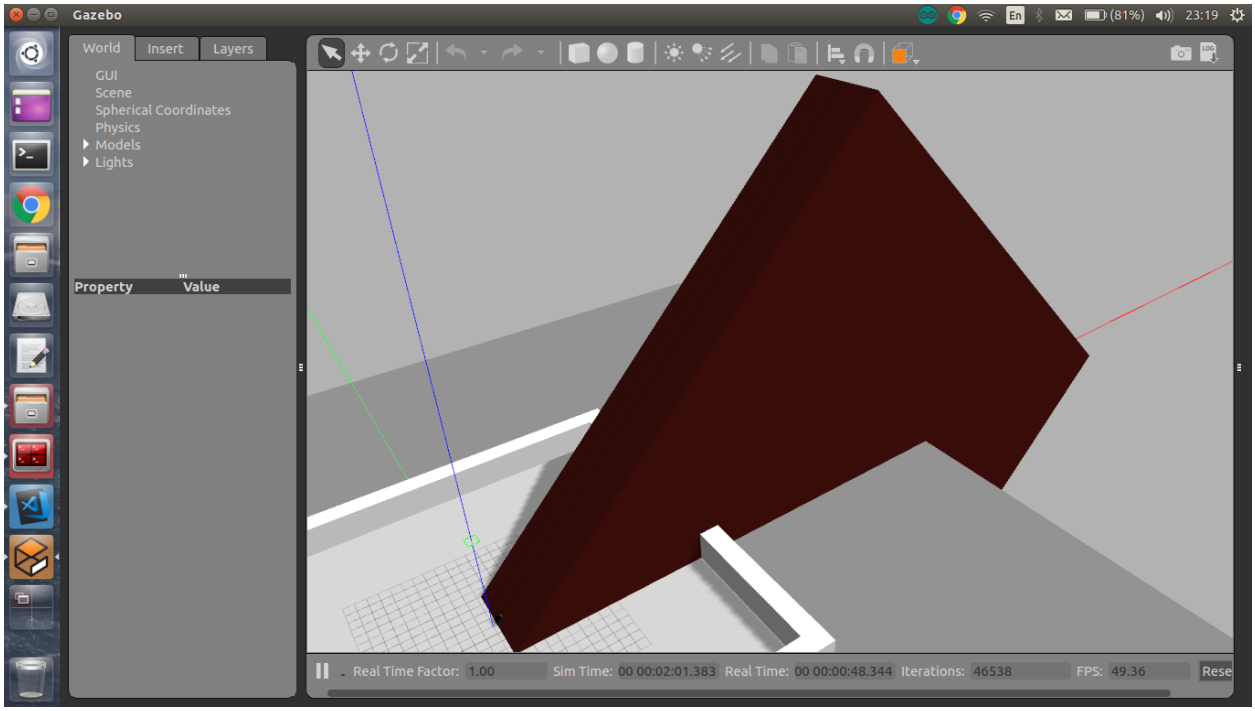


Figure 4: 43° - 45 meters slope set-up

6.2 Simulation Results

In this section we will see the experimental results of the implementations of the ideas that we talked about so far. The results are organized per simulation scenario. For each simulation scenario we see a table of statistics regarding the performance of each one of the three generators that we examined in *Chapter 5*. Then, we interpret these statistics as well as the results that the generators produced for the given simulation scenario and we outline what can be learned by them.

All simulation videos can be found at the Vimeo™ video-sharing website, by following the links that accompany each simulation scenario. For each simulation scenario there is a video of the attempt to traverse the respective slope going “straight up”. Then we see the videos of the attempts to traverse the slope with paths produced by the Hill Climbing, N-Best and Evolutionary Algorithm generators. Because of the highly probabilistic nature of the Evolutionary Algorithm’s path generation process, we will examine the results of six different Evolutionary Algorithm generated paths for each simulation scenario.

For the N-Best generators we have obstacles placed at (1.6, -0.027), (2.6, 0.8), (4.3, 1.6) and, for the 45 meters long slope, at (20.8, 0.8), (29.825, -0.675) and (38.82, 0.82). For the Evolutionary Algorithm generators we have obstacles placed at (1.6, -0.027), (2.6, 0.8), (4.3, 1.6), around (2.1028, 0.312), (4.3025, 1.36875) and (5.218125, 0.829688) and, for the 45 meters long slope, around (20.8, 0.8), (29.825, -0.675) and (38.82, 0.82). All the obstacles and obstacle formations were selected and placed by observing the path that the robot would have followed had there not been any obstacles in the way, like the path produced by the Hill Climbing generator, in order to make the simulation scenarios

as hard as possible.

We will examine the results of various simulations, most of which are available in videos. We will judge each simulation's outcome based on its visited states, path size, path length and path cost. As we saw in *Chapter 5* the path cost derives from the formula:

$$cost = 3.5 * (45.0/terrain\_slope) * \sum norm\_deviation \ + \ 10.0 * \sum pitch \ - \ 10.0 * \sum roll \ + \ (terrain\_slope/10.0) * \sum arc$$

6.2.1 The 35° slope

In this subsection the results on the 35° slope are presented. This slope may not be particularly steep relatively to our generators' capabilities in *"perfect"* terrain, however it represents a different kind of challenge, having a friction coefficient $\equiv 0.7$. We can see the result of the robot's attempt to traverse the 35° slope with a path going straight up in the video `https://vimeo.com/276935141`.

6.2.1.1 Hill-climbing generator

Table 1 presents the statistics of the Hill Climbing generator's execution on the 35° slope simulation environment.

Table 1: Hill Climbing generator on a 35° slope

States	Size	Length	$\sum$ deviation	$\sum$ pitch	$\sum$ roll	$\sum$ arc	Cost
7	11	8.033	3.675	301.656	192.948	1578.184°	6627.271

As we see in the video that can be found at `https://vimeo.com/276935031`, the robot's effort with a Hill Climbing generated path is not successful compared to climbing "straight up". While the robot is turning to follow the curved path's specifications, it loses a substantial part of its progress until that point. As a result, I doesn't climb neither as fast or as successfully as it could have done had it chosen to follow a straight path from start to goal.

6.2.1.2 N-Best generator

Table 2 presents the statistics of the N-Best generator's execution on the 35° slope simulation environment.

As we see in the video at `https://vimeo.com/276935073`, the robot reaches a little bit lower with the N-Best generator's path than it did with the Hill Climbing generator's one.

Table 2: N-Best generator on a 35° slope

States	Size	Length	∑ deviation	∑ pitch	∑ roll	∑ arc	Cost
9	14	8.046	3.978	382.119	247.172	1968.055°	8255.565

However, this is mostly coincidental, because of the first viable route that it found in order to survive the small obstacles on its way.

6.2.1.3 Evolutionary Algorithm generator

Table 3 presents the statistics of the Evolutionary Algorithm generator's execution on the 35° slope simulation environment.

Table 3: Evolutionary Algorithm generator on a 35° slope

Sim.	States	Size	Length	∑ deviation	∑ pitch	∑ roll	∑ arc	Cost
1	26	13	6.655	2.402	325.268	280.294	1841.192°	6904.726
2	26	13	9.95	1.846	346.367	250.161	967.037°	4355.0
3	26	13	6.542	4.48	339.456	258.49	1728.391°	6879.187
4	26	13	7.659	1.384	335.286	262.866	1551.816°	6161.79
5	26	13	13.967	3.292	333.383	257.994	1214.039°	5017.845
6	25	13	6.893	1.495	333.713	266.302	1769.997°	6875.823

From the videos at `https://vimeo.com/276934832`, `https://vimeo.com/276934924` and `https://vimeo.com/276934972` we understand that the success of the robot traversing a slippery incline terrain will depend the amount of time that it spends turning and moving away from a straight line upward path. Generally, too much turning will cause our robot to lose a substantial part of its progress and provide us with a result much worse than what we desire.

6.2.1.4 Takeaways

What we learned from the simulations on the 35° slope is that our path generation process, in its current version can't support the traversal of slippery incline terrain regions. In order to enable it to do so, first of all, the terrain's slippage should become a factor in our cost function. However, what the relatively successful attempt to traverse the 35° slope in a path going "straight up" tells us, is that the success of the robot traversing a slippery incline terrain will heavily depend on its platform's capabilities and, perhaps, on the velocity that its base_local_planner commands for its wheels. For, example a relatively lower speed will reduce the wheel slippage, which will reduce the error of the localization through odometry, which, in turn, will enable the robot to deduce that it reached the goal position with a much higher precision, compared to what it does now. In this direction, an AFFW system, like

the TU Darmstadt's "Adaptive Feedforward Controller" examined in *Chapter 3* is going to prove very useful, if not vital.

6.2.2 The 45° slope

In this subsection the results on the 45° slope are presented. This slope's terrain may be *"perfect"*, like the one of a new highway road, however its 45 degrees of slope represent an almost *100%* inclination. Enabling wheeled UGVs to safely traverse such steep incline terrain regions will be ground-breaking, taking the wheeled UGVs capabilities and their practical applications in a whole new level. We can see the result of the robot's attempt to traverse the 45° slope with a path going straight up in the video `https://vimeo.com/276937190`.

6.2.2.1 Hill-climbing generator

Table 4 presents the statistics of the Hill Climbing generator's execution on the 45° slope simulation environment.

Table 4: Hill Climbing generator on a 45° slope

States	Size	Length	$\sum$ deviation	$\sum$ pitch	$\sum$ roll	$\sum$ arc	Cost
7	14	6.997	3.28	481.316	326.84	2122.545°	11107.689

As we can see in the video at `https://vimeo.com/276937081`, the UGV traverses the 45° slope successfully following the path that our Hill Climbing generator produced. The path is relatively smooth, the robot's stability doesn't appear to be lacking substantially at any point of its course and we observe no wheel slippage.

6.2.2.2 N-Best generator

Table 5 presents the statistics of the N-Best generator's execution on the 45° slope simulation environment.

Table 5: N-Best generator on a 45° slope

States	Size	Length	$\sum$ deviation	$\sum$ pitch	$\sum$ roll	$\sum$ arc	Cost
9	14	7.972	3.978	503.528	285.463	1964.615°	11035.344

As we can see in the video at `https://vimeo.com/276937119`, the UGV traverses the 45° slope successfully and by staying clear of any obstacle, while following the path that our N-Best generator produced. Like in the Hill Climbing generator's case the path is smooth, the robot's stability is in no big danger during its course and we do not observe any wheel slippage. What we observe, however, is that the N-Best generator's path has more visited

states and slightly smaller cost than the Hill Climbing generator's path. The difference in cost is coincidental. Had the obstacles been in different positions, the planner path would have been different and its size, length and cost then could have easily been equal or larger than they are in the Hill Climbing generator's path. The difference in the visited states is because of the search space being expanded each time our planner meets a lethal obstacle, as it has to take a step back a look for an alternative local choice, essentially visiting an entirely new state.

6.2.2.3 Evolutionary Algorithm generator

Table 6 presents the statistics of the Evolutionary Algorithm generator's execution on the 45° slope simulation environment.

Table 6: Evolutionary Algorithm generator on a 45° slope

Sim.	States	Size	Length	$\sum$ deviation	$\sum$ pitch	$\sum$ roll	$\sum$ arc	Cost
1	26	13	6.829	2.054	442.946	308.537	1726.431°	9120.22
2	26	13	7.144	2.104	443.146	308.07	1643.832°	8755.374
3	26	13	8.997	1.702	455.149	293.074	1265.894°	7323.229
4	25	13	8.914	2.357	440.171	309.69	1706.104°	8990.526
5	26	13	9.074	1.265	451.774	295.94	1194.308°	6937.153
6	26	13	7.46	2.742	440.602	310.848	1747.873°	9172.564

As the videos at `https://vimeo.com/276936884`, `https://vimeo.com/276936964` and `https://vimeo.com/276936971` show us, the robot traverses the slope relatively smoothly and successfully, by surviving from all the lethal obstacles, while maintaining a high degree of platform stability almost all the time and presenting zero wheel slippage. We have almost four times the visited states compared to the Hill Climbing generator and and three time the visited states compared to the N-Best generator, however this is expected because the many crossovers that take place between generations are considered as visiting a new state. The path's size is slightly smaller than the size of the Hill Climbing and N-Best generators paths and its cost is considerably improved, which is a great display of the Evolutionary Computation's power when a applied to a complex problem like ours. The path's size can be smaller than the Hill Climbing or N-Best generators or larger, depending on the route that will derive from our evolutionary process.

6.2.2.4 Takeaways

As one could have expected, our Hill Climbing, N-Best and Evolutionary Algorithm generators performed successfully at the 45° slope environment. They all run relatively fast and they produced overally good paths. Still, we have to point out the Evolutionary Algorithm generator's competitive edge when it comes to finding a path with a relatively smaller cost and, especially, smaller arcs between sets of three waypoints, factors that in real life can help us characterize a path as energy efficient or not. Energy efficiency is something of

vital importance in any real world mission. We also see that the Evolutionary Algorithm generator's paths have a much lower total pitch compared to the other generators' paths, which is another great advantage of the evolutionary process, in terms of the path's overall safety.

6.2.3 The 43° slope with 45 meters of distance between start and goal

In this subsection the results on an approximately 43° slope with 45 meters of length are presented. Like in the previous simulation scenario, this slope's terrain is also *"perfect"* and has an almost *100%* inclination. We chose this incline to be a little less than 45° in order to minimize the odometry's localization error because of the front wheels not touching the ground while the robot is taking some "close" turns. We can see the result of the robot's attempt to traverse the 43° - 45m slope with a path going straight up in the video `https://vimeo.com/276939333`.

6.2.3.1 Hill-climbing generator

Table 7 presents the statistics of the Hill Climbing generator's execution on the 43° - 45m slope simulation environment.

Table 7: Hill Climbing generator on a 43° - 45m slope

States	Size	Length	$\sum$ deviation	$\sum$ pitch	$\sum$ roll	$\sum$ arc	Cost
85	88	56.597	19.054	3553.714	652.428	12371.694°	82280.929

As we see in the video at `https://vimeo.com/276939137`, the Hill Climbing generator's path helps our robot traverse relatively smoothly in the beginning, however as the time goes by we see the robot's movement losing a part of its initial smoothness. This is because of the localization error created by improper odometry whenever the UGV's front wheels, momentarily loose contact with the terrain. The same applies to the robot's stability and to the precision with which we approach our goal, because of the very same localization error. Still, we have no signs of wheel slippage which is a very good thing. Overally, the Hill Climbing generator's attempt can be considered successful because of the high degree of difficulty that the 43° - 45m slope simulation scenario represents.

6.2.3.2 N-Best generator

Table 8 presents the statistics of the N-Best generator's execution on the 43° - 45m slope simulation environment.

As the video at `https://vimeo.com/276939182` shows us, in the N-Best generator's path execution, like in the Hill Climbing generator's path execution, we observe the same gradual decline in terms of smoothness, stability of the robot and precision to the goal's approach, because of the localization error explained above. Again, we have no wheel slip-

Table 8: N-Best generator on a 43° - 45m slope

States	Size	Length	∑ deviation	∑ pitch	∑ roll	∑ arc	Cost
84	88	56.965	22.751	3552.11	651.828	12402.857°	82418.432

page and, overally, we can say that the whole attempt was successful. Furthermore, like it was pointed out at the $45°$ slope's simulation scenario result, the path's size, length and cost depend on the lethal obstacles locations. In this simulation scenario, we can comment that the obstacles placement can trim a search space, as their position can exclude a number of options from our search.

6.2.3.3 Evolutionary Algorithm generator

Table 9 presents the statistics of the Evolutionary Algorithm generator's execution on the $43°$ - 45m slope simulation environment.

Table 9: Evolutionary Algorithm generator on a 43° - 45m slope

Sim.	States	Size	Length	∑ deviation	∑ pitch	∑ roll	∑ arc	Cost
1	130	72	58.447	23.838	3133.029	576.519	8965.286°	64203.151
2	130	69	53.086	29.941	3175.885	528.774	8927.611°	64969.512
3	130	70	57.404	27.333	3154.436	543.513	8540.456°	62933.309
4	130	67	52.123	20.412	3127.77	632.192	8576.786°	61910.73
5	130	69	52.063	23.274	3136.49	603.638	8763.505°	63096.838
6	130	69	56.515	21.292	3136.52	596.143	8245.466°	60937.267

As the videos at `https://vimeo.com/276938854` and `https://vimeo.com/276938909` show us, the Evolutionary Algorithm's approach to the recorded simulation scenarios was successful. For the path's execution smoothness, the robot's stability and the goal's approach precision we make the same observations as the ones outlined in the Hill Climbing and N-Best generators' paths commentary. However, because of the relatively big $43°$ - 45m slope's environment, we see that the partially random decisions in terms of waypoints placement that our Evolutionary process does, affect the overall path's quality in a substantial degree. The same can be said for the path's length and cost, but in this case in a much more positive note. The Evolutionary Algorithm generator seems to be much better than both the Hill Climbing and the N-Best generator in terms of overall path optimality. Overally, the Evolutionary Algorithm's approach performed successfully. It still has a lot of room for improvement, but it seems very promising in terms of what we can achieve with a more evolved and mature Evolutionary Algorithm generator.

6.2.3.4 Takeaways

Generally, the smooth curves approach works fast and well in a complex and big simulation environment, like the one of the $43°$ - 45m slope. The extended length of the course helps as see clearly that there is still room for improvement in our method, in terms of precision and overall smoothness. This extended length also helps the Evolutionary Algorithm generator to be distinguished clearly for the optimality of the solutions that it can produce in an average case. Also, like it was pointed out in the $45°$ slope case, the Evolutionary Algorithm generator produces path with much lower total arc between sets of three waypoints and much lower total pitch, meaning that it produces much more energy efficient and safe paths.

Table 10: Hill Climbing (Hill Cl.) generator's sample execution times compared to Evolutionary Algorithm (E.A.) generator's sample execution times under the same conditions

Gen.	Sim.	Obstacles	Slope	Distance	Time (Hill Cl.)	Time (E.A.)
Hill Cl.	**1**	No	$35°$	$5.8m$	$0.622ms$	$2.874ms$
Hill Cl.	**2**	No	$35°$	$5.8m$	$0.678ms$	$2.864ms$
Hill Cl.	**3**	No	$35°$	$5.8m$	$0.589ms$	$2.913ms$
Hill Cl.	**4**	No	$45°$	$5.71m$	$0.602ms$	$2.769ms$
Hill Cl.	**5**	No	$45°$	$5.71m$	$0.801ms$	$2.682ms$
Hill Cl.	**6**	No	$45°$	$5.71m$	$0.797ms$	$2.977ms$
Hill Cl.	**7**	No	$43°$	$44.0m$	$59.543ms$	$9.318ms$
Hill Cl.	**8**	No	$43°$	$44.0m$	$58.579ms$	$9.424ms$
Hill Cl.	**9**	No	$43°$	$44.0m$	$59.239ms$	$9.471ms$

Table 11: N-Best generator's sample execution times compared to Evolutionary Algorithm (E.A.) generator's sample execution times under the same conditions

Gen.	Sim.	Obstacles	Slope	Distance	Time (N-Best)	Time (E.A.)
N-Best	**1**	Simple	$35°$	$5.8m$	$0.866ms$	$2.729ms$
N-Best	**2**	Simple	$35°$	$5.8m$	$0.776ms$	$2.759ms$
N-Best	**3**	Simple	$35°$	$5.8m$	$0.687ms$	$2.619ms$
N-Best	**4**	Simple	$45°$	$5.71m$	$0.612ms$	$2.828ms$
N-Best	**5**	Simple	$45°$	$5.71m$	$0.63ms$	$2.859ms$
N-Best	**6**	Simple	$45°$	$5.71m$	$0.65ms$	$3.075ms$
N-Best	**7**	Simple	$43°$	$44.0m$	$32.268ms$	$10.512ms$
N-Best	**8**	Simple	$43°$	$44.0m$	$27.138ms$	$10.8ms$
N-Best	**9**	Simple	$43°$	$44.0m$	$32.131ms$	$10.877ms$

Table 12: Evolutionary Algorithm generator's sample execution times

Generator	Sim.	Obstacles	Slope	Distance	Time
Evolutionary Algorithm	**1**	Complex	$35°$	$5.8m$	$2.866ms$
Evolutionary Algorithm	**2**	Complex	$35°$	$5.8m$	$2.829ms$
Evolutionary Algorithm	**3**	Complex	$35°$	$5.8m$	$2.868ms$
Evolutionary Algorithm	**4**	Complex	$45°$	$5.71m$	$2.859ms$
Evolutionary Algorithm	**5**	Complex	$45°$	$5.71m$	$2.946ms$
Evolutionary Algorithm	**6**	Complex	$45°$	$5.71m$	$2.966ms$
Evolutionary Algorithm	**7**	Complex	$43°$	$44.0m$	$10.851ms$
Evolutionary Algorithm	**8**	Complex	$43°$	$44.0m$	$10.748ms$
Evolutionary Algorithm	**9**	Complex	$43°$	$44.0m$	$10.681ms$

6.3 Time efficiency

Tables 10, 11 and 12 present the execution times recorded by various sample runs of our three path generators under the variety of conditions that was presented above. More specifically, for each generator we run three times each simulation scenario that was examined above, in which we have a specific distance between the start and the goal position. As we have already discussed, in order for the testing scenarios to be reasonable, we run the Hill Climbing generator considering no lethal obstacles between the start and the goals position, we run the N-Best generator considering simple obstacles the start and the goal position and we run the Evolutionary Algorithm generator considering complex obstacles between the start and the goal position. However, the Evolutionary Algorithm generator can also be run with no obstacles and simple obstacles as well and, as it is the most advanced path generator which we discussed, it is useful to see how it performs under such conditions in comparison to the Hill Climbing and the N-Best generators.

From the tables 10, 11 and 12, it is evident that the execution time of each path generator is relative to the complexity of the situation that it has to deal with. Here, by *complexity* we mean a combination of the straight line distance between the start and the goal and the presence of lethal obstacles in the slope. From the N-Best generator metrics, we can also observe that the presence of obstacles can indeed "trim" the search space and reduce the path generator's execution time. In the Evolutionary Algorithm generator, as expected, its execution time is depended on its partly probabilistic evolutionary process. However, its execution time is relatively stable for similar simulation scenarios. Lastly, the most important thing that we understand from the results presented in the above table is that the Evolutionary Algorithm scales up much better than the Hill Climbing and N-Best algorithms, as it handles much more complex tasks than they ever did in a small fraction of their most complex tasks recorded times. These complex tasks that we are referring to, are of course the $43°$ - 45 meters slope under various conditions of lethal obstacles presence.

What the comparison demonstrated at the tables 10 and 11 indicates, is that in relatively non-complex scenarios it is better to prefer a Hill Climbing or a N-Best generator than the Evolutionary Algorithm generator. That is because they approach the problem more

straight-forwardly than the Evolutionary Algorithm and, as a result, they manage to solve it more efficiently. But, as the complexity of the scenarios grows, even in the absence of complex lethal obstacles, the Evolutionary Algorithm generator becomes increasingly preferable compared to the Hill Climbing and N-Best generators, because, as we already pointed out it scales up much better. This can be attributed to the Evolutionary Algorithm's probabilistic nature, which in large scales is arguably much more efficient than, for example, exhaustive searches.

6.4 A summary

As the simulations presented in this chapter indicate, our smooth curves approach in its current stage may be suitable for navigation in a perfect terrain of extremely steep inclination, however it is not yet suitable for navigation in slippery terrain. For navigation in slippery terrain to be possible, its friction coefficient should become a factor in the generators' inner cost function and it would also be of great help if an AFFW system could run in parallel to counter any momentary wheel slippage that it detects. Also, there is still room for improvement in our method's precision and overall smoothness. We may observe no problem in any short distance use, however as the distance between start and goal grows larger, the need for improvement in these areas becomes increasingly noticeable.

That being said, we have in our hands a good first implementation of an approach that seems promising to one day step out of the lab and be used successfully in the real world. It is also evident that an Evolutionary Algorithm-generated path is the most proper way to face the incline terrain navigation problem, as the paths that can be derived from a well designed evolutionary process will be much safer most of the time and always much more energy efficient. Also, the Evolutionary Algorithm scales up much better than any other algorithm, meaning that as the complexity of the scenarios that it has to face grows, it can be expected to perform much faster than any other algorithm and we can also guess, based on our evidence, that this speedup will increase as the complexity grows even more.

7. REAL LIFE DEMONSTRATION

For the validation of the smooth curves method potential in the real world, a real life demonstration was performed. In this chapter we are going to take a look at the results of this demonstration and discuss what possibly remains for our method to be used in the real world.

7.1 Where the demonstration took place

The demonstration took place in the NCSR "Demokritos" campus, with the assistance of the Roboskel Group. "Rulah", the Dr Robot's Jaguar UGV which we have already mentioned was used. The initialization of its move_base ROS node and the call of the ugv_navigation_goals package ROS node happened like we saw at the simulations of *Chapter 6*. The demonstration's environment consisted of a slope of about 30° of incline with a timeworn asphalt terrain. The robot had to traverse this slope while trying to avoid certain complex lethal obstacles formations that had been placed in its field of knowledge. For visualization purposes when recording, in the coordinates of these lethal obstacles, yellow chairs were placed. For the path's generation, the Evolutionary Algorithm generator which we examined in the previous chapters was used.

7.2 How Evolutionary Algorithm performed

Table 13 presents the statistics of the Evolutionary Algorithm generator's execution on the real life demonstration's environment.

Table 13: Evolutionary Algorithm generator on the real life demonstration's environment

Sim.	States	Size	Length	$\sum$ deviation	$\sum$ pitch	$\sum$ roll	$\sum$ arc	Path's cost
1	42	20	14.277	6.168	661.345	350.269	2487.437°	11844.555
2	40	17	11.708	3.839	647.683	375.049	2061.91°	9960.314
3	40	20	13.908	4.795	637.854	383.357	2583.581°	11609.084
4	40	17	11.701	3.605	636.327	391.145	2061.496°	9683.285
5	42	18	13.814	4.888	667.487	338.272	2424.65°	11800.426
6	42	19	12.837	6.43	665.973	338.458	2449.778°	11878.316

As the video at `https://vimeo.com/276945630` shows us, the robot manages to traverse the incline terrain safely and successfully. However, there is a substantial drop in quality in the generated path's execution, from simulation to the real world. That is mainly because of odometry error. It is clear that the robot suffers from wheel slippage at many moments during the demonstration, slipping either due to the timeworn terrain or due to dried pine-leaves that can be observed in the terrain.

Figure 5: Real life demonstration set-up

7.3 Takeaways

The main takeaway from the real life demonstration that was presented in this chapter is that there is still a substantial amount of work that has to be done for our method to become reliable in the real world. We already suspected this, however, we now know that this work's aspects can be classified as either algorithmic or engineering related. For, example taking into account an estimation of the terrain's slippage while generating our path has to be dealt with at an algorithmic level. Likewise, making sure that we have a precise knowledge of the lethal obstacles formations positions is better to be dealt with at an engineering level, as is it highly affected from the available hardware's capabilities and quality. Making sure that the robot is able to localize itself properly is a slightly more complex task, as it is expected to require a simultaneous algorithmic and engineering approach.

8. CONCLUSIONS AND FUTURE WORK

8.1 A synopsis

In this thesis we discussed terrain traversability problems and, more specifically, the incline terrain traversability problem. An approach which can help a wheeled UGV traverse a terrain region of steep incline was also proposed. This approach evolved around providing the robot with a path consisted of smooth curves. These curves were approximately quadratic Bezier curves and their paths were generated by three different generators. The first generator that we saw was a Hill Climbing generator, from which we expected great performance in a ideal terrain, clear of lethal obstacles. The second generator was a N-Best generator, from which we expected good performance as well as the capacity to deal with some small lethal obstacles. The third generator was an Evolutionary Algorithm generator, from which we expected good performance as well as being able to deal with complex and extensive lethal obstacles.

As the simulations of *Chapter 6* demonstrated our method can successfully help a robot traverse a terrain region of very steep incline, even for relatively long distances compared to the range of the UGV localization sensors available on the market at the time of writing this thesis. Yet, there is still much room for improvement to make our smooth curves method safely applicable to major real world outdoor terrain scenarios. The criteria of the paths generation have to be reevaluated in order for the terrain's slipperiness to become a factor. Also, many improvements can be made in terms of the path's form, so that its smoothness and precision do not gradually decline through the course of the path, due to the accumulation of small momentary localization error.

That being said, the simulations results demonstrate sufficient progress in the incline terrain traversability problem for us to be encouraged to see how our method operates in real world, in its current version. It is also obvious that the most suitable path generation method for real world scenarios is using an evolutionary process, like the one that comes with the Evolutionary Algorithm generator that we presented. That is because, the Evolutionary Algorithm generator has the capacity to produce much more time efficient *(smaller total cost)* and safe *(smaller total pitch)* paths than any other generator. Additionally, it does not require any compromises in terms of performance and scales up really well. The relatively simple real life demonstration that we saw in *Chapter 7* comes to reassure that a mature version of our method can find its place in real world applications. This will largely depend on the success with which we are going to tackle the odometry's error problem for which we talked about, as well, in *Chapter 7*.

8.2 Generalizing our path planning process

The smooth curves method that was proposed in this thesis and is implemented by the ugv_navigation_goals package is still experimental and, in many aspects, in a primal stage, however it is highly promising for the future of autonomous navigation in challenging environments, not only in the ground but also in the air and the water. In the case

of the ground, it is necessary for the method to be evolved so that it can safely navigate the vehicle in a real-world outdoor incline terrain where the slope is not going to be the only challenge, but the roughness, the step and, of course, the slipperiness will also have to be taken into account. Naturally, something like that will most probably never depend solely on the ugv_navigation_goals software package, but it will need assistance from all the other software packages whose nodes may be running in the UGV simultaneously. But there is definitely room for improvement in the ugv_navigation_goals core algorithmic components like the path evaluation function, so that the roughness, step and slipperiness of the terrain will be taken into account. Also, some machine learning components would be useful if they were added, as the node will be able to use the feedback of the move_base to evaluate its path cost assignment accuracy and adapt it on-the-fly, in order to minimize its error and ensure that the locally optimum Bezier curve is always selected. This is going to be very important in case of planning for a very large incline terrain region, like a whole mountain, were we may have a series of predefined navigation goals, perhaps wherever the slope of the mountain changes significantly, and we want to move from one goal to the next by creating a “good-enough” Bezier path.

If we review the incline terrain navigation problem abstractly, we see that we have an autonomous vehicle that has to navigate towards a goal position by moving inside a strictly defined region, while facing challenges like a force negative to its movement (gravity) and prohibited sub-regions (lethal obstacles). It is also clear that one of the best ways to go against such a problem is to adopt a smooth-curve-based moving pattern, no matter whether we talk about manned or unmanned vehicles, no matter whether we are moving in the air, in the ground or in the water. For example, an airplane has to deviate slightly from its course in order to increase the time needed to reach the goal-airport in case it learns, while in the air, that the landing strip of this airport is crowded from other airplanes with higher priority. Also, both in the air and in the sea, a vehicle may be forced to deviate from its course in order to avoid falling into bad weather. In all of these problematic situations, following an alternative smooth-curved path, like a Bezier path, may be one of the most effective and efficient ways for the vehicle to complete its mission or even survive.

ABBREVIATIONS - ACRONYMS

AFFW	Adaptive Feed-forward
OS	Operating System
ROS	Robot Operating System
RPY	Roll Pitch Yaw
UGV	Unmanned Ground Vehicle

APPENDIX A. ROBOT OPERATING SYSTEM (ROS)

The ideas presented in this thesis were developed, tested and verified at the Clearpath's Husky UGV [57] and Dr Robot's Jaguar UGV [15], both running the Robot Operating System. So it was deemed necessary to devote an Appendix of this thesis to presenting ROS, its aspects and its components that affected the design of our Bezier path's approach.

A.1 A few words about ROS

ROS is a *middleware* for robot software development created by the robotics research lab Willow Garage. At 2012, Willow Garage span out Open Source Robotics Foundation, which is in charge of ROS distribution and development up to this date [20]. The fact that ROS is a middleware means that it is actually a collection of frameworks and not an operating system (OS), as its name suggests. However, it provides its user with some of a modern OS's core functionalities, like hardware abstraction, low-level device control, message-passing between processes and package management. A ROS-based process is called a *node* and running a ROS-based system commonly means running a set of nodes. This set of nodes can be represented by a graph, with each graph's vertex *(node)* representing the functional entity of a ROS node and each graph's edge representing a communication channel between two nodes. Two ROS nodes *(processes)* communicate via either a TCP or a UDP protocol, so the problem of designing a well structured ROS software package can be seen as trying to design a system programs' package. This is very important to remember, because, while reactivity and low latency are of vital importance in robot control, ROS is not a real-time OS by nature, meaning that it generally cannot process data as it comes in, without buffer delays. Still, with careful design and many tweaks, it is possible to operate a real-time system in ROS. Officially, the need of support for real-time systems is being addressed in the upcoming ROS 2.0 [31], which will represent a major ROS renovation, rendering it capable of supporting even industrial applications.

A.2 Navigation stack

One of the most important ROS components is the 2D navigation stack. It takes as input information from the various robot localization sensors and outputs safe velocity commands, which the robot should follow in order to reach a predefined goal.

A.2.1 move_base

The move_base implements one of the fundamental actions that a moving robot performs. This is that given a goal in the world the robot should be able to reach it. To achieve this the move_base uses a *global* and a *local* planner. In simple terms, the global planner is in charge of making sure that the robot will attempt to reach the goal that it was assigned in

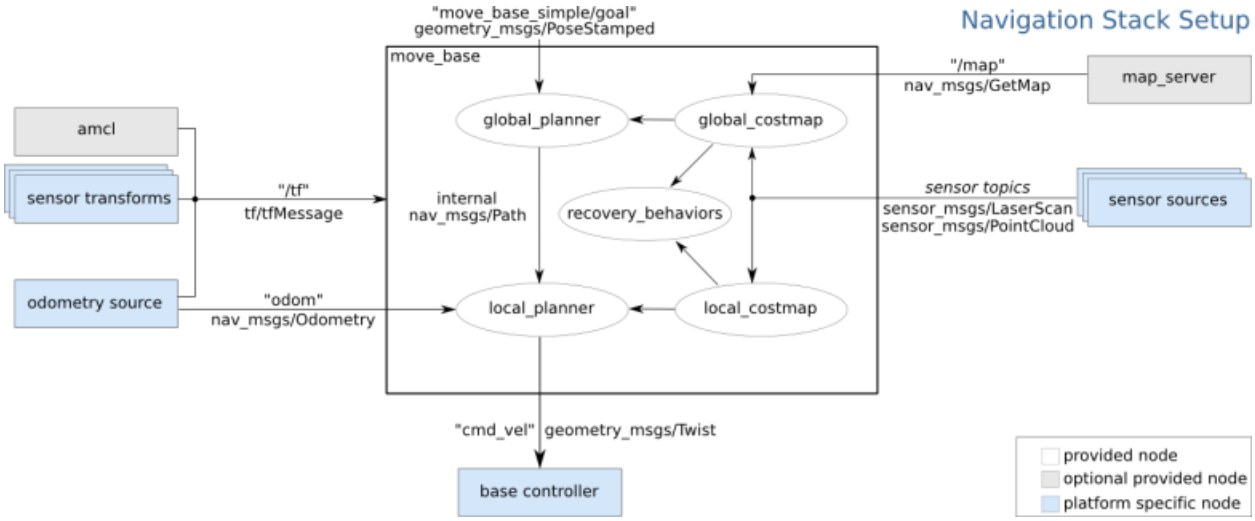


Figure 6: A typical robot's Navigation Stack Setup [42]

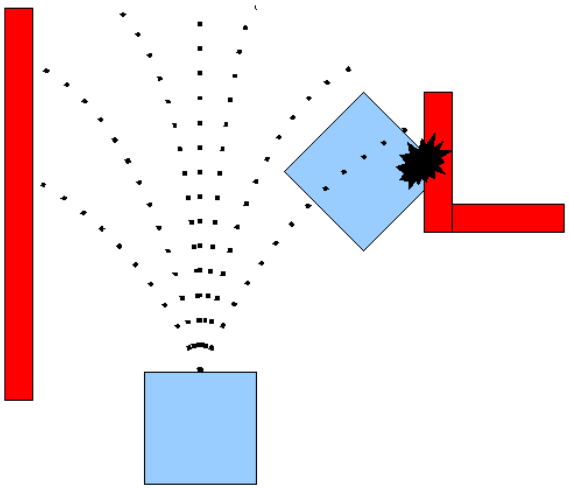


Figure 7: A graphic representation of how ROS base_local_planner works [41]

its world, while the local planner is responsible of producing the correct velocity commands for the robot to navigate its "immediate" environment successfully, by avoiding obstacles and various unwanted locations.

APPENDIX B. ROBOTIC PLATFORM DYNAMICS

Our robotic platform's dynamics play a pivotal role in the path planning process. Our cost function, as presented in the previous chapters takes into account factors like the estimated UGV's roll and pitch, in order to assign a cost to a waypoint. In this chapter, we are going to see a brief introduction in the robotic *vehicle* dynamics and we will discuss the role they can play in an offline path planning process, like the one we use to solve our incline terrain traversability problem.

B.1 Vehicle movement principal axes

The principal axes of which we can use to describe a vehicle's movement are *roll (longitudinal axis), pitch (transverse axis) and yaw (vertical axis)*. By having an accurate knowledge of a vehicle's position relative to these three axes, we can accurately position a vehicle in a free space. These axes are commonly used to describe an aircraft's orientation during a flight, but in principle there is absolutely no difference in using them to describe a ground vehicle's orientation while moving in an uneven terrain. In other terms, we can say that roll represents the rotation of a rigid body around the *x* axis, pitch represents its rotation around the *y* axis and yaw around the *z* axis.

In this thesis, we see vehicle's roll and pitch playing an important role in a waypoint's and, in extension, a path's evaluation in our path planning process. When a vehicle climbs an incline terrain region "straight up", its pitch angle is relatively greater than while climbing it under a certain angle, like it does most of the time when it follows a smooth curved path. In the same logic, the vehicle's roll angle is relatively greater most of the time when climbing an incline terrain region in a smooth curved path, than what it is when climbing "straight up". So, we can see our path planning problem as a *constraint programming* problem, where the optimal solution consists of maintaining a "healthy" balance between the robot's roll and pitch and making sure that it doesn't drift from the straight line course between start and goal more than what it is necessary while climbing an incline terrain

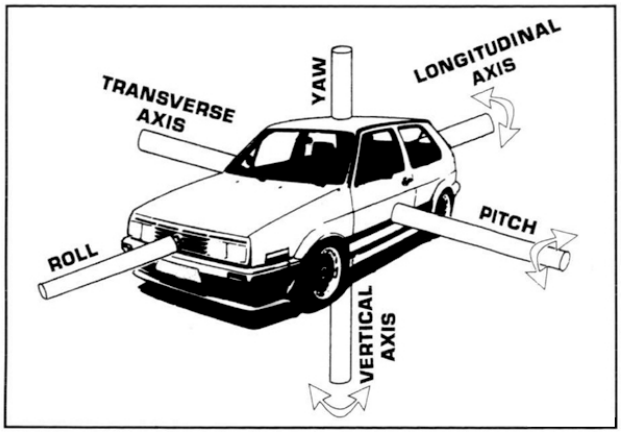


Figure 8: Roll, pitch and yaw motions defined along the principal axes of movement [13]

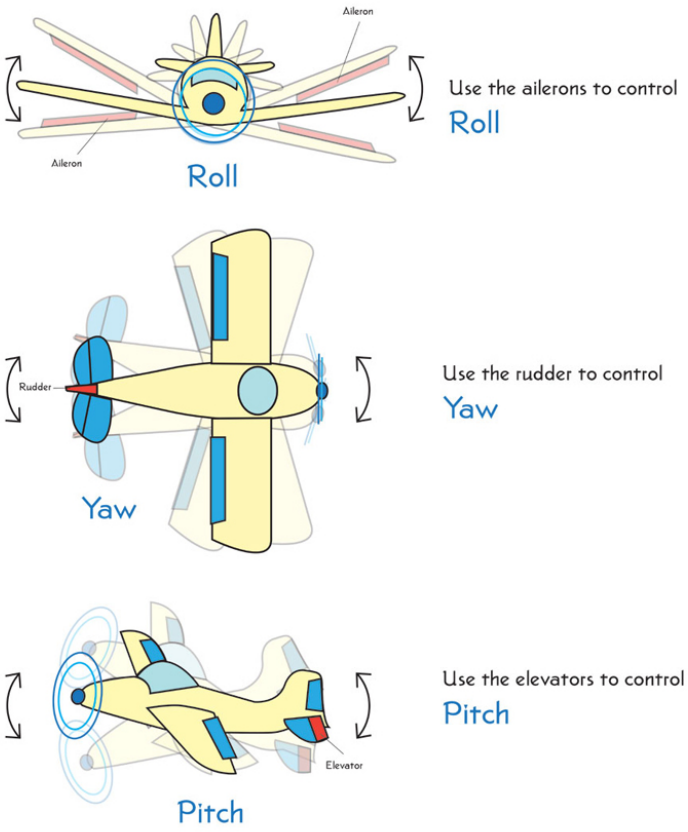


Figure 9: Roll, pitch and yaw at an aircraft's body [63]

region.

B.1.1 How we estimate the roll, pitch, yaw (RPY)

We estimate Roll, Pitch, Yaw (RPY) using formulas taken by the great work of Fariz Ali and Atsuo Kawamura at *"Diagonal Walking Strategy on Inclined Floor with Orientation Based Inverse Kinematics for Biped Robot"* Ali and Kawamura [1]. Their work, actually, studies the concept of a biped robot walking on an incline floor and their mathematic analysis focuses on an abstract view of the robot's feet. However, since the geometry of the robot's feet abstraction resembles closely an abstract view of our UGV's body, which is a *parallelepiped*, the same formulas were used for our analysis as well. If the reader wants to know more about these formulas and how they are derived, it is heavily recommended that he reads the original Ali and Kawamura's work.

REFERENCES

[1] F. Ali and A. Kawamura. Diagonal walking strategy on inclined floor with orientation based inverse kinematics for biped robot. *International Journal of Mechanical and Mechanics Engineering*, 11(4):38–44, 2011.

[2] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *The International Journal of Robotics Research*, 32(1):19–34, 2013.

[3] A. Angelova, L. Matthies, D. Helmick, G. Sibley, and P. Perona. Learning to predict slip for ground robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3324–3331. IEEE, 2006.

[4] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Fast terrain classification using variable-length representation for autonomous navigation. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[5] E. Z. ASL. Traversability estimation's Github repository. `https://github.com/ethz-asl/traversability_estimation`, 2017.

[6] T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on robotics and automation*, 14(6):926–939, 1998.

[7] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for demo iii. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 326–331. IEEE, 2000.

[8] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner. Three decades of driver assistance systems: Review and future perspectives. *IEEE Intelligent Transportation Systems Magazine*, 6(4):6–22, 2014.

[9] F. L. G. Bermudez, R. C. Julian, D. W. Haldane, P. Abbeel, and R. S. Fearing. Performance analysis and terrain classification for a legged robot over rough terrain. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 513–519. IEEE, 2012.

[10] Boston Dynamics. Official Boston Dynamics website, the "about" section. `https://www.bostondynamics.com/about`, 2017.

[11] M. Castelnovi, R. Arkin, and T. R. Collins. Reactive speed control system based on terrain roughness detection. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 891–896. IEEE, 2005.

[12] Y.-L. Chow and R. Schwartz. The n-best algorithm: An efficient procedure for finding top n sentence hypotheses. In *Proceedings of the workshop on Speech and Natural Language*, pages 199–202. Association for Computational Linguistics, 1989.

[13] R. de Oliveira Santos. Understeer and oversteer balance: The mechanics behind it, and 3 ways in which it is affected by suspension. `http://racingcardynamics.com/understeer-and-oversteer/`, 2015. Last accessed June 2018.

[14] C. S. Dima, N. Vandapel, and M. Hebert. Classifier fusion for outdoor obstacle detection. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 665–671. IEEE, 2004.

[15] DrRobot. Dr Robots's Jaguar UGV website. `http://jaguar.drrobot.com/specification_4x4w.asp`, 2018.

[16] N. Dulay. Interaction to genetic algorithms, department of computing, Imperial College London. `http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.`

html, 2018. Last accessed June 2018.
[17] ETH Zurich ASL. Elevation mapping's Github repository. https://github.com/ethz-asl/elevation_mapping, 2017.
[18] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart. Robot-centric elevation mapping with uncertainty estimates. In *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.
[19] T. Fitzpatrick. Opportunity on verge of new discovery. "https://source.wustl.edu/2011/09/opportunity-on-verge-of-new-discovery/", 2011. Last accessed June 2018.
[20] W. Garage. Open source robotics foundation announcement by willow garage. http://www.willowgarage.com/blog/2012/04/16/open-source-robotics-foundation, 2012. Last accessed June 2018.
[21] D. B. Gennery. Traversability analysis and path planning for a planetary rover. *Autonomous Robots*, 6(2):131–146, 1999.
[22] A. Good. Opportunity hunkers down during dust storm. https://www.nasa.gov/feature/jpl/opportunity-hunkers-down-during-dust-storm, 2018. Last accessed June 2018.
[23] I. Halatci, C. A. Brooks, and K. Iagnemma. Terrain classification and classifier fusion for planetary exploration rovers. In *Aerospace Conference, 2007 IEEE*, pages 1–11. IEEE, 2007.
[24] F. Hauer, A. Kundu, J. M. Rehg, and P. Tsiotras. Multi-scale perception and path planning on probabilistic obstacle maps. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4210–4215. IEEE, 2015.
[25] N. Heckman, J.-F. Lalonde, N. Vandapel, and M. Hebert. Potential negative obstacle detection by occlusion labeling. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2168–2173. IEEE, 2007.
[26] R. Hoffman and E. Krotkov. Terrain roughness measurement from elevation maps. In *Mobile Robots IV*, volume 1195, pages 104–115. International Society for Optics and Photonics, 1990.
[27] M. Hoffmann, K. Štěpánová, and M. Reinstein. The effect of motor action and different sensory modalities on terrain classification in a quadruped robot running with multiple gaits. *Robotics and Autonomous Systems*, 62(12):1790–1798, 2014.
[28] A. Howard, H. Seraji, and E. Tunstel. A rule-based fuzzy traversability index for mobile robot navigation. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 3067–3071. IEEE, 2001.
[29] A. Howard, M. Turmon, L. Matthies, B. Tang, A. Angelova, and E. Mjolsness. Towards learned traversability for robot navigation: From underfoot to the far field. *Journal of Field Robotics*, 23(11-12):1005–1017, 2006.
[30] iRobot. Official iRobot website, the "About iRobot" section. http://www.irobot.com/About-iRobot/Company-Information.aspx, 2017.
[31] K. Jackie. Proposal for implementation of real-time systems in ROS 2.
[32] R. Jonschkowski and O. Brock. State representation learning in robotics: Using prior knowledge about physical interaction. In *Robotics: Science and Systems*, 2014.
[33] R. Jonschkowski and O. Brock. Learning state representations with robotic priors. 2015.
[34] J.-Y. Jun, J.-P. Saut, and F. Benamar. Pose estimation-based path planning for a tracked mobile robot traversing uneven terrains. *Robotics and Autonomous Systems*, 75:325–339, 2016.
[35] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.
[36] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman,

R. Mandelbaum, T. Pilarski, et al. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483, 2006.

[37] A. Kelly, N. Chan, H. Herman, D. Huber, R. Meyers, P. Rander, R. Warner, J. Ziglar, and E. Capstick. Real-time photorealistic virtualized reality interface for remote mobile robot control. *The International Journal of Robotics Research*, 30(3):384–404, 2011.

[38] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 518–525. IEEE, 2006.

[39] I. S. Kweon and T. Kanade. High resolution terrain map from multiple sensor data. In *Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on*, pages 127–134. IEEE, 1990.

[40] D. Langer, J. Rosenblatt, and M. Hebert. A behavior-based system for off-road navigation. *IEEE Transactions on Robotics and Automation*, 10(6):776–783, 1994.

[41] D. V. Lu. ROS base local planner. `http://wiki.ros.org/base_local_planner`, 2018. Last accessed June 2018.

[42] D. V. Lu. ROS navigation stack setup. `http://wiki.ros.org/move_base`, 2018. Last accessed June 2018.

[43] I. Lunden. Softbank is buying robotics firms Boston Dynamics and Schaft from Alphabet. `https://techcrunch.com/2017/06/08/softbank-is-buying-robotics-firm-boston-dynamics-and-schaft-from-alphabet/?guccounter=1`, 2017. Last accessed June 2018.

[44] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous robots*, 18(1):81–102, 2005.

[45] F. Martín, R. Triebel, L. Moreno, and R. Siegwart. Two different tools for three-dimensional mapping: De-based scan matching and feature-based loop detection. *Robotica*, 32(1):19–41, 2014.

[46] A. Milella, G. Reina, and J. Underwood. A self-learning framework for statistical ground classification using radar and monocular vision. *Journal of Field Robotics*, 32(1):20–41, 2015.

[47] NASA Jet Propulsion Laboratory. In-situ exploration and sample return: Autonomous planetary mobility. `https://mars.nasa.gov/mer/technology/is_autonomous_mobility.html`, 2018. Last accessed June 2018.

[48] NASA Mars Exploration Program Missions website. Official spirit and opportunity mars exploration rovers mission summarization page. `https://mars.nasa.gov/programmissions/missions/present/2003/`, 2017.

[49] S. Nissimov, J. Goldberger, and V. Alchanatis. Obstacle detection in a greenhouse environment using the kinect sensor. *Computers and Electronics in Agriculture*, 113:104–115, 2015.

[50] M. Oliveira, V. Santos, and A. D. Sappa. Multimodal inverse perspective mapping. *Information Fusion*, 24:108–121, 2015.

[51] N. Ommer. A framework for adaptive feedforward motor-control for unmanned ground vehicles. 2016.

[52] N. Ommer, A. Stumpf, and O. von Stryk. Real-time online adaptive feedforward velocity control for unmanned ground vehicles. 2017.

[53] D. K. Pai and L.-M. Reissell. Multiresolution rough terrain motion planning. *IEEE Transactions on robotics and automation*, 14(1):19–33, 1998.

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

[55] T. Peynot, S.-T. Lui, R. McAllister, R. Fitch, and S. Sukkarieh. Learned stochastic mobility prediction for planning with control uncertainty on unstructured terrain. *Journal of Field Robotics*, 31(6):969–995, 2014.

[56] M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[57] C. Robotics. Clearpath's Husky UGV website. `https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/`, 2018.

[58] À. Santamaria-Navarro, E. H. Teniente, M. Morta, and J. Andrade-Cetto. Terrain classification in complex three-dimensional outdoor environments. *Journal of Field Robotics*, 32(1):42–60, 2015.

[59] M. Saska, V. Vonásek, T. Krajník, and L. Přeučil. Coordination and navigation of heterogeneous mav–ugv formations localized by a 'hawk-eye'-like approach under a model predictive control scheme. *The International Journal of Robotics Research*, 33(10):1393–1412, 2014.

[60] M. Saska, V. Spurnỳ, and V. Vonásek. Predictive control and stabilization of nonholonomic formations with integrated spline-path planning. *Robotics and Autonomous Systems*, 75: 379–397, 2016.

[61] M. Shneier, T. Chang, T. Hong, W. Shackleford, R. Bostelman, and J. S. Albus. Learning traversability models for autonomous mobile vehicles. *Autonomous Robots*, 24(1):69–86, 2008.

[62] D. Silver, J. A. Bagnell, and A. Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research*, 29(12): 1565–1592, 2010.

[63] Smithsonian National Air and Space Museum. Roll, pitch, and yaw. `http://howthingsfly.si.edu/flight-dynamics/roll-pitch-and-yaw`, 2018. Last accessed June 2018.

[64] Stanford Racing Team. The new official Stanford Racing Team's website, the "Stanley" section. `https://cs.stanford.edu/group/roadrunner/stanley.html`, 2017.

[65] Stanford Racing Team. The old official Stanford Racing Team's website, the "Technology" section. `http://cs.stanford.edu/group/roadrunner//old/technology.html`, 2017.

[66] A. Stelzer, E. Mair, and M. Suppa. Trail-map: A scalable landmark data structure for biologically inspired range-free navigation. In *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, pages 2138–2145. IEEE, 2014.

[67] P. N. Stuart Russell. Artificial intelligence: A modern approach, third edition. pages 122–125, 2010.

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

[69] K. Sullivan, W. Lawson, and D. Sofge. Fusing laser reflectance and image data for terrain classification for small autonomous robots. In *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, pages 1656–1661. IEEE, 2014.

[70] M. Tarokh. Hybrid intelligent path planning for articulated rovers in rough terrain. *Fuzzy Sets and Systems*, 159(21):2927–2937, 2008.

[71] S. R. Team. Stanford racing team's entry in the 2005 darpa grand challenge. *Published on DARPA Grand Challenge*, 2005.

[72] S. Thrun, M. Montemerlo, and A. Aron. Probabilistic terrain analysis for high-speed desert driving. In *Robotics: Science and Systems*, pages 16–19, 2006.

[73] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[74] TU Darmstadt. Adaptive feedforward controller's Github repository. `https://github.com/`

tu-darmstadt-ros-pkg/affw_control, 2017.

[75] H. Tulleken. The website of a Bezier curves tutorial writen for game developers. http://devmag.org.za/2011/04/05/bzier-curves-a-tutorial/, 2018.

[76] N. Vandapel, D. F. Huber, A. Kapuria, and M. Hebert. Natural terrain classification using 3-d ladar data. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 5117–5122. IEEE, 2004.

[77] P. A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 2016 International Conference on*, pages 261–265. IEEE, 2016.

[78] K. Walas. Terrain classification and negotiation with a walking robot. *Journal of Intelligent & Robotic Systems*, 78(3-4):401, 2015.

[79] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter. Navigation planning for legged robots in challenging terrain. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016), Daejeon, South Korea, October 2016*, Oct. 2016.