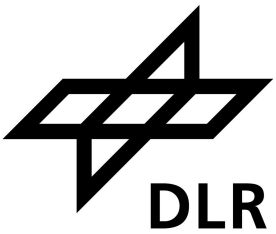


Technische Hochschule Ingolstadt

Faculty of Electrical Engineering and Computer Science

AI Engineering of Autonomous Systems (M.Eng)

Master's Thesis

Motion Planning for Minimizing Wheel Wear in Planetary Rovers on Rough Terrain

Author	**Karem Magdy Ramadan Mohamed**
First Examiner	Prof. Dr.-Ing. Alen Turnwald
Second Examiner	Prof. Dr.-Ing. Ali Kanso
Advisor	Moritz Kuhne, M.Sc.
Institute	Robotics and Mechatronics Center, DLR

Erklärung

Ich erkläre hiermit, dass ich die Arbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum	Karem Mohamed

Acknowledgments

As my fingers are hitting the keyboard to conclude this thesis, I find myself experiencing a wierd feeling mixed with relief, happiness, and sadness. Relief that this journey is finally coming to an end, happiness for having accomplished this milestone, and sadness that this chapter of my life is closing.

First and foremost, I would like to express my gratitude to my supervisor at DLR, Moritz Kuhne, for his continuous support, guidance, and encouragement throughout this work. He has been very patient with me and provided valuable insights that greatly contributed to the quality of this thesis. I am also thankful to my examiner at THI, Prof. Alen Turnwald, for accepting to supervise my thesis and for his constructive feedback. Also for Prof. Ali Kanso for accepting to be the second examiner of this thesis.

Over the past 9 months, I was fortunate to meet amazing coleagues at DLR who became close friends and made this chapter of my life unforgettable. To name a few, Bocchio, Laura, Loneli, Fabsi, Basi, and many more. I am thankful to all of you for the unforgettable memories that shall remain in my heart forever.

Lastly, I would like to thank my family, especially my mom, for their unconditional love and support throughout my life. I love you!

Karem Mohamed
Munich, Germany
February 2026

Abstract

Planetary rovers operating on abrasive terrain experience progressive wheel degradation, with steering maneuvers on rough surfaces identified as a significant contributing factor. While existing motion planners optimize for geometric feasibility, collision avoidance, and energy efficiency, they typically treat all terrain as mechanically equivalent and do not account for the physical consequences of steering maneuvers on wheel lifespan. This thesis investigates motion planning strategies that reduce steering-induced wheel wear by incorporating terrain-aware cost formulations into the planning process.

A curvature-based cost model is developed that penalizes high-curvature maneuvers in proportion to local terrain roughness. Path curvature and its rate of change serve as computationally efficient proxies for steering effort, capturing the kinematic relationship between rover motion and wheel-level steering demands. This formulation shifts wear mitigation from the execution layer, where operational strategies such as modified driving primitives have previously been applied, to the deliberative planning layer.

The terrain-aware cost is integrated into two planning frameworks: a sampling-based kinodynamic RRT* planner with nonlinear trajectory optimization, and a search-based lattice A* planner extended from the Nav2 SMAC framework. For the sampling-based planner, a hybrid rejection–corridor sampling strategy is developed that improves steering feasibility and accelerates convergence; this is followed by a smoothing stage that incorporates terrain-aware curvature penalties. For the lattice-based planner, terrain-dependent edge costs are introduced that penalize high-curvature motion primitives on rough terrain.

Both planners are evaluated in simulation using the Lightweight Rover Unit (LRU) platform across hand-crafted navigation scenarios and 100 randomly generated environments. Results demonstrate that terrain-aware costs generally reduce wheel-level curvature and cumulative steering effort in rough terrain regions, though the mechanism differs by planner. The lattice-based planner achieves consistent improvements by selecting alternative motion-primitive sequences that defer turning until leaving rough terrain, at the cost of increased path length. The sampling-based planner refines trajectory geometry within a fixed route topology, achieving curvature reductions with minimal path extension but exhibiting higher variability across scenarios.

This work contributes a practical framework for embedding wear-conscious objectives into rover motion planning, demonstrating that terrain-aware steering penalties can meaningfully influence trajectory generation without requiring high-fidelity terramechanics models.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Background and Motivation

Planetary rovers serve as mobile scientific platforms that must operate reliably in harsh and unpredictable extraterrestrial environments. Their mobility systems are exposed to continuous mechanical stress from abrasive soils and sharp embedded rocks. The Mars Science Laboratory rover, *Curiosity*, provides one of the most well-documented examples of these challenges. Within its first years of operation, the rover's aluminum wheels experienced punctures, cracks, and tear propagation severe enough to cause changes in both mechanical design and day-to-day driving strategies [1, 2, 3]. These studies identified multiple contributing factors, including localized hazards from angular ventifacts, repeated high-resistance steering, and terrain-induced concentration of mechanical loads.

Although more recent missions, such as *Perseverance*, employ redesigned and mechanically reinforced wheels [4], hardware improvements alone cannot fully mitigate the cumulative effects of long-duration surface exploration. Future mission concepts target geologically rich regions such as crater rims, talus slopes, or sedimentary outcrops where both terrain roughness and mobility demands are elevated. As a result, rover mobility increasingly depends not only on mechanical robustness but also on operational strategies that reduce exposure to wear-inducing maneuvers, particularly steering actions executed on abrasive terrain.

Motion planning is a key component of these strategies. Conventional onboard planners optimize for geometric feasibility, collision avoidance, path length, and energy efficiency [5, 6, 7]. However, as exploration missions target areas with increasingly diverse terrain types it becomes increasingly important for planners to account for terrain-dependent operational risks. This motivates research into planning methods that discourage steering behaviors known to elevate wheel wear when executed on rough or abrasive surfaces.

1.2. Problem Statement

While existing rover motion planners successfully avoid geometric obstacles and respect kinematic constraints, they typically treat all terrain as mechanically equivalent and do not account for the physical consequences of steering maneuvers on wheel lifespan. Operational experience from the *Curiosity* mission has shown that steering under load—such as turning in place or executing tight-radius maneuvers on sharp or high-friction terrain—significantly accelerates wheel degradation [1].

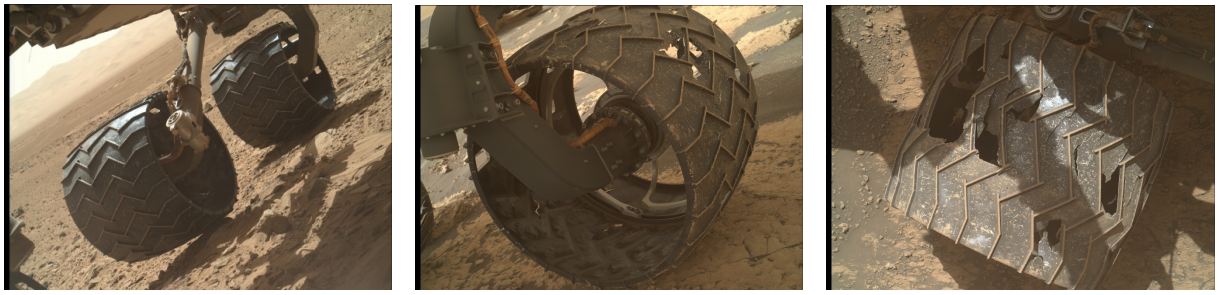


Figure 1.1.: Progressive deterioration of the *Curiosity* rover's wheels from Sol 490, Sol 1681, and Sol 3195, as captured by the Mars Hand Lens Imager (MAHLI). The sequence illustrates the cumulative effects of repeated high-stress steering on sharp and abrasive terrain.

Figure 1.1 illustrates the progressive deterioration of *Curiosity*'s wheels across several mission phases, highlighting how repeated high-stress interactions with sharp terrain features lead to punctures, torn tread, and material loss. These observations highlight that wheel wear is not merely a long-term maintenance issue but an operational constraint influenced directly by how paths are generated and executed.

This thesis therefore addresses the incorporation of a practical and computationally efficient proxy for steering-related wheel wear into the motion-planning pipeline. High-fidelity terramechanics and soil–wheel interaction models, though mechanically accurate, are computationally expensive for onboard real-time planning. Instead, this work investigates geometric indicators that can approximate when steering maneuvers are likely to impose elevated mechanical stress. These indicators offer an operationally meaningful, real-time–compatible representation of conditions under which wear may accumulate. The objective is to produce trajectories that remain feasible, efficient, and sensitive to terrain-induced steering difficulty, thereby reducing exposure to maneuvers known to accelerate mechanical degradation.

1.3. Scope of the Thesis

This thesis focuses exclusively on motion-planning strategies for mitigating steering-induced wheel wear, without attempting to model the physical degradation process or the complex terramechanics that govern soil–wheel interactions [8]. The proposed methods rely on indirect yet practical indicators of steering hazards rather than mechanically accurate stress estimates.

The planning algorithms developed in this thesis are evaluated in simulation using the Lightweight Rover Unit (LRU) from the Robotics and Mechatronics Center at the German Aerospace Center (DLR), shown in Figure 1.2. The LRU is a terrestrial prototype developed for future planetary exploration missions [9]. It features four independently steerable and driven wheels, providing high maneuverability over rough terrain. An active body control system, consisting of two bogie actuators for the front and rear wheel pairs, ensures improved weight distribution and traction while acting as a passive suspension in combination with flexible tire spokes. The wheels are rigid

Figure 1.2.: Lightweight Rover Unit (LRU) from the German Aerospace Center (DLR).

hollow cylinders with circumferentially mounted plates that function as shovels for enhanced traction on loose soil.

This work assumes that terrain information is available as a binary roughness map derived from Digital Elevation Models (DEMs). While such maps are crucial for rover autonomy [10], their generation from sensor data lies outside the scope of this thesis.

The planning algorithms are evaluated within a local horizon of approximately 4 meters, corresponding to the effective perception range of the LRU's stereo-vision system. Beyond this distance, terrain and obstacle information are unavailable.

Additionally, both planners assume quasi-static motion, meaning that inertial and dynamic effects—such as momentum and wheel-ground interaction forces that become significant at higher speeds—are negligible. This assumption is reasonable for slow-moving planetary rovers, where velocities and accelerations are small and wheel-terrain interactions can be approximated as quasi-static. Consequently, planning focuses on kinematic feasibility, obstacle clearance, and steering effort, rather than dynamic stability or high-speed dynamics.

By isolating the motion-planning component and using the LRU as a representative platform, this thesis evaluates how terrain-aware steering penalties influence trajectory generation independent of perception, localization, or hardware modifications.

1.4. Research Objectives

The objective of this thesis is to reduce steering-induced mechanical stress on the rover wheels by embedding terrain awareness into the motion planning cost formulation. Motivated by evidence that tight turns and high-resistance steering accelerate wheel degradation, this work develops planning methods that discourage such maneuvers on rough terrain.

More specifically, the thesis pursues the following objectives:

1. **Formulate a terrain-aware steering penalty** based on geometric quantities that can be computed efficiently during planning, and that assigns higher cost to steering maneuvers

on rough terrain.

2. **Integrate the same cost model into two complementary planning paradigms**:
   - a sampling-based kinodynamic RRT* planner with a nonlinear program (NLP) steering function and a post-processing optimization-based trajectory smoothing stage, and
   - a search-based lattice A* planner implemented within the Nav2 SMAC planning framework [11].

   This dual integration enables a direct comparison of how an optimization-based planner and a search-based planner respond to the same terrain-aware cost formulation.
3. **Evaluate both planners in simulation** in terms of planning success, runtime behavior, and wheel-level steering metrics within rough terrain, quantifying the trade-offs between path efficiency and reduced steering effort.

1.5. Contributions

The main contributions of this thesis are:

- A curvature-based, terrain-weighted steering penalty that serves as a practical proxy for steering-related wheel stress and can be applied consistently across continuous and lattice-based planners.
- A kinodynamic RRT* planning pipeline with NLP-based steering, hybrid rejection-corridor sampling, and a terrain-aware smoothing stage that reduces curvature-intensive maneuvers on rough terrain.
- An extension of the Nav2 SMAC lattice planner with terrain-dependent curvature penalties in the edge cost function, enabling terrain-aware trajectory generation on rough terrain.
- A comparative evaluation of each planner against its baseline variant, as well as a cross-comparison of the terrain-aware formulations, using wheel-level curvature and steering metrics together with runtime statistics to assess performance, robustness, and the path-length trade-offs associated with wear reduction.

1.6. Structure of the Thesis

The remainder of this thesis is organized as follows:

- **Chapter 2** reviews related work on rover mobility, terrain assessment, and motion-planning approaches relevant to planetary exploration.
- **Chapter 3** introduces the rover kinematic model, terrain and obstacle representations, the wheel-wear proxy, and theoretical background on the motion planning formulations employed in this work.

- **Chapter 4** presents the rejection-corridor sampling method and the two planning frameworks: kinodynamic RRT* with a trajectory optimization smoothing stage, and lattice-based A*.
- **Chapter 5** describes the experimental setup and provides a comparative evaluation of the planners under their baseline and terrain-aware configurations.
- **Chapter 6** summarizes the main findings and discusses limitations and directions for future research.

2. Literature Review

This chapter reviews motion planning methodologies for planetary rovers navigating rough terrain, organized into four paradigms: search-based, sampling-based, optimization-based, and learning-based methods. The review examines core algorithms, terrain integration mechanisms, and inherent limitations, providing context for this thesis's contribution to minimizing steering-induced wheel wear.

2.1. Search-Based Planning Methods

Search-based planning algorithms represent environments as discretized graphs or grids, using heuristic-driven search to identify optimal or near-optimal paths. These methods have been deployed on multiple planetary missions, demonstrating reliability and computational tractability for onboard systems.

Field D* served as the foundation for global path planning aboard the Mars Exploration Rovers (MER) [5]. Unlike standard grid-based planners constraining motion to fixed headings, Field D* interpolates costs across grid cells, producing smoother paths. Terrain information is incorporated through a local goodness map constructed from stereo vision, encoding terrain favorability using geometric and textural cues. Regions judged safer receive lower traversal costs, guiding the planner toward benign terrain while avoiding hazards.

Despite operational success, Field D* remains fundamentally a geometric planner. Its goodness map captures coarse traversability but does not encode kinematic feasibility or steering demands. Consequently, the algorithm may generate collision-free, geometrically smooth trajectories that still require tight-radius turns or steering under load—maneuvers known to increase wheel stress.

Graph-based search has also been applied to energy-efficient rover planning. Dijkstra's algorithm generates global paths guided by power-consumption models learned from rover–terrain simulations [12]. Terrain roughness, rover attitude, and predicted power usage combine into the search cost, with Bézier curves smoothing the discrete path to reduce steering as an energy-saving measure. Although this approach integrates terrain geometry and vehicle dynamics, its primary aim is energy optimization rather than mitigating steering-induced mechanical wear. Steering reduction is considered only insofar as it lowers power demand, not as an explicit wear mitigation objective.

The ENav algorithm deployed on the *Perseverance* rover represents a recent advancement in tree-based path planning [13]. Operating in a receding-horizon framework with a 6 m planning

horizon, ENav executes only the initial 1 m segment before replanning. At each cycle, it constructs a 2.5D terrain heightmap from stereo imagery, evaluating paths using global and collision cost components. The global cost incorporates terrain roughness penalties, encouraging paths through smoother regions. Nevertheless, ENav's terrain cost prioritizes collision avoidance and general roughness without explicitly accounting for mechanical stresses induced by steering maneuvers—a known contributor to wheel degradation.

2.2. Sampling-Based Planning Methods

Sampling-based algorithms construct roadmaps by randomly sampling states from free configuration space and connecting them via local feasibility criteria. These methods excel in high-dimensional or continuous spaces where exhaustive discretization becomes prohibitive, offering probabilistic completeness and, in some variants, asymptotic optimality guarantees.

Rapidly-Exploring Random Trees (RRT) and their optimal variant RRT* have been adapted for planetary rover navigation. A terrain-aware extension of RRT* improves tree expansion by incorporating traversability checks operating directly on LiDAR point cloud data [14]. Candidate nodes are evaluated using geometric criteria including state-to-state distance, predicted rover attitude on terrain surfaces, and cost functions derived from these parameters, allowing the planner to avoid regions compromising stability or clearance.

Despite these advantages, practical performance remains sensitive to sampling efficiency, especially under kinodynamic constraints. Although the method accounts for general traversability through roll and pitch limitations, it does not explicitly target mechanical hazards such as high-friction steering loads unless additional cost terms are introduced.

Risk-aware path planning combining Rapidly-Exploring Random Graphs (RRG) with A* search targets terrain hazards not captured by purely geometric obstacle detection [15]. The method incorporates machine learning terrain classifiers trained on visual imagery to identify risks such as soft soil and sharp embedded rocks. While this improves sensitivity to subtle hazards, the approach depends strongly on classifier robustness and may fail on novel or poorly represented terrain types. As a multi-query method requiring graph construction for each planning cycle, it can be computationally expensive. Like many sampling-based approaches, it does not inherently guarantee smooth paths without additional post-processing.

2.3. Optimization-Based Planning Methods

Optimization-based methods formulate motion planning as minimizing a cost functional subject to kinematic, dynamic, and environmental constraints. Compared to search-based and sampling-based planners, these methods explicitly encode constraints within a unified optimization framework, enabling direct control over trajectory smoothness, curvature, clearance, and other motion characteristics. They typically produce smooth, continuous trajectories suitable for

execution.

Nonlinear programming (NLP) provides a flexible framework for generating optimal rover trajectories over complex terrain while accounting for vehicle dynamics and wheel–terrain interactions [16]. The NLP solver minimizes a utility functional encoding multiple objectives such as risk, traversal time, smoothness, and energy efficiency. Penalizing high roll and pitch angles discourages prolonged travel on steep slopes. By incorporating terrain geometry into the motion model, the optimizer avoids geometrically infeasible trajectories and correctly evaluates true 3D path length, which can differ substantially from planar projections in rough environments. The primary drawback is the computational cost as high-fidelity dynamics and detailed terramechanics models make real-time, high-rate replanning on resource-limited onboard systems impractical.

The Safe Convex Corridor (SCC) method formulates rover trajectory optimization as a sequence of convex subproblems, ensuring collision-free motion in cluttered 2D environments [17]. While effective for planar obstacle avoidance, it does not account for terrain roughness, limiting applicability in scenarios where wheel stress and steering hazards are critical.

2.4. Learning-Based Planning Methods

Learning-based planners use machine learning, most commonly Deep Reinforcement Learning (DRL) and learned heuristics, to map sensory inputs to navigation decisions. They can produce fast inference at runtime, but typically trade this for reduced interpretability, large training demands, and sensitivity to distribution shifts that affect generalization.

End-to-end rover navigation with DRL has been demonstrated using Proximal Policy Optimization (PPO) for lunar scenarios [18]. Trained in Gazebo with terrain derived from real lunar DEM data, the policy consumes depth images and LiDAR point clouds and outputs control actions directly. Safety is enforced through reward shaping that penalizes high roll and pitch and slope-based slip risk, encouraging avoidance of steep or unstable terrain. However, achieving stable policies requires extensive simulation data and training time, and performance can degrade under sim-to-real transfer. Moreover, risk measures such as slip prediction are often based on simplified cues (e.g., slope), which do not capture frictional and steering-related interactions relevant to wheel-wear modeling.

For global planning, DRL has also been applied to the resource-constrained shortest path (RCSP) problem using Deep Q-Networks (DQN) and Markov Decision Processes (MDP) [19]. In this formulation, constraints derived from lunar DEM data are embedded into a grid as penalty terms, including terramechanical costs based on squared slope, while internal resources such as thermal budget and power reserves are treated as path-dependent constraints. Although this yields a unified representation of environmental and vehicle-state constraints, it often restricts motion to cardinal directions and assumes uniform traversal time per grid cell for computational efficiency. Since traversal time in practice varies with slope and friction, this abstraction limits the ability to optimize continuous steering behavior and fine-grained maneuvers that matter for wheel wear.

2.5. Research Gap and Thesis Positioning

Across search-based, sampling-based, optimization-based, and learning-based rover planners surveyed above, terrain awareness is typically used to improve geometric safety and traversability, or to reduce energy use and slip risk. These objectives correlate with mobility performance, but they do not explicitly capture the steering effort demanded by a trajectory, especially when turning on rough or abrasive terrain. As a result, a planner can prefer a path that is collision-free and traversable while still containing short-radius turns or frequent steering changes that increase mechanical loading at the wheels.

Mitigations reported in the rover operations literature often act at the execution layer, for example by modifying how a commanded path is driven through speed control or arc-based primitives. While effective, these strategies do not change the upstream planning objective that selects the path in the first place. This thesis therefore targets the planning layer by introducing a computationally simple proxy for steering-related wear risk and embedding it directly into the planner cost evaluation.

The proposed approach penalizes trajectory curvature and integrates the same formulation into two planning paradigms: a kinodynamic RRT* pipeline with NLP-based steering and post-smoothing, and a lattice-based A* planner using motion primitives. This enables a controlled comparison between continuous and discrete planning under an identical terrain-aware steering objective.

3. Problem Formulation

This chapter establishes the mathematical foundations for the terrain-aware motion planning approaches developed in this thesis. Section 3.2 introduces the kinematic model of the LRU rover. Section 3.3 defines the operational constraints that govern feasible motion. Section 3.4 describes how terrain and obstacle information is represented. Section 3.6 presents the curvature-based formulation that serves as a proxy for steering-induced wheel wear. Finally, Section 3.1 formalizes the motion planning problem for both search-based path planning and sampling-based trajectory planning paradigms.

3.1. Motion Planning Problem

The motion planning frameworks considered in this work differ in their representation of the solution. *Path planning* produces a collision-free geometric path specifying a sequence of configurations (x, y, θ) without temporal information. *Trajectory planning* extends this to find a time-parameterized state trajectory $\mathbf{x}(t)$ and control inputs $\mathbf{u}(t)$ that respect system dynamics, kinematic constraints, and actuator limits.

Path Planning

Let the rover configuration be

$$q = (x, y, \theta) \in \mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1, \tag{3.1}$$

where $\mathcal{C}_{\text{free}} \subset \mathcal{C}$ denotes the collision-free subset. Given a start configuration $q_{\text{start}} \in \mathcal{C}_{\text{free}}$ and a goal region $\mathcal{G} \subset \mathcal{C}_{\text{free}}$, the path planning problem results in a continuous curve

$$\sigma : [0, 1] \to \mathcal{C}_{\text{free}}, \qquad \sigma(0) = q_{\text{start}}, \quad \sigma(1) \in \mathcal{G}, \tag{3.2}$$

which specifies a sequence of collision-free configurations without temporal information or control variables.

Trajectory Planning

For kinodynamic systems, the trajectory planning problem includes the evolution of state and control variables over time. The rover configuration is:

$$\mathbf{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} \in \mathbb{R}^2 \times \mathbb{S}^1, \tag{3.3}$$

with control input:

$$\mathbf{u}(t) = \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \in \mathbb{R}^2, \tag{3.4}$$

where $v(t)$ is the longitudinal velocity and $\omega(t)$ is the angular velocity.

Given an initial state $\mathbf{x}_0$ and a goal region $\mathcal{X}_g \subset \mathbb{R}^2 \times \mathbb{S}^1$, the trajectory planning problem solves for a time-parameterized trajectory

$$\mathbf{x} : [t_0, t_f] \to \mathbb{R}^2 \times \mathbb{S}^1, \qquad \mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) \in \mathcal{X}_g, \tag{3.5}$$

along with control variables $\mathbf{u}(t)$ subject to a kinematic model:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)). \tag{3.6}$$

Sampling-based Planning

Sampling-based planners construct a discrete representation of the configuration space (graph or tree) by drawing random samples from the configuration space $\mathcal{C}$ and employing a steering function to connect nearby collision-free configurations.

Representative algorithms include Probabilistic Roadmaps (PRM) [20] and Rapidly-exploring Random Trees (RRT) [21]. A PRM constructs a reusable roadmap by connecting random samples in $\mathcal{C}_{\text{free}}$, enabling path queries through standard graph search, and is particularly suited for multi-query planning. In contrast, a RRT grows a tree incrementally from the start configuration for efficient single-query planning. While standard RRT often yields suboptimal solutions, variants such as RRT* achieve asymptotic optimality through cost-aware rewiring, which progressively refines path quality as the tree grows [22].

For systems with differential constraints, simple linear interpolation in configuration space can produce infeasible trajectories. In such cases, a steering function that respects system dynamics must be employed to solve a boundary value problem (BVP) between two configurations.

Sampling-based methods scale well to high-dimensional spaces and avoid explicit discretization, but initial solutions are often suboptimal and require post-processing or refinement.

Trajectory Optimization via Optimal Control

Trajectory optimization formulates the motion planning problem as an optimal control problem [23]. The goal is to find the optimal trajectory $\mathbf{x}^*(t)$ and control input $\mathbf{u}^*(t)$ over the time interval $[t_0, t_f]$ that minimizes a cost functional of the form:

$$\min_{t_0, t_f, \mathbf{x}(t), \mathbf{u}(t)} \phi(t_0, \mathbf{x}_0, t_f, \mathbf{x}_f) + \int_{t_0}^{t_f} g(t, \mathbf{x}, \mathbf{u})\, dt \tag{3.7}$$

subject to:

- **System Dynamics:** $\dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{u})$

- **Constraints:** $c_{\min} < c(t, \mathbf{x}, \mathbf{u}) < c_{\max}$
- **Boundary Conditions:** $b_{\min} < b(t_0, \mathbf{x}_0, t_f, \mathbf{x}_f) < b_{\max}$,

where ϕ is a terminal cost term, and the integral term represents the running cost over the trajectory.

In order to solve this continuous-time optimal control problem, it is transformed into a discrete-time nonlinear program (NLP) in the following form:

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \quad & \bar{J}(z) \\ \text{subject to} \quad & f(z) = 0, \\ & g(z) \leq 0, \\ & z_{\text{low}} \leq z \leq z_{\text{upp}}, \end{aligned} \tag{3.8}$$

where z denotes the vector of decision variables consisting of discretized states and control inputs. The objective function $\bar{J}(z)$ represents the discretized cost function, $f(z)$ encodes equality constraints enforcing the system dynamics, initial conditions, and boundary conditions, and $g(z)$ encodes inequality constraints such as obstacle avoidance. The box constraints z_{low} and z_{upp} define lower and upper bounds on the decision variables. The resulting NLP is then solved using a numerical solver such as the interior-point method implemented in IPOPT [24].

Search-based Planning

Search-based planners discretize the configuration space into a graph and then search this graph for a lowest-cost path. A common discretization scheme is a state lattice where vertices represent discrete states and edges represent feasible transitions that respect the system's kinematic constraints.

On such a graph, Dijkstra's algorithm is the basic shortest-path search method [25]. Dijkstra finds the shortest paths by expanding vertices towards the goal in the order of increasing cost-to-come. A* search [26] speeds up the search by using a heuristic $h(n)$ that estimates the remaining cost to the goal. It expands vertices in order of increasing $f(n)$,

$$f(n) = g(n) + h(n), \tag{3.9}$$

prioritizing those with lower values. If $h(n)$ never overestimates the true cost-to-go, A* finds an optimal path on the discretized graph while expanding fewer vertices than Dijkstra.

Search-based methods provide completeness and optimality guarantees on the discretized graph, but solution quality is limited by the lattice resolution and computational cost grows with finer discretizations.

3.2. Rover Kinematic Model

Kinematic modeling provides the mathematical foundation for developing motion planning algorithms in autonomous systems. It describes the relationships that govern how control inputs

translate into vehicle motion. This section presents the kinematic model employed throughout this thesis, covering both continuous-time and discrete-time formulations.

Continuous-Time Model

The rover's configuration at time t consists of its position and orientation in the world frame:

$$\mathbf{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} \tag{3.10}$$

where $(x(t), y(t))$ represent the planar position of the rover's geometric center, and $\theta(t)$ denotes the heading angle measured counterclockwise from the positive x-axis. The control inputs are defined as:

$$\mathbf{u}(t) = \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3.11}$$

where $v(t)$ is the longitudinal velocity along the rover's forward direction in the body frame (positive forward, negative backward) and $\omega(t)$ is the angular velocity about the vertical axis (positive counterclockwise).

The kinematic evolution of the rover is governed by the standard non-holonomic differential equations:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ \omega(t) \end{bmatrix} = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.12}$$

Discrete-Time Model

For implementation purposes, the continuous-time kinematic model is discretized to enable numerical computation and forward integration of the rover's state over time. The continuous-time model is discretized using the Forward Euler method. The discrete-time state at time step k is denoted $\mathbf{x}_k = [x_k, y_k, \theta_k]^\top$, and the control input is $\mathbf{u}_k = [v_k, \omega_k]^\top$. The discrete state update equations are as follows:

$$x_{k+1} = x_k + v_k \cos(\theta_k)\,\Delta t \tag{3.13}$$
$$y_{k+1} = y_k + v_k \sin(\theta_k)\,\Delta t \tag{3.14}$$
$$\theta_{k+1} = \theta_k + \omega_k\,\Delta t \tag{3.15}$$
$$v_{k+1} = v_k + a_k\,\Delta t \tag{3.16}$$

where Δt is the time step between successive discrete states.

Wheel Kinematics

Individual wheel kinematics can be computed from body-frame velocities and are essential for evaluating wheel-level metrics related to steering effort and wear. The rover's velocity is assumed to align with the rover's x-axis, with no lateral velocity component ($v_y = 0$).

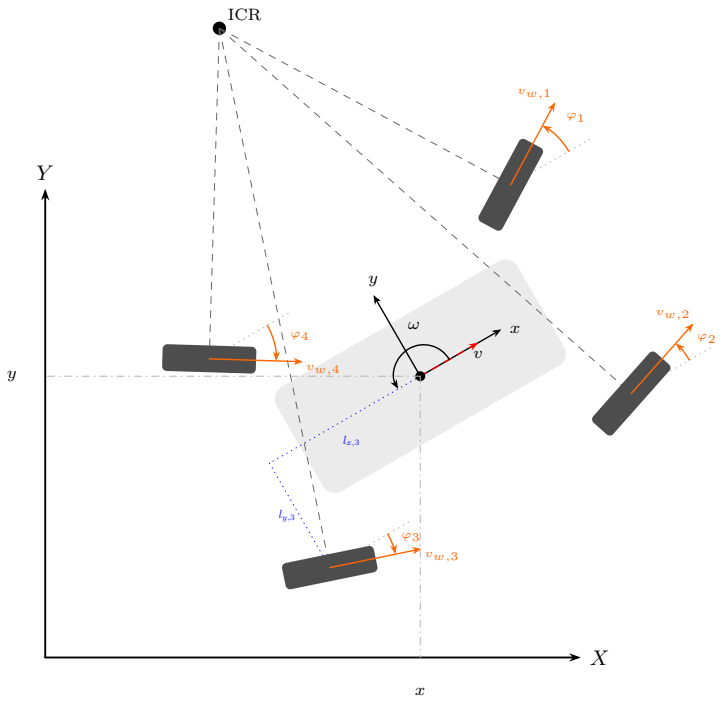


Figure 3.1.: Kinematic model of the LRU rover.

For the i-th wheel located at $(l_{x,i}, l_{y,i})$ relative to the rover's geometric center, the wheel velocity components in the body frame are:

$$v_{x,i} = v - l_{y,i}\,\omega \tag{3.17}$$

$$v_{y,i} = l_{x,i}\,\omega \tag{3.18}$$

From these components, the steering angle of the wheel follows as

$$\varphi_i = \arctan\left(\frac{v_{y,i}}{v_{x,i}}\right) \tag{3.19}$$

and the wheel-center speed is given by

$$v_{w,i} = \sqrt{v_{x,i}^2 + v_{y,i}^2} \tag{3.20}$$

These relations directly map the rover's body velocity and yaw rate to the kinematic quantities of each individual wheel.

3.3. Planning Constraints

The motion planning problem is subject to several classes of constraints that ensure the generated paths or trajectories are kinematically feasible and collision-free.

Kinematic Constraints

The rover's actuators and operational safety requirements impose bounds on the achievable velocities and accelerations. The rover's longitudinal velocity is bounded by:

$$v_{\min} \leq v_k \leq v_{\max}, \tag{3.21}$$

where these bounds define the admissible control inputs for the system, ensuring that the commanded velocities remain physically achievable and consistent with actuator and safety constraints. The rover's angular velocity is bounded by:

$$\omega_{\min} \leq \omega_k \leq \omega_{\max}, \tag{3.22}$$

where the bounds $\omega_{\min}$ and $\omega_{\max}$ reflect the maximum steering rates the vehicle can achieve without compromising stability. For symmetric steering capabilities, typically $\omega_{\min} = -\omega_{\max}$. The longitudinal acceleration is also bounded to ensure smooth motion and prevent sudden changes in velocity:

$$a_{\min} \leq a_k \leq a_{\max}. \tag{3.23}$$

Obstacle Avoidance

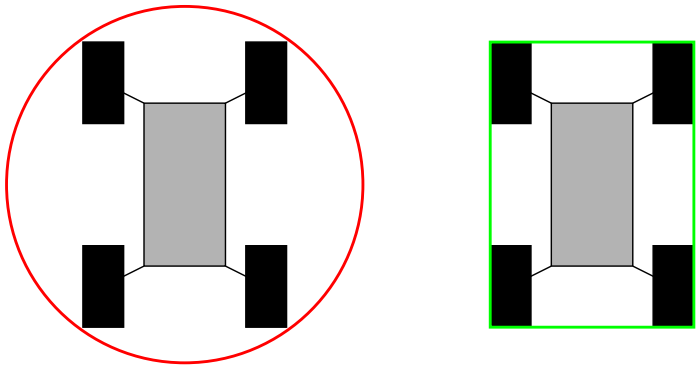


Figure 3.2.: Circular and rectangular footprint approximations. A circular footprint is used for its rotation invariance and low collision-checking cost, at the expense of increased conservatism.

The rover must remain collision-free along its trajectory. For the LRU rover, the footprint is approximated by a circle centered at the rover reference point. This choice makes collision checking independent of the rover orientation and yields a simple, conservative safety margin compared to an oriented rectangular footprint (Figure 3.2).

Let $R_{\text{footprint}}$ denote the radius of the circular approximation, chosen to enclose the rover body and wheels. The footprint at configuration $\mathbf{x}_k = (x_k, y_k, \theta_k)$ is:

$$\mathcal{F}(\mathbf{x}_k) = \left\{ \mathbf{p} \in \mathbb{R}^2 : \|\mathbf{p} - \mathbf{c}_k\| \leq R_{\text{footprint}} \right\}, \tag{3.24}$$

where $\mathbf{c}_k = (x_k, y_k)$ is the rover center position in the world frame.

Let $\mathcal{O} \subset \mathbb{R}^2$ denote the set of occupied or hazardous regions. Collision-free motion requires:

$$\mathcal{F}(\mathbf{x}_k) \cap \mathcal{O} = \emptyset \qquad \forall k. \tag{3.25}$$

3.4. Terrain Representation

The terrain representation used in this work is a binary roughness map that encodes surface abrasiveness information. Formally, the roughness map is defined as:

$$r(x, y) \in \{0, 1\}, \tag{3.26}$$

where $r(x, y) = 0$ indicates smooth terrain with low abrasiveness and $r(x, y) = 1$ indicates rough or abrasive terrain.

This binary roughness map is hand-crafted for the simulated environments and partitions the workspace into low-roughness and high-roughness regions. While real terrain varies along different levels of abrasiveness depending on surface texture and rock content, a binary map is used here to keep the terrain information simple, interpretable, and easy to control in experiments. It provides a clear separation between benign and hazardous terrain, making it straightforward to interpret the planned steering behavior relative to the terrain-aware cost terms.

3.5. Obstacle Representation

While the roughness map encodes surface abrasiveness, obstacle information captures geometric hazards that the rover cannot traverse. A common representation for obstacle information is the Occupancy Grid Map (OGM), which discretizes the environment into a regular 2D grid where each cell stores a binary value indicating whether the corresponding region is occupied. While occupancy grids are widely used due to their simplicity, the binary nature provides limited information for motion planning—cells are either fully occupied or fully free, with no notion of clearance or proximity to obstacles.

To address these limitations, the two planning paradigms in this work employ richer obstacle representations suited to their algorithmic structures.

Signed Distance Field

For the kinodynamic RRT* planner and the smoothing NLP, obstacles are represented by a signed distance field (SDF). The SDF is defined as:

$$d_{\text{SDF}}(x, y) \in \mathbb{R}, \tag{3.27}$$

where $d_{\text{SDF}}(x, y)$ returns the Euclidean distance from the continuous position (x, y) to the nearest obstacle boundary. By convention:

$$d_{\text{SDF}}(x, y) \begin{cases} > 0 & \text{in free space,} \\ = 0 & \text{on the obstacle boundary,} \\ < 0 & \text{inside obstacles.} \end{cases} \tag{3.28}$$

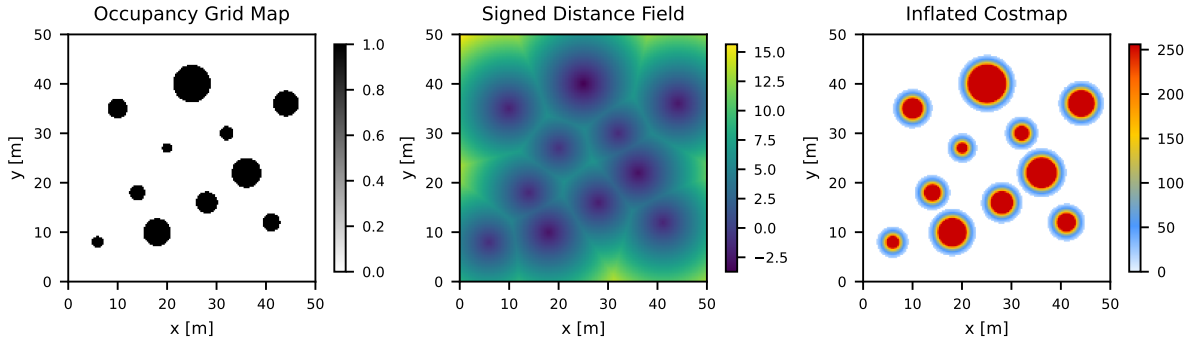


Figure 3.3.: Obstacle representations: Left: binary Occupancy Grid Map (OGM) where occupied cells are shown in black and free space in white. Middle: Signed Distance Field (SDF), where positive values (lighter shades) indicate distance to the nearest obstacle in free space and negative values (darker shades) indicate depth inside obstacles. Right: inflated costmap where obstacle cells are assigned lethal cost (dark red) and surrounding cells receive decaying inflation costs (gradient from red to cyan) based on proximity to obstacles.

This representation is continuous and piecewise differentiable, making it suitable for gradient-based trajectory optimization. Collision avoidance constraints can be formulated as:

$$d_{\text{SDF}}(x_k, y_k) \geq R_{\text{footprint}}, \tag{3.29}$$

ensuring that the rover center maintains sufficient clearance from obstacles.

Inflated Costmap

For the lattice-based A* planner, obstacles are represented on a 2D grid costmap constructed by inflating an occupancy grid. Each cell stores a traversal cost:

$$c_{\text{map}}(i, j) \in [0, 254], \tag{3.30}$$

where the cost values follow the Nav2 costmap convention:

- $c_{\text{map}}(i, j) = 254$: lethal cost, indicating cells occupied by obstacles,
- $c_{\text{map}}(i, j) \in (0, 253]$: inflated cost, assigned to free-space cells within the inflation radius to discourage proximity to obstacles,
- $c_{\text{map}}(i, j) = 0$: free space with no traversal penalty.

The inflation radius is set to match the rover's circular footprint radius $R_{\text{footprint}}$, ensuring that any configuration whose center lies in a cell with lethal cost would result in a collision. This discrete representation suits graph search on a state lattice and enables fast collision checks by evaluating the cost of grid cells overlapping the rover footprint.

Figure 3.3 illustrates different obstacle representations: the occupancy grid map, the SDF used for trajectory optimization, and the inflated costmap used for lattice-based search.

The obstacle representations are constructed independently of the binary roughness map. Obstacles are defined based on geometric properties that make areas physically infeasible to traverse, whereas the roughness map captures surface abrasiveness that makes steering maneuvers undesirable but not impossible. The two representations serve complementary roles: obstacles enforce hard feasibility constraints via collision checking, while roughness influences soft cost penalties via the terrain-aware steering cost.

3.6. Curvature as Wheel Wear Proxy

Path curvature and its rate of change are used as geometric indicators of steering effort during motion planning. The rover follows a trajectory whose curvature is defined by its forward velocity v and angular velocity ω. The curvature of the path is given by:

$$\kappa = \frac{\omega}{v}. \tag{3.31}$$

As shown in Figure 3.4, for a wheel located at a longitudinal offset $l_{x,i}$ from the rover center, the body rotation induces a lateral velocity component at the wheel,

$$v_{y,i} = l_{x,i}\,\omega. \tag{3.32}$$

Substituting $\omega = \kappa v$ in Equation (3.32) yields:

$$v_{y,i} = l_{x,i}\,\kappa\,v, \tag{3.33}$$

which shows that, for a fixed vehicle geometry and forward speed, tighter turns directly increase the lateral velocity experienced by each wheel. This lateral component corresponds to the steering motion that needs to be executed by the wheels and therefore provides a kinematic measure of steering effort. As a result, penalizing large values of curvature discourages tight turns in rough terrain regions where steering is undesirable.

Curvature alone, however, does not capture how abruptly the steering demand changes along the path. Rapid transitions between different curvature values require sudden adjustments in steering, even if the absolute curvature remains moderate. To address this, the rate of change of curvature is also penalized. In continuous time, it is defined as:

$$\dot{\kappa}(t) = \frac{d\kappa}{dt} = \frac{d}{dt}\left(\frac{\omega(t)}{v(t)}\right). \tag{3.34}$$

Applying the quotient rule yields:

$$\dot{\kappa}(t) = \frac{v(t)\cdot\dot{\omega}(t) - \omega(t)\cdot\dot{v}(t)}{v(t)^2}, \tag{3.35}$$

revealing that $\dot{\kappa}$ depends on both angular acceleration $\dot{\omega}$ and longitudinal acceleration $\dot{v}$. High values of $|\dot{\kappa}|$ indicate rapid changes in the steering command. Rapid steering transitions require wheels to quickly adjust their orientation angles. The wheel steering angle is given by $\varphi_i = \arctan(v_{y,i}/v_{x,i})$, and its time derivative is:

$$\dot{\varphi}_i = \frac{v_{x,i}\dot{v}_{y,i} - v_{y,i}\dot{v}_{x,i}}{v_{x,i}^2 + v_{y,i}^2}. \tag{3.36}$$

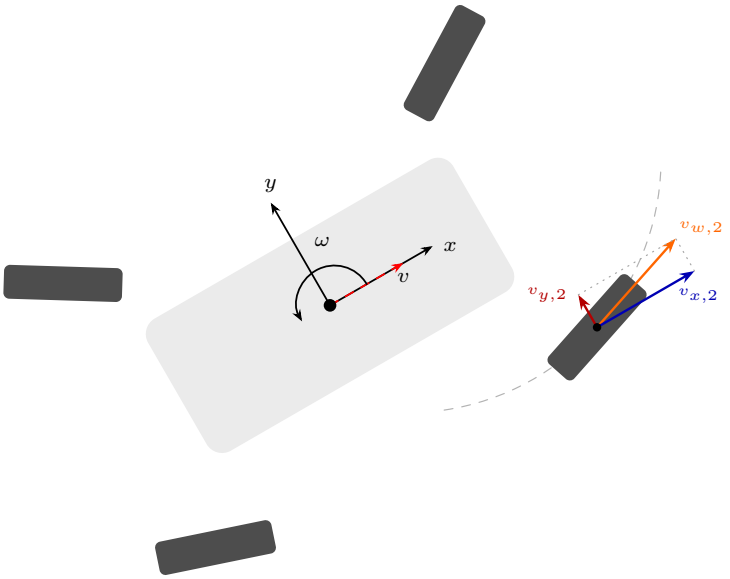


Figure 3.4.: Wheel velocity components in rover's frame, illustrating the lateral velocity component during a turning maneuver.

Substituting the wheel velocity components from Equations (3.17) and (3.18):

$$v_{x,i} = v - l_{y,i}\,\omega, \qquad \dot{v}_{x,i} = \dot{v} - l_{y,i}\,\dot{\omega}, \tag{3.37}$$

$$v_{y,i} = l_{x,i}\,\omega, \qquad \dot{v}_{y,i} = l_{x,i}\,\dot{\omega}, \tag{3.38}$$

into Equation (3.36) yields:

$$\dot{\varphi}_i = \frac{(v - l_{y,i}\,\omega)(l_{x,i}\,\dot{\omega}) - (l_{x,i}\,\omega)(\dot{v} - l_{y,i}\,\dot{\omega})}{(v - l_{y,i}\,\omega)^2 + (l_{x,i}\,\omega)^2}. \tag{3.39}$$

Expanding the numerator:

$$\dot{\varphi}_i = \frac{l_{x,i}\,v\,\dot{\omega} - l_{x,i}\,l_{y,i}\,\omega\,\dot{\omega} - l_{x,i}\,\omega\,\dot{v} + l_{x,i}\,l_{y,i}\,\omega\,\dot{\omega}}{(v - l_{y,i}\,\omega)^2 + (l_{x,i}\,\omega)^2}, \tag{3.40}$$

the cross terms cancel, leaving:

$$\dot{\varphi}_i = \frac{l_{x,i}(v\,\dot{\omega} - \omega\,\dot{v})}{(v - l_{y,i}\,\omega)^2 + (l_{x,i}\,\omega)^2}. \tag{3.41}$$

The numerator $v\,\dot{\omega} - \omega\,\dot{v}$ is proportional to $\dot{\kappa}$ through Equation (3.35), since $\dot{\kappa} = (v\,\dot{\omega} - \omega\,\dot{v})/v^2$. This confirms that the wheel steering rate $\dot{\varphi}_i$ depends directly on the rate of change of path curvature, establishing $\dot{\kappa}$ as a meaningful proxy for steering aggressiveness at the wheel level.

For simplicity and ease of computation, the curvature rate is approximated using backward differences:

$$\dot{\kappa}_k \approx \frac{\kappa_k - \kappa_{k-1}}{\Delta t}, \qquad k = 1, \ldots, N-1. \tag{3.42}$$

Large values of $|\dot{\kappa}_k|$ indicate abrupt changes in steering demand along the trajectory. Penalizing this term encourages smoother curvature profiles, resulting in more gradual wheel steering changes along the trajectory.

Curvature κ and its rate of change $\dot{\kappa}$ provide a simple and computationally efficient way to bias the planner toward gentler steering behavior. These quantities depend only on the planned path geometry and velocity profile, making them well suited for inclusion in the cost formulation of both planning paradigms considered.

3.7. Summary

This chapter has established the mathematical foundations for terrain-aware motion planning. First, Section 3.1 formalized the path and trajectory planning problems and provided background on sampling-based, search-based, and optimization-based planning paradigms. Section 3.2 introduced the kinematic model of the LRU rover in both continuous and discrete time, along with the wheel-level kinematics required for steering analysis. Section 3.3 defined the kinematic bounds on velocity and acceleration, as well as the collision avoidance constraints using a circular footprint approximation. Section 3.4 described the binary roughness map for encoding terrain abrasiveness and the two obstacle representations used by the planners: signed distance fields for trajectory optimization and grid costmaps for lattice-based search. Finally, Section 3.6 formulated path curvature and its rate of change as geometric proxies for steering effort, providing computationally efficient cost terms that penalize aggressive turning on rough terrain.

The next chapter builds upon these foundations to develop two terrain-aware planning frameworks: a kinodynamic RRT* planner with trajectory optimization and a lattice-based A* planner with terrain-aware edge costs.

4. Methodology

This chapter formalizes the two terrain-aware planning frameworks employed in this thesis and describes how the curvature-based terrain-aware cost model introduced in Chapter 3 is realized in the motion planning algorithms. Two planning approaches are presented: a sampling-based kinodynamic planner followed by an optimization-based smoothing step, and a search-based lattice planner. Both use the same terrain-aware cost structure, but differ in how they represent and explore the state space.

The sampling-based planner operates in the continuous configuration space, generating dynamically feasible trajectories through kinodynamic steering. The resulting trajectory provides an initial guess for the subsequent smoothing step. This approach produces smooth trajectories while respecting kinematic constraints.

The search-based planner discretizes the configuration space into a state lattice and composes precomputed motion primitives using graph search. By trading continuous flexibility for a structured representation, it achieves predictable computation times and efficient exploration, which is advantageous for real-time deployment.

The remainder of this chapter is organized as follows. Section 4.1 presents the sampling-based kinodynamic planner, describing the RRT* framework, the hybrid sampling strategy, the NLP-based steering function, and the NLP-based smoothing step. Terrain-aware curvature costs are incorporated in the final NLP smoothing step. Section 4.2 then introduces the search-based planner, detailing the lattice discretization, offline generation of a minimal motion primitive set, geometric construction of curvature-constrained primitives, and planning with A* graph search including the terrain-aware cost formulation.

4.1. Sampling-Based Kinodynamic Planning

4.1.1. Algorithmic Framework

The sampling-based planner incrementally grows a tree of dynamically feasible trajectories in the continuous state space. Candidate states are generated using a kinematics-aware rejection-corridor sampling strategy, and trajectories connecting states are produced via the NLP-based steering function to ensure feasibility with respect to the rover's kinematic constraints and control limits. Tree expansion continues until a feasible trajectory to the goal is found or the maximum iteration budget is exceeded. This trajectory is then used to warm-start an NLP-based

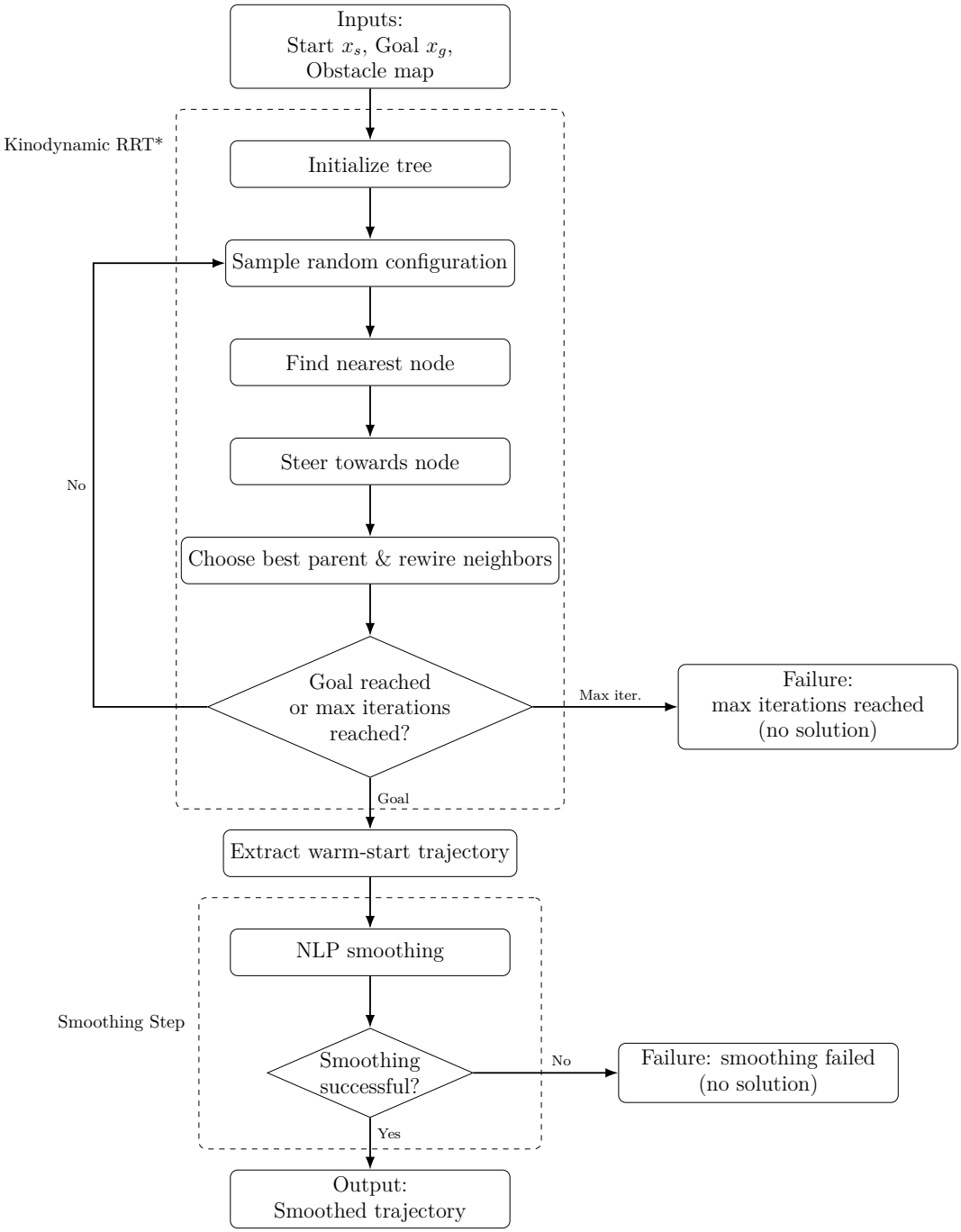


Figure 4.1.: A block diagram of the kinodynamic RRT* planning framework with NLP-based steering and a post-smoothing step, showing the main components of the framework and data flow.

smoothing step that incorporates terrain-aware curvature penalties. The smoother produces the final planned trajectory.

A block diagram of the overall planning framework is shown in Figure 4.1, illustrating the main components of the framework and data flow.

4.1.2. Sampling Strategy

Efficient sampling plays a critical role in the performance of kinodynamic sampling-based planners. Each candidate sample requires solving an expensive steering function to produce a dynamically feasible connection, but naive uniform sampling, even with goal biasing, often generates samples for which the steering attempt fails, wasting computational effort and slowing convergence.

A hybrid sampling strategy addresses this inefficiency by combining conservative rejection sampling with goal-directed corridor sampling. The rejection component filters out samples that are unlikely to yield feasible connections based on inexpensive kinematic checks. The corridor component, activated once sufficient tree structure exists, biases sampling toward promising regions between the current tree and goal while preserving exploration. Together, these mechanisms balance exploration and exploitation, improving convergence without sacrificing robustness. Algorithm 1 summarizes the procedure.

Algorithm 1 Rejection-Corridor Sampling

Input: Current iteration n, tree vertices $\mathcal{V}$, goal state x_{goal}, corridor width w_{corridor}, activation iteration N_{act}, corridor sampling probability η_{corr}, goal bias p_{goal}
Output: Sample configuration x_{sample}
if $\textsc{Random}() < p_{\text{goal}}$ **then** ▷ Goal biasing
  return x_{goal}
end if
if $n \geq N_{\text{act}}$ **and** $\textsc{Random}() < \eta_{\text{corr}}$ **then** ▷ Corridor sampling
  $x_{\text{promising}} \leftarrow \textsc{SelectPromisingVertex}(\mathcal{V}, x_{\text{goal}})$
  $x_{\text{sample}} \leftarrow \textsc{SampleFromCorridor}(x_{\text{promising}}, x_{\text{goal}}, w_{\text{corridor}})$
  return x_{sample}
else ▷ Rejection sampling
  repeat
    $x_{\text{rand}} \leftarrow \textsc{UniformRandom}()$
    $x_{\text{near}} \leftarrow \textsc{GetNearestVertex}(x_{\text{rand}})$
  until $\textsc{Accept}(x_{\text{rand}}, x_{\text{near}})$
  return x_{rand}
end if

Rejection Heuristic

In a kinodynamic RRT* framework, extending the tree toward a sampled configuration requires solving a nonlinear steering problem to generate a dynamically feasible trajectory. This steering step constitutes one of the most computationally expensive components of the planner. Consequently, attempting to connect every uniformly sampled configuration is inefficient, as many samples are trivially unreachable under kinematic constraints or lie in collision.

To mitigate this issue, a conservative rejection heuristic is applied to filter out unpromising samples before invoking the steering function. The rejection tests are designed to be computationally inexpensive while eliminating samples that are provably infeasible.

The acceptance function Accept in Algorithm 1 evaluates two criteria:

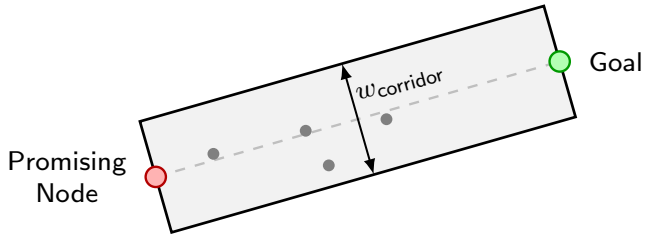


Figure 4.2.: Illustration of corridor-based sampling toward the goal.

Kinematic Reachability: The sampler estimates the minimum time required to reach a candidate configuration x_{rand} from the nearest tree vertex x_{near} under the rover's maximum linear and angular velocities as follows:

$$t_{\text{linear}} = \frac{\|x_{\text{rand}} - x_{\text{near}}\|}{v_{\max}}, \quad t_{\text{angular}} = \frac{|\Delta\theta|}{\omega_{\max}} \tag{4.1}$$

where $\Delta\theta$ denotes the angular difference between the heading at x_{near} and the direction toward x_{rand}. Samples are rejected if either time estimate exceeds a predefined threshold, since such configurations cannot be reached within a reasonable steering horizon.

Obstacle Clearance: The sampler queries the signed distance field at x_{rand} and rejects samples in collision ($d_{\text{SDF}} \leq 0$). This prevents redundant steering attempts toward configurations that are immediately infeasible due to collisions.

Although these kinematic checks are conservative because they assume maximum linear and angular velocities, they significantly reduce the number of failed steering attempts, thereby improving overall planning efficiency.

Corridor Sampling

While uniform rejection sampling ensures global exploration, it can become inefficient once the search tree has identified regions that are directionally aligned with the goal. In such cases, continuing to sample uniformly can slow convergence.

To address this issue, a corridor-based sampling strategy is introduced after an initial exploration phase. Once the iteration count exceeds the activation threshold N_{act}, corridor sampling is enabled with probability η_{corr}, while rejection sampling remains active otherwise to preserve global coverage. This probabilistic blending maintains the exploratory properties of RRT* while biasing sampling toward promising regions of the search space.

During corridor sampling, a promising vertex is selected from the existing tree based on directional alignment with the goal and sufficient separation from the goal state. A sample is then generated within a corridor defined along the straight-line segment connecting this vertex to the goal. Lateral perturbations are bounded by the corridor width w_{corridor}. The concept is illustrated in Figure 4.2.

By alternating between corridor-based and rejection sampling, the planner balances informed exploitation with continued exploration, leading to improved convergence without sacrificing probabilistic completeness.

4.1.3. Steering NLP Formulation

The steering NLP computes a trajectory from a start state $(x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}}, v_{\text{start}}, \omega_{\text{start}})$ to a target position $(x_{\text{target}}, y_{\text{target}})$ while satisfying dynamics and control limits. The problem is formulated as follows:

$$
\begin{aligned}
\min_{z \in \mathbb{R}^{6N+1}} \quad & J_{\text{steer}}(z) = w_T \cdot T + \frac{1}{N} \sum_{k=0}^{N-1} \left[w_a \cdot a_k^2 + w_\omega \cdot \omega_k^2 \right] \\
\text{s.t.} \quad & x_{k+1} = x_k + \Delta t \, v_k \cos(\theta_k), && k = 0, \ldots, N-2 \\
& y_{k+1} = y_k + \Delta t \, v_k \sin(\theta_k), && k = 0, \ldots, N-2 \\
& \theta_{k+1} = \theta_k + \Delta t \, \omega_k, && k = 0, \ldots, N-2 \\
& v_{k+1} = v_k + \Delta t \, a_k, && k = 0, \ldots, N-2 \\
& d_{\text{SDF}}(x_k, y_k) \geq R_{\text{footprint}}, && k = 0, \ldots, N-1 \\
& a_{\min} \leq a_k \leq a_{\max}, && k = 0, \ldots, N-1 \\
& v_{\min} \leq v_k \leq v_{\max}, && k = 0, \ldots, N-1 \\
& \omega_{\min} \leq \omega_k \leq \omega_{\max}, && k = 0, \ldots, N-1 \\
& (x_0, y_0, \theta_0, v_0, \omega_0) = (x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}}, v_{\text{start}}, \omega_{\text{start}}) \\
& (x_{N-1}, y_{N-1}) = (x_{\text{target}}, y_{\text{target}}) \\
& T > 0,
\end{aligned}
$$

with the following decision vector:

$$
z = \begin{pmatrix} x_0, \ldots, x_{N-1} \\ y_0, \ldots, y_{N-1} \\ \theta_0, \ldots, \theta_{N-1} \\ v_0, \ldots, v_{N-1} \\ \omega_0, \ldots, \omega_{N-1} \\ a_0, \ldots, a_{N-1} \\ T \end{pmatrix},
$$

with N denoting the number of discretization steps and T the total trajectory duration.

Feasibility is enforced through four constraint groups. The dynamics constraints propagate (x, y, θ, v) forward in time using the rover's kinematic model, ensuring that consecutive states are consistent with the control inputs. Collision avoidance is enforced by constraining the rover's footprint to remain collision-free at each discretization point, $d_{\text{SDF}}(x_k, y_k) \geq R_{\text{footprint}}$. Boundary constraints constrain the trajectory to start at the given start state and force the final position to match the target position.

Finally, box constraints bound a_k, v_k, and ω_k within actuator limits and enforce $T > 0$.

The NLP is solved using the IPOPT nonlinear solver through the CasADi optimization library. In cases where the solver fails to converge or the problem is deemed infeasible, the steering function reports failure and the corresponding sample is discarded from the tree expansion.

4.1.4. Terrain-Aware Trajectory Smoothing

After the kinodynamic RRT* returns a feasible trajectory, this solution is passed as a warm-start to a second NLP that refines the trajectory with terrain-aware objectives. This two-stage design keeps tree expansion efficient by reserving the more expensive terrain-aware optimization for a single post-processing step, rather than evaluating curvature penalties inside every steering call.

The smoothing NLP extends the steering formulation in three key ways:

1. **Terrain-aware cost terms:** The objective function incorporates curvature-based penalties that are activated only in rough terrain, penalizing steering maneuvers that contribute to wheel wear.
2. **Angular acceleration as a decision variable:** The angular velocity derivative $\dot{\omega}_k$ is introduced as an explicit decision variable with box constraints, enabling direct control over steering smoothness and preventing abrupt changes in angular velocity.
3. **Full terminal state constraints:** While the steering NLP fixes only the terminal position to connect states during tree growth, the smoothing formulation enforces the complete goal state including heading and velocity components.

The smoothing NLP problem is formulated as follows:

$$
\begin{aligned}
\min_{z\in\mathbb{R}^{7N+1}} \quad & J_{\text{smooth}}(z) = w_T \cdot T + \frac{1}{N}\sum_{k=0}^{N-1}\left[w_a \cdot a_k^2 + w_\omega \cdot \omega_k^2 + r_k \cdot \left(w_\kappa \cdot \kappa_k^2 + w_{\dot{\kappa}} \cdot \dot{\kappa}_k^2\right)\right] \\
\text{s.t.} \quad & x_{k+1} = x_k + \Delta t\, v_k \cos(\theta_k), && k = 0,\dots,N-2 \\
& y_{k+1} = y_k + \Delta t\, v_k \sin(\theta_k), && k = 0,\dots,N-2 \\
& \theta_{k+1} = \theta_k + \Delta t\, \omega_k, && k = 0,\dots,N-2 \\
& v_{k+1} = v_k + \Delta t\, a_k, && k = 0,\dots,N-2 \\
& \omega_{k+1} = \omega_k + \Delta t\, \dot{\omega}_k, && k = 0,\dots,N-2 \\
& d_{\text{SDF}}(x_k, y_k) \geq R_{\text{footprint}}, && k = 0,\dots,N-1 \\
& a_{\min} \leq a_k \leq a_{\max}, && k = 0,\dots,N-1 \\
& v_{\min} \leq v_k \leq v_{\max}, && k = 0,\dots,N-1 \\
& \omega_{\min} \leq \omega_k \leq \omega_{\max}, && k = 0,\dots,N-1 \\
& \dot{\omega}_{\min} \leq \dot{\omega}_k \leq \dot{\omega}_{\max}, && k = 0,\dots,N-1 \\
& (x_0, y_0, \theta_0, v_0, \omega_0) = (x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}}, v_{\text{start}}, \omega_{\text{start}}) \\
& (x_{N-1}, y_{N-1}, \theta_{N-1}, v_{N-1}, \omega_{N-1}) = (x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}}, v_{\text{goal}}, \omega_{\text{goal}}) \\
& T > 0,
\end{aligned}
\tag{4.2}
$$

with the extended decision vector:

$$z = \begin{pmatrix} x_0, \dots, x_{N-1} \\ y_0, \dots, y_{N-1} \\ \theta_0, \dots, \theta_{N-1} \\ v_0, \dots, v_{N-1} \\ \omega_0, \dots, \omega_{N-1} \\ \dot{\omega}_0, \dots, \dot{\omega}_{N-1} \\ a_0, \dots, a_{N-1} \\ T \end{pmatrix}.$$

The smoother objective function J_{smooth} augments the steering cost function with terrain-dependent curvature penalties. The terrain indicator $r_k \in [0, 1]$ specifies whether the k-th state lies in rough terrain. In smooth regions ($r_k = 0$), the curvature terms vanish and the cost reduces to time and control effort. The terrain indicator r_k is obtained through B-spline interpolation of the roughness map to ensure smooth transitions from rough to smooth terrain and vice versa. The curvature κ_k and its rate of change $\dot{\kappa}_k$ are computed from the control variables as defined in Section 3.6.

The introduction of $\dot{\omega}_k$ as a decision variable, together with the dynamics constraint $\omega_{k+1} = \omega_k + \Delta t\, \dot{\omega}_k$, allows the optimizer to reason explicitly about angular acceleration. The box constraints $\dot{\omega}_{\min} \leq \dot{\omega}_k \leq \dot{\omega}_{\max}$ prevent abrupt steering transitions that would otherwise cause aggressive wheel reorientation. This formulation provides finer control over trajectory smoothness compared to the steering NLP.

The full terminal state constraint ensures that the smoothed trajectory terminates at the exact goal configuration, including heading and velocity, rather than only matching the goal position as in the steering NLP.

4.2. Search-Based Planning in State Lattice

4.2.1. Algorithmic Framework

The search-based planner discretizes the rover's configuration space into a state lattice, where each state represents a quantized position and heading. Edges correspond to kinematically feasible trajectory segments connecting these states. The planner employs the A* algorithm to find minimum-cost paths, building upon the SMAC planner framework in Nav2 [11] with extensions for terrain-aware cost evaluation. By precomputing motion primitives offline and reusing them during search, the approach ensures kinematic feasibility and achieves computation times suitable for real-time execution.

4.2.2. State Lattice Representation

The continuous configuration space $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$ is discretized into a regular lattice, where positions are quantized uniformly to grid cells of resolution Δ_{xy} and headings into N_θ angular bins. A

(a) Uniform: $\theta_m = m \cdot \frac{2\pi}{N_\theta}$, $m \in \{0, \ldots, 15\}$ (b) Non-uniform: $\theta_m = \arctan(j/i)$

Figure 4.3.: Comparison of heading discretization strategies for straight-line motion primitives on a 5×5 state lattice. (a) Uniform angular spacing ($\Delta\theta = 22.5°$) causes 8 of 16 primitives (red) to miss all lattice nodes. (b) Non-uniform discretization aligns angles with lattice geometry, ensuring all 16 primitives terminate exactly on lattice nodes.

lattice state is represented as a tuple (i, j, m) corresponding to the continuous configuration:

$$(x, y, \theta) = (x_{\min} + i \cdot \Delta_{xy},\ y_{\min} + j \cdot \Delta_{xy},\ \theta_m),$$

where $i, j \in \mathbb{Z}$ are grid indices and $m \in \{0, 1, \ldots, N_\theta - 1\}$ is the heading index.

Heading angles θ_m are selected non-uniformly to ensure that straight-line motion primitives terminate exactly at grid cell centers. Uniform heading discretization fails for more than eight orientations, since most evenly spaced angles do not align with lattice vertices [27]. By sampling headings corresponding to rational slopes, the planner can represent long straight-line motions accurately, as shown in Figure 4.3, preventing cumulative discretization errors during trajectory composition.

4.2.3. Motion Primitive Library

Motion primitives are short, kinematically feasible path segments that connect pairs of lattice states and form the discrete action set of the state lattice planner. Each primitive represents a locally executable motion of the rover and guarantees feasibility with respect to the kinematic model by explicitly enforcing a minimum turning radius constraint $R_{\min}$.

Each motion primitive connects a reference start configuration to a discrete lattice endpoint and is parameterized as a sequence of at most two segments, an optional initial or terminal straight segment and a circular arc with radius $R \geq R_{\min}$.

By restricting the planner's search to this set of curvature-constrained primitives, all candidate paths generated during graph search are kinematically valid by construction, eliminating the need for additional feasibility checks during planning.

Minimal Primitives Set Generation

The motion primitive library is constructed to form a minimal control set, i.e., the smallest collection of primitives sufficient to reach any lattice state that is kinematically reachable by

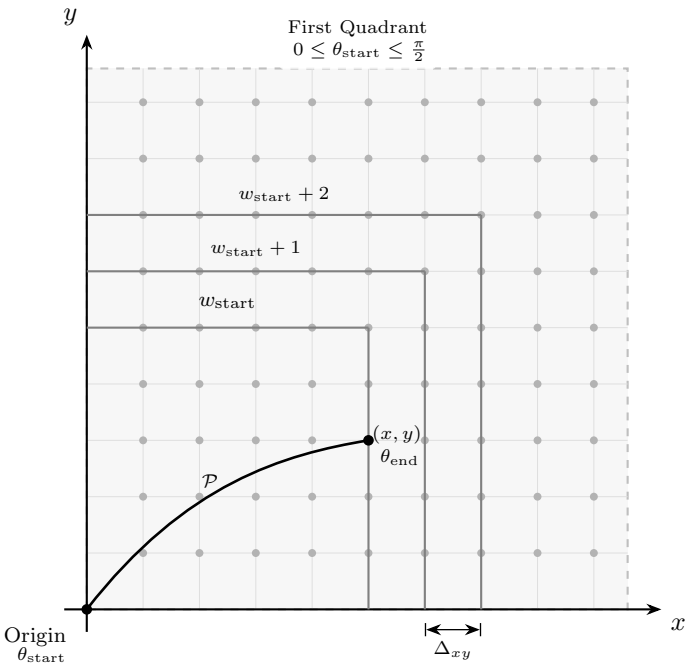


Figure 4.4.: Wavefront points generation for candidate primitive endpoints.

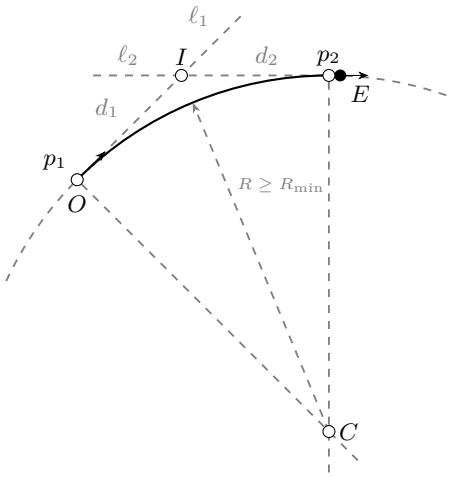
the rover [28]. Minimizing the number of available primitives directly reduces the branching factor during graph search, improving planning efficiency without sacrificing reachability or completeness.

The primitive set is generated offline using a wavefront expansion procedure centered at a reference lattice state. Candidate endpoints are evaluated at increasing distances from the origin, and a motion primitive is considered minimal only if it cannot be decomposed from a sequence of previously accepted primitives. This enforces minimality by rejecting redundant motions that do not expand the reachable set of lattice states.

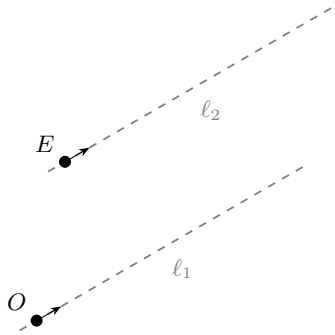
An important property exploited during primitive generation is the translational invariance of motion primitives. Due to the uniform spatial discretization of the lattice, the geometric structure of a feasible primitive depends only on the relative displacement between its start and end states, not on their absolute position in the environment. As a result, primitives need only be generated once for each discrete heading at a reference location and can be translated to any other lattice state while preserving feasibility and optimality.

The generation process is performed independently for each admissible heading, and geometric symmetry is used to populate the remaining orientations. In addition to forward motion primitives, in-place rotation primitives are included explicitly to allow pure heading changes in confined environments where translation is not feasible.

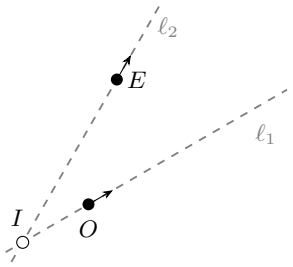
An illustration of the wavefront expansion principle used to identify candidate primitive endpoints is shown in Figure 4.4 and a detailed description of the wavefront-based generation procedure is provided in Appendix A.1, where the full algorithmic steps are summarized.



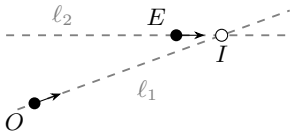
(a) Feasible arc construction with valid intersection I.



(b) Rejected: parallel and not colinear lines.



(c) Rejected: intersection point I lies behind start point O.



(d) Rejected: intersection point I lies beyond end point E.

Figure 4.5.: Geometric cases encountered during motion primitive generation. The algorithm constructs curvature-continuous trajectories by connecting start point O with heading defined by line ℓ_1 to endpoint E with heading defined by line ℓ_2.

Primitive Construction

Given a start pose and a target lattice pose, the construction is formulated geometrically using two directed lines, one originating from the start pose with heading θ_{start}, and the other originating from the endpoint with heading θ_{end}. When a feasible solution exists, these two directed lines uniquely define a circular arc tangent to both lines. Straight-line segments connect the arc to the start or end poses, yielding a curvature-continuous path that satisfies the minimum turning radius constraint $R \geq R_{\min}$.

The relative orientation and placement of the two directed lines can result in several geometric configurations, which determine whether a valid primitive exists. These cases are illustrated in Figure 4.5.

If the directed lines are parallel and collinear, the primitive simplifies to a pure straight-line

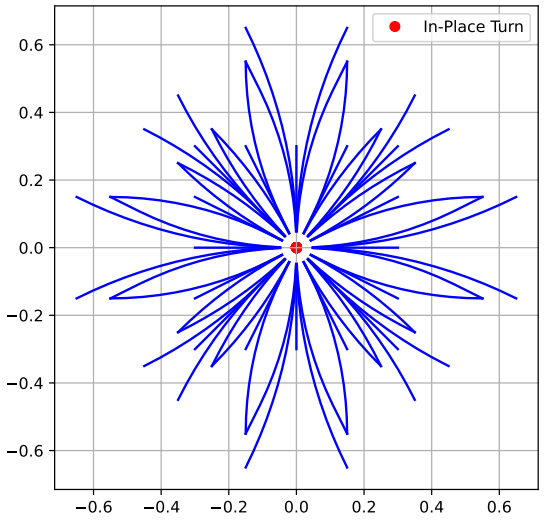


Figure 4.6.: The generated motion primitive set for all heading discretization bins for a minimum turning radius of 1 m.

primitive. If the lines are parallel but offset, no curvature-continuous path exists that respects the heading constraints and the configuration is rejected. Similarly, when the intersection of the directed lines lies opposite to the direction of motion of either pose, the resulting arc would require backward motion or overshoot and is therefore infeasible.

When the intersection lies between the two poses and the minimum-radius constraint is satisfied, a unique tangent circle can be constructed. The complete primitive is formed by concatenating the straight segments and the circular arc, and discretizing the resulting path at a fixed spatial resolution to obtain $\mathcal{P} = \{(x_k, y_k, \theta_k)\}_{k=0}^{N_p}$.

The generated motion primitive set for a minimum turning radius of 1 m is illustrated in Figure 4.6 and the primitive geometric construction procedure is provided in Appendix A.2.

4.2.4. Graph Search with A*

Given the motion primitive library $\mathcal{M}$, planning reduces to a shortest-path search on the state-lattice graph. Each lattice state represents a discretized rover pose, and successors are generated by applying the primitives associated with the current heading bin. The search is performed using A*, which orders candidate states by

$$f(s) = g(s) + h(s), \tag{4.3}$$

where $g(s)$ is the accumulated cost-to-come and $h(s)$ is an admissible heuristic estimate of the remaining cost-to-go.

For a state s, each primitive $\mathcal{P} \in \mathcal{M}$ produces a successor s' and an associated discretized path. A primitive is discarded if any pose along the path is in collision. For each valid successor, the cost-to-come is updated as

$$g(s') = g(s) + c(s, s', \mathcal{P}), \tag{4.4}$$

and if this improves the best known cost for s', the predecessor pointer of s' is set to s and s' is inserted into the open set. The search terminates when the goal state is selected for expansion. The final path is reconstructed by backtracking predecessor pointers and concatenating the corresponding motion primitives,

$$\text{Path} = \mathcal{P}_1 \oplus \mathcal{P}_2 \oplus \cdots \oplus \mathcal{P}_N.$$

Edge Cost Function

The cost $c(s, s', \mathcal{P})$ of traversing from state s to successor s' via primitive $\mathcal{P}$ is defined as:

$$c(s, s', \mathcal{P}) = c_{\text{base}} + c_{\text{maneuver}} + c_{\text{terrain}}, \tag{4.5}$$

combining path progress, costmap traversal, maneuver quality, and terrain-dependent costs.

Base Travel Cost The base cost reflects the primitive trajectory length $L_{\mathcal{P}}$ normalized by grid resolution and weighted by the environmental traversal difficulty:

$$c_{\text{base}} = \frac{L_{\mathcal{P}}}{\Delta_{xy}} \cdot \left(w_{\text{distance}} + w_{\text{costmap}} \cdot \bar{c}_{\mathcal{P}}\right), \tag{4.6}$$

where $\bar{c}_{\mathcal{P}}$ denotes the maximum normalized costmap value encountered along the primitive.

Maneuver Quality Cost To discourage inefficient or oscillatory motion patterns, maneuver costs penalize turning behavior relative to the incoming trajectory:

$$c_{\text{maneuver}} = \begin{cases} 0, & \text{if } \mathcal{P} \text{ is straight,} \\ c_{\text{base}} \cdot w_{\text{turn}}, & \text{if } \mathcal{P} \text{ continues the parent turn direction,} \\ c_{\text{base}} \cdot (w_{\text{turn}} + w_{\text{change}}), & \text{if } \mathcal{P} \text{ reverses turn direction,} \\ w_{\text{rot}} \cdot (1 + w_{\text{costmap}} \cdot \bar{c}_{\mathcal{P}}), & \text{if } L_{\mathcal{P}} \approx 0. \end{cases} \tag{4.7}$$

The final case assigns a fixed penalty to in-place rotation primitives, preventing zero-cost transitions while allowing reorientation when necessary.

Terrain-Aware Curvature Cost The terrain-dependent cost penalizes high-curvature maneuvers on abrasive terrain:

$$c_{\text{terrain}} = r(s) \cdot \left(w_\kappa \cdot \kappa_{\mathcal{P}}^2 \cdot \frac{L_{\mathcal{P}}}{\Delta_{xy}} + w_{\dot{\kappa}} \cdot (\kappa_{\mathcal{P}} - \kappa_{\text{parent}})^2 \right), \tag{4.8}$$

where $r(s) \in \{0, 1\}$ is the terrain roughness at state s, and $\kappa_{\mathcal{P}}$ and κ_{parent} denote the curvatures of the current candidate and the parent primitives, respectively.

4.3. Summary

This chapter presented two complementary motion planning approaches for terrain-aware rover navigation: a kinodynamic RRT*-based sampling planner and a lattice-based search planner. The sampling-based planner incrementally constructs dynamically feasible trajectories using rejection and corridor-guided sampling, followed by a terrain-aware smoothing stage that accounts for wheel wear-inducing maneuvers. In contrast, the lattice-based planner relies on a precomputed library of kinematically feasible motion primitives to efficiently explore a structured state lattice while explicitly incorporating terrain-dependent costs during graph search.

The following chapter evaluates these planning approaches quantitatively and qualitatively in hand-crafted navigation scenarios and assesses their generalization through randomized scenarios. Their performance is compared in terms of steering smoothness, wheel curvature, path quality, and computational efficiency, providing empirical insight into how the proposed terrain-aware costs influence rover navigation on rough and abrasive terrain.

5. Experimental Evaluation

This chapter evaluates the terrain-aware motion planning frameworks presented in Chapter 4. The evaluation proceeds in three stages. First, the sampling strategies developed for the kinodynamic RRT* planner are validated in Section 5.1. Second, both planners are compared across hand-crafted navigation scenarios in Section 5.3. Third, robustness and generalization are assessed through randomized trials in Section 5.4 and computational efficiency is evaluated in Section 5.5.

All experiments were conducted on a PC equipped with an AMD Ryzen 7 6800H CPU (8 cores, 3.2 GHz) running Ubuntu 22.04. The RRT*+Smoother planner was implemented in Python 3 using CasADi 3.7 with the IPOPT solver for trajectory optimization. The SMAC lattice planner is built upon the C++ implementation provided by the Nav2 navigation stack under ROS 2 Humble.

5.1. Evaluation of Sampling Strategy

Rejection Heuristic Bounds

Kinodynamic RRT*-based planners rely on repeated calls to a steering function in order to connect sampled states through dynamically feasible trajectories. For kinodynamic systems with velocity and acceleration limits, many randomly sampled state pairs cannot be connected within these control bounds. Attempting to solve such infeasible boundary value problems leads to wasted computation and inefficient tree expansion.

The rejection heuristic introduced in Eq. (4.1) filters candidate samples based on estimates of linear motion time and angular alignment time. In order to select practical upper bounds for these quantities, an empirical study is conducted to characterize the region of steering feasibility under idealized conditions.

Specifically, start and goal configurations are sampled uniformly from a $6\,\mathrm{m} \times 6\,\mathrm{m}$ empty workspace, ensuring that steering success or failure is governed solely by the rover's kinematic limits rather than obstacle interactions. For each sampled pair, a steering attempt is performed and the attempt is labeled as successful if a feasible trajectory is found. Subsequently, for each steering attempt, the corresponding linear motion time t_{linear} and angular alignment time t_{angular} are recorded.

The distribution in Figure 5.1 shows a clear feasibility region. Steering succeeds predominantly when both the required angular alignment time and linear motion time remain within bounded limits imposed by the rover's kinematics. Outside this feasibility region, the sampled start–goal

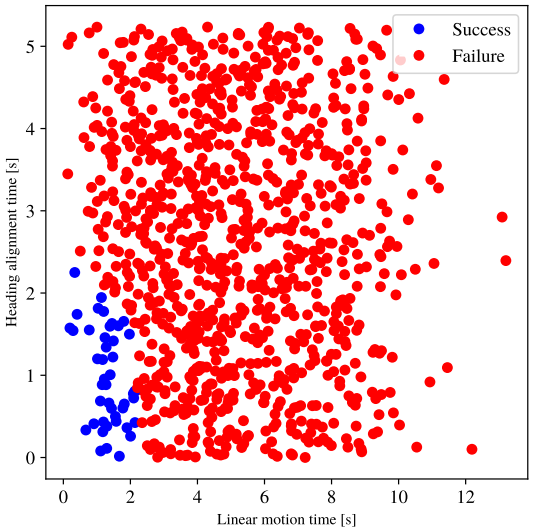


Figure 5.1.: Steering feasibility in an empty 6 m × 6 m workspace as a function of linear motion time and angular alignment time. The plot summarizes $N = 1000$ uniformly sampled start–goal configuration pairs evaluated using the same steering solver employed during planning; successful steering attempts are shown in blue, while failed attempts are shown in red.

pairs are not reachable within the imposed maximum forward velocity and steering rate limits, as the required linear displacement and/or orientation change exceeds what can be realized over the available motion horizon. Based on this empirical envelope, conservative upper bounds on t_{linear} and t_{angular} are selected for use in the rejection heuristic, ensuring that only samples with a reasonable likelihood of admitting a feasible steering solution are passed to the NLP solver.

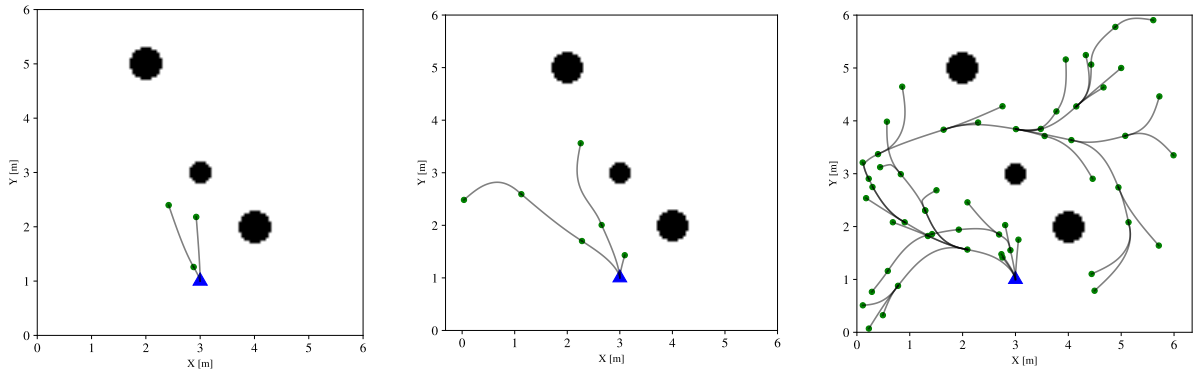
Rejection Sampling and Steering Success Rate

Building on the feasibility bounds identified above, a rejection-based sampling strategy is employed in which candidate samples are filtered prior to steering based on the heuristic defined in Eq. (4.1). This mechanism reduces the number of infeasible boundary value problems passed to the NLP solver, thereby improving the efficiency of tree expansion.

Figure 5.2 qualitatively compares RRT* tree growth under uniform sampling, potential field sampling, and rejection sampling in the same environment. While all strategies operate under identical conditions, rejection sampling produces a markedly more coherent and exploratory tree structure, indicating a higher proportion of feasible state connections.

To quantitatively assess steering feasibility, the number of successful and failed steering attempts is recorded for each sampling strategy over identical planning runs. A steering attempt is considered failed if the boundary value problem cannot be solved within the rover's kinematic constraints.

The results in Table 5.1 show that rejection sampling substantially reduces the number of failed steering attempts compared to uniform and potential field sampling. While goal-biased sampling



(a) Uniform sampling (b) Potential field sampling (c) Rejection sampling

Figure 5.2.: Qualitative comparison of RRT* tree expansion under different sampling strategies in the same environment.

Table 5.1.: Number of successful and failed steering attempts out of a total of 100 attempts for different sampling strategies.

Sampling Strategy	**Successful**	**Failed**
Uniform Sampling	3	97
Potential Field Sampling	6	94
Rejection Sampling	56	44

improves directionality, it does not explicitly enforce steering feasibility. In contrast, rejection sampling focuses computation on state pairs that are more likely to admit feasible trajectories, leading to improved tree coverage and more effective use of computational resources.

Corridor-Based Sampling and Convergence Acceleration

While rejection sampling improves the feasibility of individual steering attempts, overall convergence toward a solution can still be slow in complex environments. To further accelerate tree growth, a corridor-based sampling strategy is introduced. In this approach, samples are drawn preferentially from a bounded corridor connecting the current tree to the goal region, while still allowing limited lateral exploration.

Corridor-based sampling is evaluated in the same environment shown in Figure 5.2, with the goal located at $(5.0, 5.0)$. To obtain statistically meaningful results, each sampling scheme is evaluated over 100 independent RRT* planning runs with a maximum of 500 iterations using different random seeds. For this evaluation, only rejection sampling and rejection combined with corridor sampling are considered.

Convergence performance is measured in terms of the number of RRT* iterations required to first reach the goal region. Table 5.2 summarizes the average convergence iteration count, standard deviation, and success rate for both sampling strategies.

The results in Table 5.2 demonstrate that corridor-based sampling both accelerates convergence

Table 5.2.: Convergence performance of rejection-based sampling strategies over 100 runs. Values report the average number of iterations to first reach the goal with standard deviation rounded to the nearest integer, along with the corresponding success rate.

Sampling Strategy	**Iterations**	**Success Rate**
Rejection Sampling	293 ± 112	85 %
Rejection-Corridor Sampling	143 ± 69	99 %

and improves reliability. While rejection sampling alone reaches the goal in 85% of runs, the combined rejection–corridor strategy succeeds in 99%, nearly eliminating planning failures within the 500-iteration budget. Additionally, successful runs converge in roughly half as many iterations (143 vs. 293), confirming that focused, structured exploration outperforms broader sampling in both speed and robustness.

5.2. Experimental Setup

This section describes the experimental configuration used to evaluate the two proposed terrain-aware planning frameworks. The algorithmic parameters for each planner are detailed, the cost function weights that distinguish baseline from terrain-aware operation are specified, the navigation scenarios are defined, and the wheel-level metrics used to quantify trajectory quality are introduced.

Both planners are evaluated under two cost formulations; a baseline configuration that prioritizes geometric efficiency and control effort, and a terrain-aware configuration that additionally penalizes curvature and its rate of change on rough terrain. To isolate the effect of the cost formulation, all algorithmic parameters are held fixed between the baseline and terrain-aware variants for each planner.

Cost Function Weights

The two planners employ distinct cost formulations as detailed in Chapter 4, the RRT*+Smoother smoothing stage minimizes the objective J_{smooth} from Equation (4.2), while the SMAC lattice planner accumulates edge costs according to Eq. (4.5) with terrain-aware penalties from Eq. (4.8).

The key distinction between baseline and terrain-aware configurations lies in the activation of terrain-aware penalties. In baseline mode, the weights w_κ and $w_{\dot{\kappa}}$ are set to zero, causing both planners to optimize for path length and control smoothness without regard to terrain roughness. In terrain-aware mode, these weights are activated, penalizing high-curvature maneuvers proportionally to the local terrain roughness. The remaining weights control trajectory duration, control effort, and maneuver quality, and are held constant across both configurations to isolate the effect of terrain-aware curvature penalties. Table 5.3 summarizes the complete weight settings.

Table 5.3.: Cost function weights for baseline and terrain-aware configurations.

Planner	Weight	Baseline	Terrain-Aware
NLP Smoother	w_T (time)	1.0	1.0
	w_a (acceleration)	1.0	1.0
	w_ω (angular velocity)	1.0	1.0
	w_κ (curvature)	0.0	0.1
	$w_{\dot{\kappa}}$ (curvature rate)	0.0	1.0
SMAC Lattice Planner	w_{distance} (travel distance)	0.975	0.975
	w_{costmap} (costmap traversal)	2.0	2.0
	w_{rot} (rotation in place)	5.0	5.0
	w_{turn} (turn penalty)	1.05	1.05
	w_{change} (direction change)	0.05	0.05
	w_κ (curvature)	0.0	3.0
	$w_{\dot{\kappa}}$ (curvature rate)	0.0	1.0

5.2.1. Scenario Configuration

The evaluation scenarios are designed to test the planners under varying levels of environmental complexity while maintaining consistent baseline conditions. All scenarios share the following elements:

- **Start configuration:** The rover begins at the origin facing the positive y-axis, i.e., $(x_0, y_0, \theta_0) = (0, 0, 90°)$.
- **Rough terrain patch:** A rectangular region extending $\pm 2\,\text{m}$ in both x and y directions around the start location, simulates abrasive terrain that should discourage high-curvature maneuvers under terrain-aware planning.

The scenarios differ in goal placement and obstacle configuration. Goal positions are selected to require traversal through the rough patch, while obstacles count, size, and placement vary to create varying degrees of planning difficulty. This diversity tests the planners' ability to balance path efficiency against terrain-aware curvature minimization under different geometric constraints.

5.2.2. Evaluation Metrics

The terrain-aware cost formulation targets reduced wheel wear by discouraging high-curvature maneuvers on rough terrain, where scrubbing is most severe. Therefore, trajectory quality is evaluated at the wheel level. All metrics are computed from the discretized rover path using the geometric wheel-kinematics procedure described in Appendix B. This procedure yields, for each wheel and each path transition, a steering angle and corresponding curvature that reflect the combined effect of the rover's heading change and the wheel's steering change. All reported values correspond to path segments within the rough terrain region only. Values summarize each metric across the four wheels using the mean, and the accompanying standard deviation

indicates the metric variability across wheels.

Mean absolute wheel curvature (m^{-1}) This metric reports the average magnitude of wheel curvature along the path. The metric captures how sharply each wheel is effectively turning in the global frame, including the contribution of both rover rotation and wheel steering rotation. Lower values indicate that, on average, the wheels follow gentler arcs with less turning demand.

Cumulative wheel steering (deg) Cumulative steering measures the total amount of steering activity executed by a wheel along the path, computed as the sum of absolute changes in its steering angle between consecutive path samples. It increases when the wheel undergoes frequent steering reversals or when large steering changes occur. Lower values therefore indicate less steering actuation and smoother steering behavior.

Normalized cumulative wheel steering (deg/m) To allow a fair comparison of trajectories of different lengths, cumulative steering is normalized by the geometric path length, yielding the average steering activity per meter traveled. Lower values indicate that the rover achieves its motion with less steering effort per unit distance.

Note on Metrics Limitations These geometry-based metrics quantify steering effort as cumulative changes in steering angle along the discretized path. However, they do not account for temporal factors such as the duration of steering maneuvers. Consequently, a prolonged gentle arc may accumulate higher cumulative steering than a brief in-place rotation, even though the arc distributes actuation over a longer time period and may result in lower instantaneous mechanical stress. This limitation is particularly relevant when comparing trajectories that differ in their use of in-place rotations (see Scenario 3).

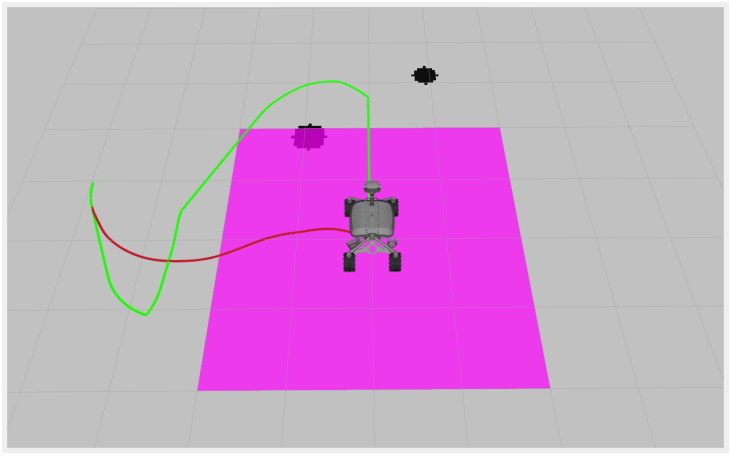
5.3. Scenario-Based Evaluation

This section presents results for each scenario, comparing baseline and terrain-aware configurations both qualitatively and quantitatively. For each scenario, the planned trajectories are visualized and the corresponding wheel-level metrics are reported.

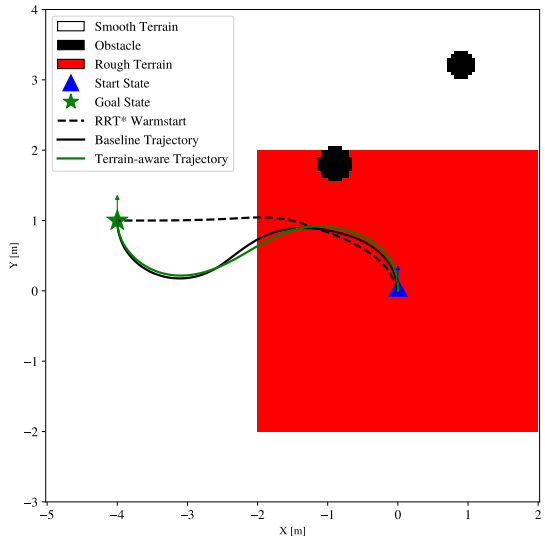
Scenario 1: Straight Exit Opportunity

Scenario 1 places the goal at $(-4.0, 1.0, 90^\circ)$ near the rough patch boundary with two obstacles constraining available paths. This configuration tests whether the terrain-aware planners can identify trajectories that minimize steering effort while traversing the rough surface.

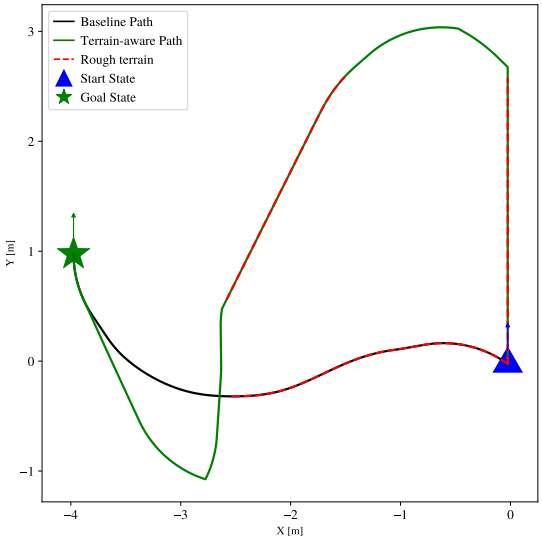
Figure 5.3 shows a clear qualitative difference in how the two planners respond to terrain-aware costs. The obstacle layout admits two viable routing options, an immediate turn within the



(a) Scenario 1 RViz environment overview. Rough terrain is indicated by the square patch in magenta, the SMAC baseline path in red, and the terrain-aware SMAC path in green.



(b) RRT*+Smoother planned trajectories overlay.



(c) SMAC paths overlay.

Figure 5.3.: Scenario 1: environment overview and planned trajectories for baseline and terrain-aware configurations for both planners.

rough patch, or a straight exit from the rough region followed by a delayed turn on smoother terrain outside. The two planners respond to this choice differently.

Both baseline and terrain-aware configurations of RRT*+Smoother in Figure 5.3b follow the same global route topology. This behavior arises because the RRT*-generated warm-start determines the global route topology, thereby constraining the smoothing stage to local refinement within that fixed structure. Within this fixed structure, terrain-aware costs act through local trajectory refinement rather than path selection. Consequently, mean absolute curvature is reduced by 84.8% as the initial turning maneuver is distributed more gradually, and cumulative steering decreases by 54.6%. The 56.2% reduction in normalized cumulative steering confirms that these

improvements correspond to genuinely smoother motion per unit distance. Notably, these gains are achieved through optimizing the underlying motion profiles to eliminate the initial spike in the rover's curvature as illustrated in Appendix C, all without any increase in path length.

In contrast, SMAC paths illustrated in Figure 5.3c exhibit explicit path switching under terrain-aware costs. The baseline solution turns immediately while still inside the rough patch, minimizing distance but executing high-curvature maneuvers on abrasive terrain. When terrain-aware costs are enabled, the planner instead selects a different path that exits the rough region along a straight segment and performs the required heading change afterwards on smooth terrain. This routing decision leads to a dramatic 91.3% reduction in mean absolute curvature, as the terrain-aware path consists almost entirely of straight-line primitives, and a 84.8% reduction in cumulative steering. These improvements come at the cost of a 128.1% increase in path length, reflecting the longer detour required to delay turning.

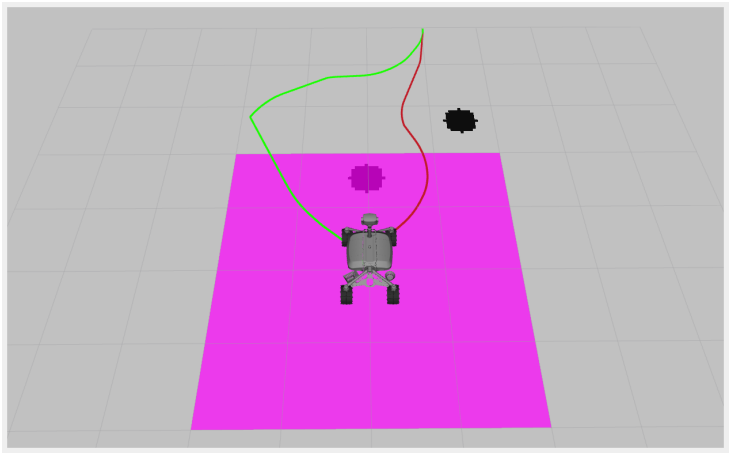
Table 5.4.: Scenario 1 RRT*+Smoother trajectory planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	5.08	5.07
Path Length in Rough Terrain (m)	2.49	2.57
Mean Abs. Curvature (m^{-1})	10.38 ± 1.66	1.58 ± 0.45
Cumulative Steering (deg)	155.98 ± 39.97	70.82 ± 24.98
Norm. Cumulative Steering (deg/m)	62.74 ± 16.08	27.51 ± 9.70

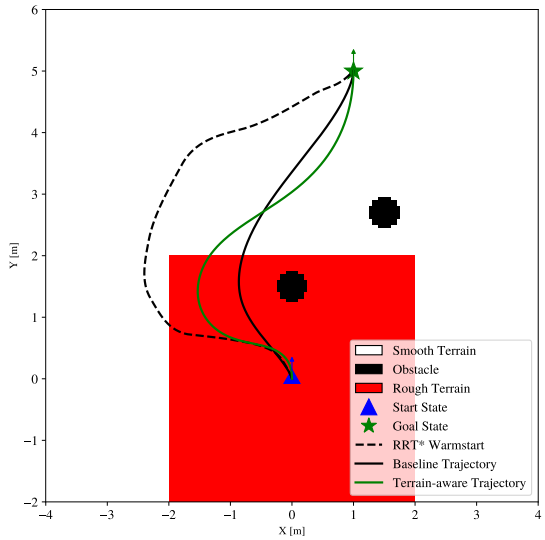
Table 5.5.: Scenario 1 SMAC lattice path planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	4.77	10.88
Path Length in Rough Terrain (m)	2.72	5.00
Mean Abs. Curvature (m^{-1})	1.83 ± 0.25	0.16 ± 0.05
Cumulative Steering (deg)	246.29 ± 18.31	37.36 ± 7.70
Norm. Cumulative Steering (deg/m)	90.67 ± 6.74	7.48 ± 1.54

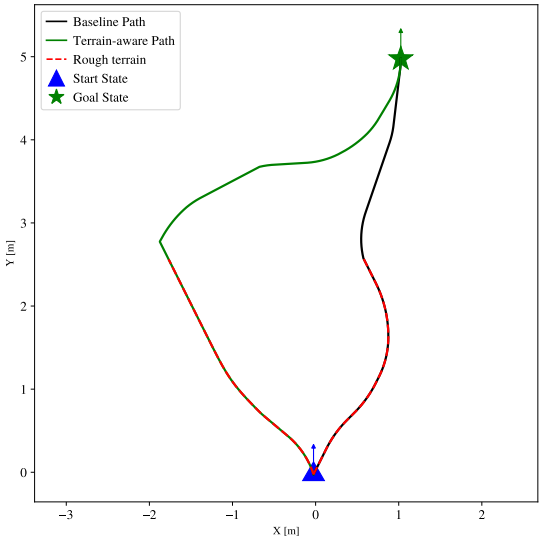
Together, these results highlight two distinct mechanisms for reducing steering effort on rough terrain. RRT*+Smoother achieves curvature reduction through fine-grained trajectory refinement within a fixed route dictated by the warm-start, yielding substantial steering improvements with minimal path length increase. SMAC, by contrast, leverages its discrete action space to select a fundamentally different route composed of low-curvature motion primitives, accepting significant path extension to achieve near-minimal curvature. Both approaches successfully reduce steering effort, but through different trade-offs between global route selection and local trajectory optimization.



(a) Scenario 2 RViz environment overview. Rough terrain is indicated by the square patch in magenta, the SMAC baseline path in red, and the terrain-aware SMAC path in green.



(b) RRT*+Smoother planned trajectories overlay.



(c) SMAC paths overlay.

Figure 5.4.: Scenario 2: environment overview and planned trajectories for baseline and terrain-aware configurations for both planners.

Scenario 2: S-Curve

Scenario 2 sets the goal at $(1.0, 5.0, 90°)$ and introduces two circular obstacles at $(0.0, 1.5)$ and $(1.5, 2.7)$ with radius 0.25 m. Their placement creates a characteristic S-shaped corridor in which a near-direct route to the goal is feasible but requires alternating left–right steering maneuvers to pass between obstacles. An alternative route also exists that follows a longer, single smooth arc around the obstacle pair.

The numerical results in Tables 5.6 and 5.7 indicate consistent improvements under terrain-aware cost formulations, though with smaller magnitude than observed in Scenario 1.

As observed in Scenario 1, the RRT* warm-start commits both RRT*+Smoother configurations to the same route along the left side obstacle pair as seen in Figure 5.4b, limiting terrain-aware costs to local refinement. However, the terrain-aware configuration still reduces mean absolute curvature by 71.0% and normalized cumulative steering by 37.3%. The smoother distributes heading changes more gradually, reducing steering effort per unit distance, with a 19.5% increase in path length.

SMAC exhibits different behavior, as shown in Figure 5.4a. In both configurations, the rover first performs an in-place rotation to align with the available passage before advancing. Following this initial reorientation, the baseline solution proceeds directly between the obstacles using an S-shaped trajectory with alternating steering directions. This choice minimizes travel distance but introduces repeated curvature sign changes while traversing the rough region, resulting in high steering effort. When terrain-aware costs are enabled, the planner instead favors a wider and straighter path around the obstacle pair after the initial rotation. This avoids steering reversals and shifts the majority of turning outside the rough patch, at the expense of a longer overall path. Quantitatively, this behavior yields a 42.1% reduction in mean absolute curvature and a 43.5% reduction in normalized cumulative steering, accompanied by a 36.0% increase in path length.

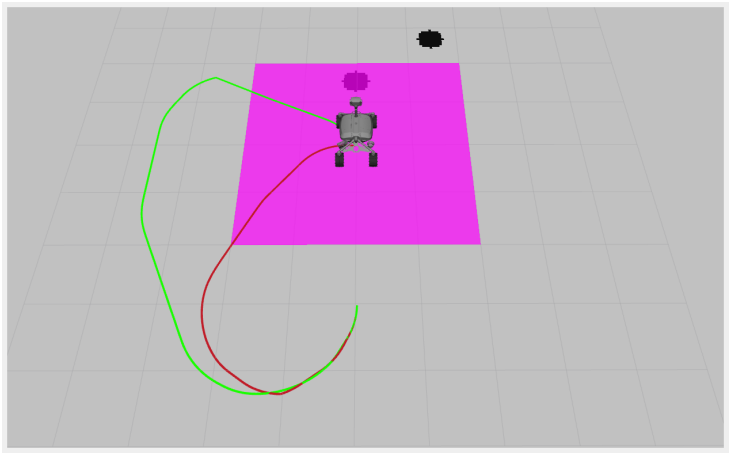
Table 5.6.: Scenario 2 RRT*+Smoother trajectory planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	5.79	6.92
Path Length in Rough Terrain (m)	2.28	2.95
Mean Abs. Curvature (m^{-1})	7.29 ± 0.77	2.11 ± 0.43
Cumulative Steering (deg)	180.58 ± 24.37	146.16 ± 25.60
Norm. Cumulative Steering (deg/m)	79.19 ± 10.69	49.62 ± 8.69

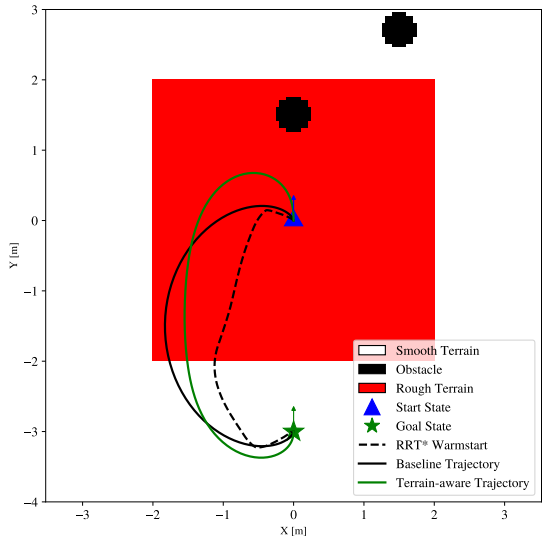
Table 5.7.: Scenario 2 SMAC lattice path planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	5.39	7.33
Path Length in Rough Terrain (m)	2.97	3.21
Mean Abs. Curvature (m^{-1})	2.14 ± 0.10	1.24 ± 0.15
Cumulative Steering (deg)	351.56 ± 12.40	213.99 ± 13.56
Norm. Cumulative Steering (deg/m)	118.21 ± 4.17	66.73 ± 4.23

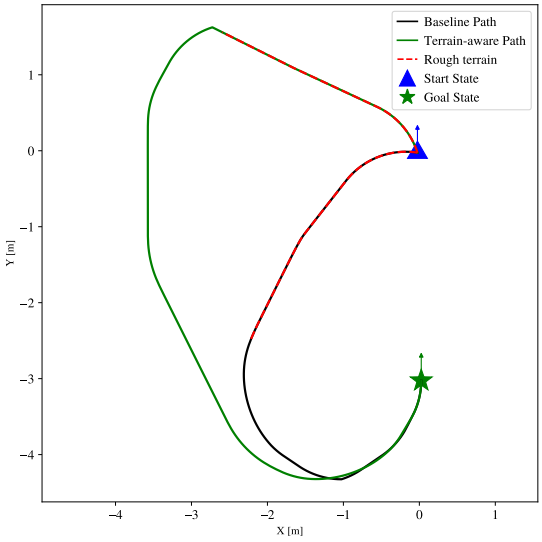
Overall, Scenario 2 highlights how terrain-aware costs influence planners with different decision mechanisms. RRT*+Smoother refines an existing trajectory to reduce curvature without altering its overall topology, whereas SMAC leverages its discrete action space to replace oscillatory steering with smoother motion when such alternatives are available.



(a) Scenario 3 RViz environment overview. Rough terrain is indicated by the square patch in magenta, the SMAC baseline path in red, and the terrain-aware SMAC path in green.



(b) RRT*+Smoother planned trajectories overlay.



(c) SMAC paths overlay.

Figure 5.5.: Scenario 3: environment overview and planned trajectories for baseline and terrain-aware configurations for both planners.

Scenario 3: Goal Behind the Rover

Scenario 3 places the goal behind the rover at $(0.0, -3.0, 90°)$, requiring a large reorientation relative to the initial heading to exit rough terrain. Two obstacles are positioned ahead of the start pose blocking the trivial straight exit of the rough patch. The scenario therefore highlights the trade-off between turning immediately on rough terrain versus delaying the heading change.

Figure 5.5 illustrates how the two planners resolve the required heading reversal under different cost formulations. While both must ultimately achieve a reorientation to move towards the goal, they differ in how this reorientation is realized along the path, ranging from concentrated turning

to more distributed, gradual curvature.

As shown in Figure 5.5b, the RRT*+Smoother baseline trajectory executes the heading reversal via a tight initial turn, introducing high curvature within the rough terrain. After reversing direction, the trajectory continues through the rough region along a smooth arc. In contrast, the terrain-aware solution spreads the initial reorientation over a longer distance, yielding a more gradual heading change, and then traverses the rough patch along a nearly straight segment.

Table 5.8 quantifies this behavior. Executing the required reorientation over a longer portion of the trajectory increases the total path length by 13.4%. At the same time, turning is less concentrated, reducing mean absolute curvature by 51.7% and normalized cumulative steering by 47.3%.

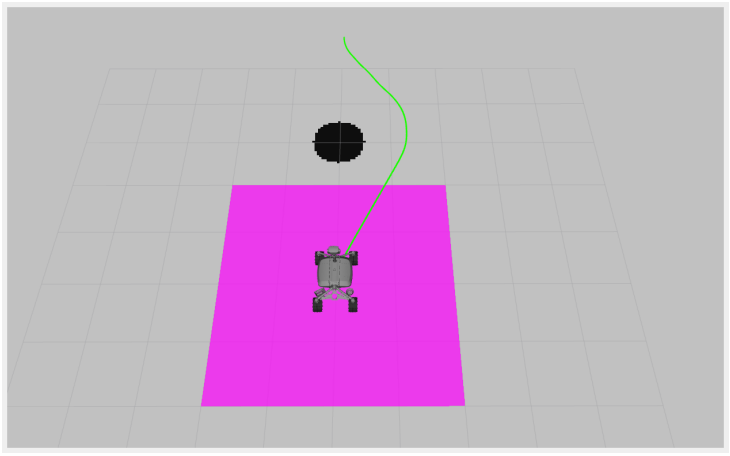
SMAC exhibits a comparatively subtle qualitative change under terrain-aware costs, as shown in Figure 5.5c. In both configurations, the rover first performs an in-place reorientation at the start to achieve a feasible exit direction, and then follows an almost straight path to exit the rough patch. While the terrain-aware path visually shows a shorter initial in-place rotation, this geometric change is only weakly captured by the steering metric, which sums discrete angle changes without accounting for rotation duration or velocity. As a consequence, improvements in curvature and steering effort remain modest; mean absolute curvature decreases by 15.6% and normalized cumulative steering decreases by 6.4%, despite a substantial 60.4% increase in total path length, as reported in Table 5.9.

Table 5.8.: Scenario 3 RRT*+Smoother trajectory planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

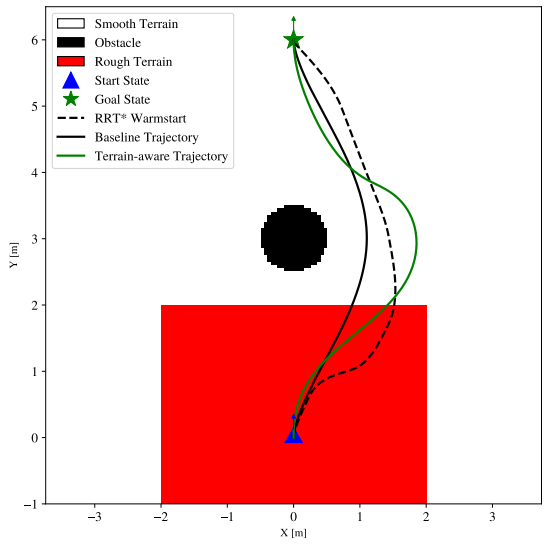
Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	5.88	6.67
Path Length in Rough Terrain (m)	3.48	4.16
Mean Abs. Curvature (m^{-1})	4.58 ± 0.69	2.21 ± 0.58
Cumulative Steering (deg)	133.48 ± 32.98	84.06 ± 32.39
Norm. Cumulative Steering (deg/m)	38.34 ± 9.47	20.22 ± 7.79

Table 5.9.: Scenario 3 SMAC lattice path planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

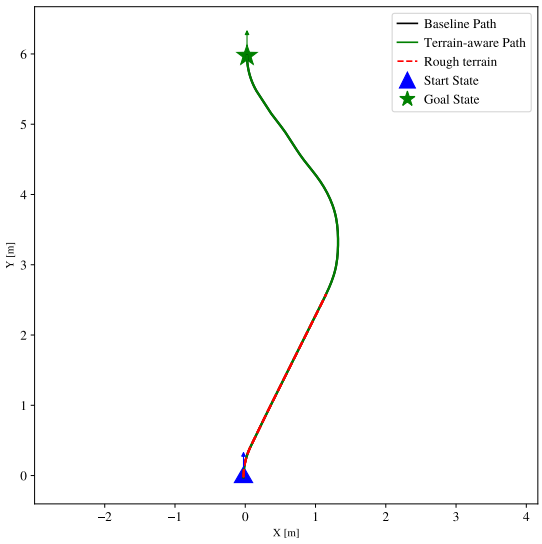
Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	7.83	12.56
Path Length in Rough Terrain (m)	3.58	3.14
Mean Abs. Curvature (m^{-1})	1.28 ± 0.36	1.08 ± 0.27
Cumulative Steering (deg)	206.82 ± 46.39	169.70 ± 25.99
Norm. Cumulative Steering (deg/m)	57.76 ± 12.96	54.05 ± 8.28



(a) Scenario 4 RViz environment overview. Rough terrain is indicated by the square patch in magenta, the SMAC baseline path in red, and the terrain-aware SMAC path in green.



(b) RRT*+Smoother planned trajectories overlay.



(c) SMAC paths overlay.

Figure 5.6.: Scenario 4: environment overview and planned trajectories for baseline and terrain-aware configurations for both planners.

Scenario 4: Minimal Obstacles

Scenario 4 presents a simpler environment with one obstacle and a goal behind it at $(0.0, 6.0, 90°)$. This scenario highlights a case where the effect of terrain-aware costs is primarily reflected in the optimization-based smoothing rather than the SMAC planner.

For RRT*+Smoother, the baseline solution in Figure 5.6b begins with a high-curvature in-place rotation maneuver, after which the trajectory exits the rough patch along an almost straight segment. Under terrain-aware costs, the optimizer alters the early maneuver, instead of concentrating heading change at the start, it produces a smooth departing arc that transitions

out of the rough region without the initial in-place rotation. The corresponding velocity profiles are shown in Appendix C.

The quantitative results in Table 5.10 strongly reflect this behavior. The mean absolute curvature decreased by 92.7%, and normalized cumulative steering experienced a decrease of 75.1%. These improvements come with a 14.4% increase in total path length from 6.45 m to 7.38 m.

On the other hand, SMAC produced exactly identical baseline and terrain-aware trajectories in Figure 5.6c. This behavior is reflected in Table 5.11, where all reported metrics match across configurations. In this scenario, the planned path is already dominated by straight motion primitives, apart from the initial in-place rotation. As a result, enabling terrain-aware costs does not lead to an alternative primitive sequence that yields measurable improvements in the wheel-level curvature or steering metrics.

Table 5.10.: Scenario 4 RRT*+Smoother trajectory planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	6.45	7.38
Path Length in Rough Terrain (m)	2.20	2.56
Mean Abs. Curvature (m^{-1})	10.86 ± 1.51	0.80 ± 0.12
Cumulative Steering (deg)	150.94 ± 34.25	43.79 ± 6.33
Norm. Cumulative Steering (deg/m)	68.66 ± 15.58	17.12 ± 2.47

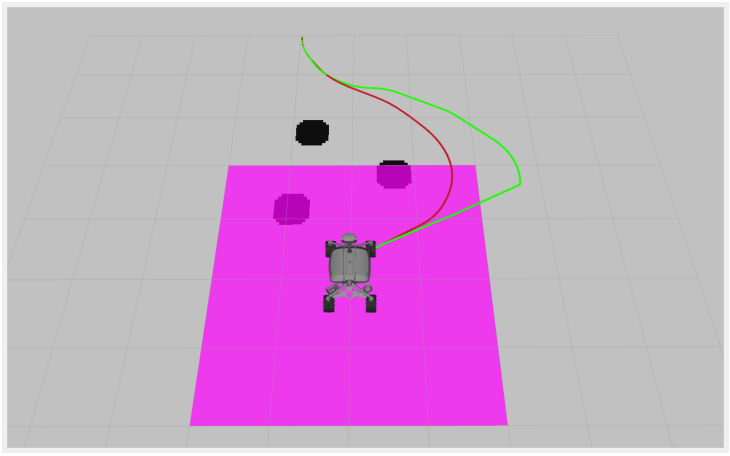
Table 5.11.: Scenario 4 SMAC lattice path planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	6.66	6.66
Path Length in Rough Terrain (m)	2.92	2.92
Mean Abs. Curvature (m^{-1})	0.54 ± 0.15	0.54 ± 0.15
Cumulative Steering (deg)	66.83 ± 12.73	66.83 ± 12.73
Norm. Cumulative Steering (deg/m)	22.86 ± 4.36	22.86 ± 4.36

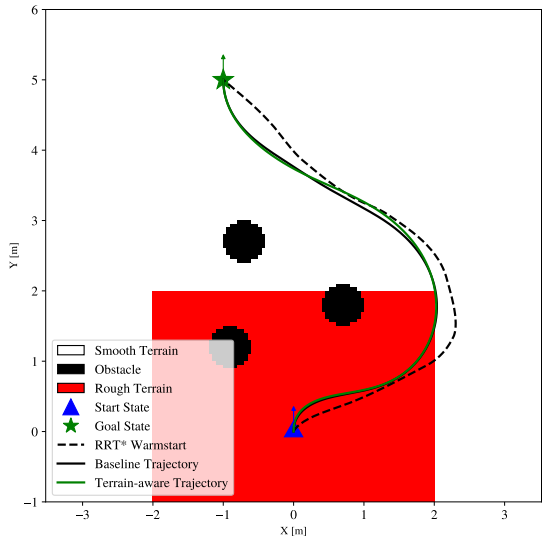
Overall, Scenario 4 illustrates that when SMAC's baseline solution already yields a low curvature path, terrain-aware costs may have no effect, whereas the optimization-based refinement stage can still exploit additional degrees of freedom to remove localized high-curvature maneuvers and substantially reduce wheel-level metrics.

Scenario 5: Constrained Environment

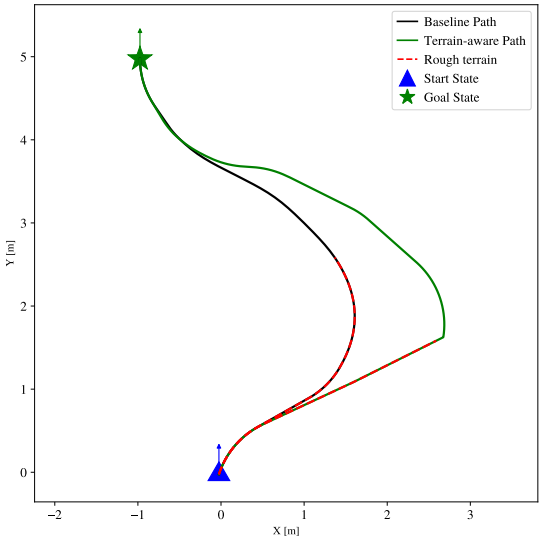
Scenario 5 places three obstacles directly in front of the rover, forcing an immediate maneuver to exit the rough area before progressing toward the goal at $(-1.0, 5.0, 90°)$. This obstacle arrangement limits the optimal solution classes and therefore tests both planners in a constrained environment.



(a) Scenario 5 RViz environment overview. Rough terrain is indicated by the square patch in magenta, the SMAC baseline path in red, and the terrain-aware SMAC path in green.



(b) RRT*+Smoother planned trajectories overlay.



(c) SMAC paths overlay.

Figure 5.7.: Scenario 5: environment overview and planned trajectories for baseline and terrain-aware configurations for both planners.

The RRT*+Smoother solutions in Figure 5.7b are almost identical at the path level. Both configurations execute very similar maneuvers to navigate around the three obstacles. The key difference is not the geometric path, but the steering behavior at the beginning of the maneuver. In the baseline configuration, the trajectory exhibits a pronounced initial curvature peak, whereas the terrain-aware configuration reduces this initial peak, leading to a smoother curvature evolution. This is reflected in Table 5.12, where the mean absolute curvature decreases from $8.13\,\mathrm{m}^{-1}$ to $4.18\,\mathrm{m}^{-1}$. However, this curvature reduction does not translate into improved steering metrics in this scenario. Cumulative steering increases from 195.52° to 232.12°, and normalized cumulative steering increases from $72.64\,\mathrm{deg/m}$ to $84.33\,\mathrm{deg/m}$.

Figure 5.7c shows that both SMAC baseline and terrain-aware solutions begin with the same in-place rotation followed by a right-turning arc to clear the obstacle cluster directly in front of the rover. After this, the paths differ in how they maneuver around the obstacles. The baseline follows a smooth left-turning arc around the obstacles, whereas the terrain-aware plan prefers to exit along a straighter segment. Qualitatively, this choice reduces the curvature change and steering transition needed to switch from a right-turning arc to a left-turning arc. This behavior is illustrated in Table 5.13 by a 25.2% reduction in mean absolute curvature from $1.43\,\mathrm{m}^{-1}$ to $1.07\,\mathrm{m}^{-1}$ and a reduction in cumulative steering from 189.41° to 169.70°. However, the normalized cumulative steering changes only by 3.1% from 54.94 deg/m to 53.24 deg/m at the expense of 25.5% increase in total path length. Overall, Scenario 5 illustrates a case where terrain-aware costs alter the primitive sequence to reduce wheel curvature and cumulative steering, while the distance-normalized steering metric shows only a small net change.

Table 5.12.: Scenario 5 RRT*+Smoother trajectory planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	7.65	7.69
Path Length in Rough Terrain (m)	2.69	2.75
Mean Abs. Curvature (m^{-1})	8.13 ± 1.48	4.18 ± 1.88
Cumulative Steering (deg)	195.52 ± 44.42	232.12 ± 56.39
Norm. Cumulative Steering (deg/m)	72.64 ± 16.50	84.33 ± 20.49

Table 5.13.: Scenario 5 SMAC lattice path planner results. Wheel-level metrics reported as mean ± std across 4 wheels.

Metric	**Baseline**	**Terrain-Aware**
Path Length (m)	6.87	8.62
Path Length in Rough Terrain (m)	3.45	3.19
Mean Abs. Curvature (m^{-1})	1.43 ± 0.09	1.07 ± 0.27
Cumulative Steering (deg)	189.41 ± 14.89	169.70 ± 25.99
Norm. Cumulative Steering (deg/m)	54.94 ± 4.32	53.24 ± 8.16

5.4. Randomized Evaluation

To assess whether the terrain-aware objective generalizes beyond the fixed, hand-crafted scenarios, both planners are evaluated on 100 randomly generated environments. The randomized evaluation tests the planners against diverse obstacle configurations and goal placements, and tests whether the smoothness gains reported in Section 5.3 generalize.

Randomized Scenario Generation

Each randomized environment shares the same start configuration as the hand-crafted scenarios. The rover begins at the origin with heading 90°, positioned at the center of a 4×4 m rough terrain patch. Goal positions are sampled uniformly at distances between 4 and 6 m from the start, with random heading orientations. Between 2 and 5 circular obstacles populate each environment, with radii sampled uniformly from 0.1 to 0.5 m. To avoid collisions at the start or goal positions that would prevent planning, obstacles are constrained to maintain a minimum clearance from both the start and goal positions equal to the sum of the robot footprint radius and the obstacle radius. Additionally, obstacles are not permitted to overlap with one another. This procedure yields environments that vary in difficulty while reducing trivial infeasibility, such that failures primarily reflect kinodynamic constraints rather than immediate collisions at the start or goal.

Reporting Conventions

Baseline and terrain-aware variants of both planners are evaluated and two complementary comparisons are reported.

First, a within-planner evaluation isolates the effect of enabling the terrain-aware objective while holding the planning paradigm fixed. For each planner, performance differences are reported using the difference in metrics computed per scenario as

$$\Delta = m_{\text{baseline}} - m_{\text{terrain-aware}}, \tag{5.1}$$

where m denotes the value of a given metric for that scenario. With this sign convention, $\Delta > 0$ indicates that the terrain-aware configuration achieved a lower metric value than baseline. For curvature and steering metrics, lower values represent improvements. For path length, $\Delta < 0$ (negative values) indicate that terrain-aware paths are longer, which is the expected trade-off for achieving smoother motion. Results are reported only on scenarios where both the baseline and terrain-aware configurations succeed. To capture both central tendency and distribution shape, mean ± standard deviation together with the median are reported.

Second, a cross-planner evaluation compares the terrain-aware configurations of both planners. This comparison clarifies whether one planner tends to achieve smoother or shorter solutions when both optimize for steering effort on rough terrain. In this case, the metrics differences are defined as

$$\Delta = m_{\text{Smoother(terrain-aware)}} - m_{\text{SMAC(terrain-aware)}}, \tag{5.2}$$

and are computed only on scenarios where both terrain-aware planners succeed. With this convention, $\Delta > 0$ indicates that SMAC achieves a lower value than RRT*+Smoother.

In this cross-planner comparison, the two planners return outputs with different internal parameterizations. SMAC produces a discrete path defined by motion primitives, whereas RRT*+Smoother produces a time-parameterized trajectory whose spatial sampling density depends on the optimized motion profiles. To ensure a fair geometric comparison, the RRT*+Smoother

trajectory is treated as a purely geometric path and resampled to the same spatial step size used for the SMAC path before computing the metrics. Consequently, the cross-planner results compare the two terrain-aware solutions at the level of path geometry and metrics differences reflect spatial curvature and steering variation along the traversed path.

Finally, boxplots are included to visualize dispersion and outliers, since the reported metrics differences can vary substantially across randomized environments and the mean±standard deviation alone may not fully reflect skewness or occasional extreme cases.

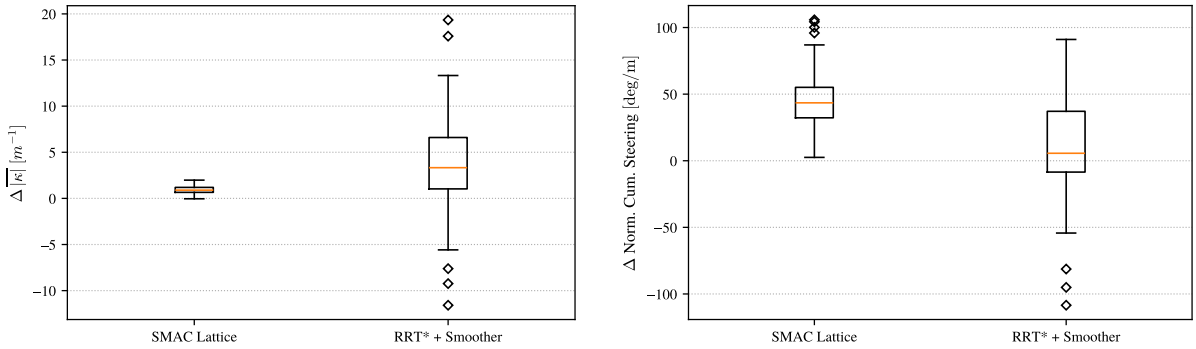
Within-Planner Comparison

Table 5.14 reports the within-planner differences between baseline and terrain-aware configurations on the common subset of scenarios where both succeed. Figure 5.8 complements these summaries by visualizing the corresponding distributions.

Table 5.14.: Within-planner results on scenarios where both configurations succeed. Scenario-wise metrics differences are summarized as mean±std and median. Positive Δ indicates an improvement over the baseline for the corresponding metric under terrain-aware costs.

Metric	**RRT*+Smoother**		**SMAC Lattice**	
	baseline	**terrain-aware**	**baseline**	**terrain-aware**
Success rate (%)	71.0	58.0	100.0	100.0
	mean±std	**median**	**mean±std**	**median**
Δ Path length (m)	-0.3 ± 0.7	-0.1	-3.2 ± 2.1	-3.4
Δ Mean Abs. Curvature (m^{-1})	3.9 ± 5.9	3.3	0.9 ± 0.4	0.9
Δ Norm. Cum. steering (deg/m)	10.0 ± 36.6	5.6	44.7 ± 21.2	43.0

For RRT*+Smoother, the scenario-wise differences in Table 5.14 are computed on the 58 scenarios where both the baseline and terrain-aware configurations succeed. On this common subset, terrain-aware costs typically reduce turning effort in rough terrain as mean absolute curvature is positive with a mean of $3.9\,\mathrm{m}^{-1}$ and median $3.3\,\mathrm{m}^{-1}$, and Δ normalized cumulative steering is also positive with mean of 10.0 deg/m and median 5.6 deg/m. The medians being positive indicate that more than half of the scenarios in the common success subset improve over the baseline under terrain-aware costs. However, the mean exceeding the median for both metrics suggests a right-skewed distribution in which a subset of the scenarios achieve particularly large improvements, inflating the mean relative to the typical case. Additionally, the metrics standard deviations exceed their corresponding means, indicating substantial variability across scenarios and implying that the average improvement is not uniformly realized as some scenarios show only small gains or even degradations, consistent with the negative outliers shown in Figure 5.8. Path-length differences are small in magnitude and centered close to zero, with a mean difference of $-0.3 \pm 0.7\,\mathrm{m}$ and a median of $-0.1\,\mathrm{m}$ indicating that the observed curvature and steering reductions come at the cost of an average increase in path length of 0.3 m. Finally,



(a) Distribution of differences in mean absolute curvature.

(b) Distribution of differences in normalized cumulative steering.

Figure 5.8.: Within-planner distributions of metrics differences, reported for scenarios where both configurations succeed. Positive differences indicate an improvement over the baseline for the corresponding metric under terrain-aware costs.

the terrain-aware configuration exhibits a lower success rate, indicating that introducing the terrain-aware objective can alter the convergence behavior of the smoothing stage, such that some scenarios solved under baseline do not yield a successful plan under the terrain-aware configuration.

For SMAC, terrain-aware costs yield consistent improvements in the steering-related metrics while maintaining a 100% success rate for both configurations. Table 5.14 shows that the difference in mean absolute curvature is positive, with a mean of $0.9 \pm 0.4\,\mathrm{m}^{-1}$ and a median of $0.9\,\mathrm{m}^{-1}$, and that the difference in normalized cumulative steering is also positive, with a mean of $44.7 \pm 21.2\,\mathrm{deg/m}$ and a median of $43.0\,\mathrm{deg/m}$. The mean values are close to the medians, suggesting that the distributions are not strongly skewed and that the improvements are broadly realized across scenarios rather than driven by a small number of extreme cases. Consistent with this, the SMAC boxplots in Figure 5.8 show a clear positive shift with comparatively compact dispersion for both metrics relative to RRT*+Smoother. While the main trade-off is path length, the difference is negative with a mean of $-3.2 \pm 2.1\,\mathrm{m}$ and a median of $-3.4\,\mathrm{m}$, indicating that the terrain-aware configuration of SMAC planner tends to substantially increase total distance traveled.

Figure 5.8 complements the table summaries by showing the full spread of scenario-wise differences. For RRT*+Smoother, both boxplots exhibit a wide dispersion and include negative outliers, indicating that terrain-aware costs do not improve every scenario and can degrade the metrics. For SMAC, the distributions are shifted more uniformly above zero with comparatively tighter dispersion, consistent with the close agreement between mean and median in Table 5.14.

Cross-planner Comparison

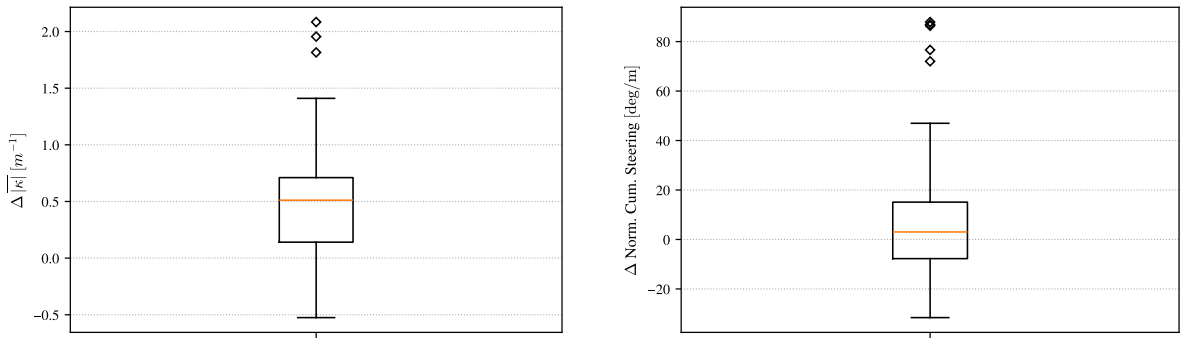
Table 5.15 reports cross-planner differences between the two terrain-aware configurations on the common successful subset of 58 scenarios where both planners return a valid solution. With

the sign convention adopted here, a positive Δ indicates that SMAC achieved a lower (better) value than RRT*+Smoother for the corresponding metric, whereas a negative Δ indicates that RRT*+Smoother performs better. Figure 5.9 complements these summaries by visualizing the full distributions of scenario-wise cross-planner differences.

Table 5.15.: Cross-planner results on scenarios where terrain-aware configurations succeed for both planners. Scenario-wise metric differences are summarized as mean±std and median across scenarios. Positive Δ indicates that SMAC shows improvement over RRT*+Smoother for the corresponding metric.

Metric	**mean±std**	**median**
Δ Path length (m)	-3.0 ± 2.8	-3.4
Δ Mean Abs. Curvature (m^{-1})	0.5 ± 0.5	0.5
Δ Norm. Cum. steering (deg/m)	8.5 ± 27.8	3.0

Table 5.15 indicates a clear trade-off between geometric efficiency and wheel-level smoothness under terrain-aware planning. The mean for Δ path length is negative (mean -3.0 ± 2.8 m; median -3.4 m), showing that RRT*+Smoother typically produces shorter terrain-aware solutions than SMAC. In contrast, the mean value for both $\Delta|\kappa|$ (mean $0.5 \pm 0.5\,\mathrm{m}^{-1}$; median $0.5\,\mathrm{m}^{-1}$) and Δ normalized cumulative steering (mean 8.5 ± 27.8 deg/m; median 3.0 deg/m) is positive, indicating that SMAC more often achieves lower curvature and lower distance-normalized steering effort than RRT*+Smoother. Moreover, the large standard deviation for the normalized cumulative steering difference highlights substantial scenario-to-scenario variability.



(a) Distribution of differences in mean absolute curvature.

(b) Distribution of differences in normalized cumulative steering.

Figure 5.9.: Cross-planner distributions of metrics differences under terrain-aware costs, reported for scenarios where both terrain-aware planners succeed. Positive differences indicate that SMAC performs better than RRT*+Smoother for the corresponding metric.

Figure 5.9 complements these summaries by visualizing the scenario-wise distributions and revealing an asymmetry in both metrics. For mean absolute curvature, the distribution is shifted to the positive side, consistent with SMAC achieving lower curvature in a majority of scenarios, but it also extends into negative values, indicating that there remains a subset of scenarios in which RRT*+Smoother is better. Importantly, the positive side of the distribution extends farther than

the negative side, suggesting that when SMAC improves over RRT*+Smoother the magnitude of the improvement is typically larger than the magnitude of the cases where RRT*+Smoother performs better. The same qualitative pattern appears for normalized cumulative steering; although both planners can be better depending on the scenario, the distribution exhibits broader spread and a stronger positive tail, implying that SMAC's steering-effort advantage, when it occurs, can be comparatively pronounced.

Edge Cases

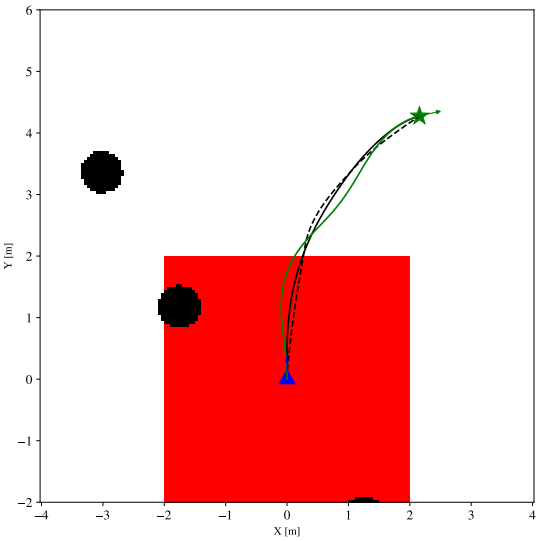
While the randomized evaluation demonstrates that terrain-aware costs often reduce wheel-level steering effort on average, a closer inspection reveals a number of edge cases in which RRT*+Smoother does not benefit from the terrain-aware formulation. Figure 5.10 highlights four representative examples drawn from the randomized set, all corresponding to RRT*+Smoother.

In the first two cases (Figures 5.10a and 5.10b), the baseline solution already exhibits favorable curvature properties when exiting the rough terrain. In these scenarios, the terrain-aware objective does not uncover a fundamentally better maneuver and instead perturbs the trajectory by introducing additional turning, either through an unnecessary curvature change along an otherwise smooth arc or via an in-place rotation within the rough patch. These examples illustrate that when the baseline trajectory is already close to optimal with respect to steering effort, the terrain-aware objective may offer limited room for improvement and can even degrade the solution.

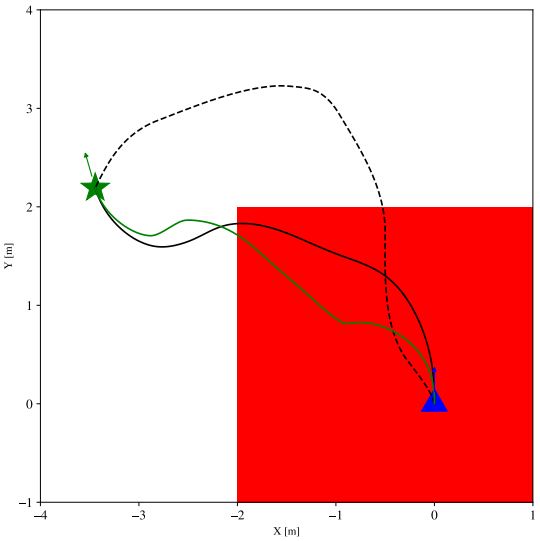
The third case (Figure 5.10c) highlights a limitation of the geometry-based steering metrics used in this evaluation. Here, the terrain-aware solution removes an initial in-place rotation by transitioning into a long, continuous arc, which is visually smoother and avoids concentrated steering at near-zero velocity. However, because the metrics accumulate steering changes along the entire path and do not account for the duration or velocity of the maneuver, the prolonged arc results in higher cumulative steering values than the baseline, which performs a short in-place turn followed by straight motion. This case highlights that the current metrics capture geometric steering effort but do not fully reflect temporal aspects of actuation or time under load.

Finally, the fourth case (Figure 5.10d) represents a scenario in which both configurations converge to nearly identical trajectories in rough terrain. In such cases, the terrain-aware formulation neither improves nor degrades performance.

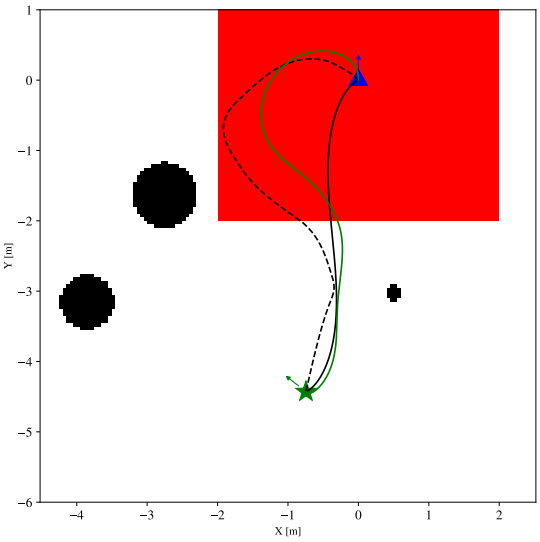
Overall, these edge cases provide useful context for interpreting the aggregate results. They show that the degradations observed for RRT*+Smoother arise from scenarios where the baseline solution is already well-conditioned or where the optimization landscape offers limited alternatives within the fixed route topology.



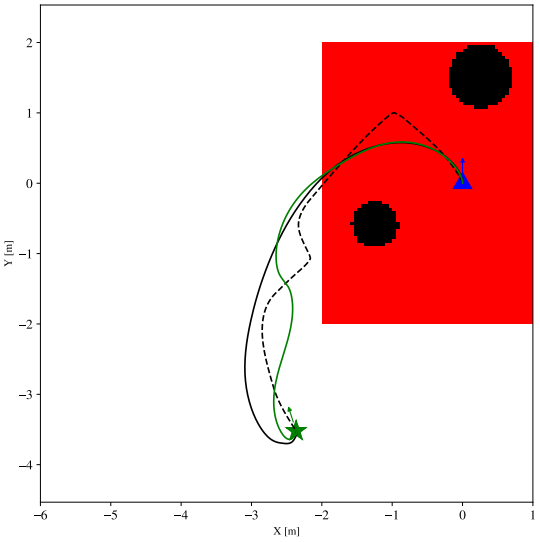
(a) Edge case 1: Baseline exits the rough terrain along an already smooth arc, while the terrain-aware solution introduces additional unnecessary turning.



(b) Edge case 2: Terrain-aware optimization introduces an unnecessary in-place rotation inside the rough terrain, whereas the baseline exits along a smooth arc without concentrated steering.



(c) Edge case 3: Terrain-aware solution avoids an initial in-place turn but replaces it with a prolonged arc, resulting in higher accumulated steering despite a visually smoother motion.



(d) Edge case 4: Baseline and terrain-aware solutions are nearly identical.

Figure 5.10.: Representative edge cases for RRT*+Smoother, illustrating scenarios in which terrain-aware costs provide limited benefit or lead to worse wheel-level metrics.

5.5. Runtime Analysis

Runtime is an important practical consideration for deployment, but the absolute planning times reported here should not be interpreted as a benchmark between the two methods.

RRT*+Smoother is implemented as research-oriented Python code, whereas SMAC lattice is highly optimized C++ code and reflects a production-grade software stack. As a result, the primary purpose of this section is not to claim direct computational superiority, but to characterize the runtime behavior of each planning paradigm—in particular, the stochastic variability of RRT*+Smoother versus the more predictable runtime of the deterministic lattice-based search.

Table 5.16.: Runtime statistics over the successful runs of the 100 randomized environments.

Planner	**Mean ± Std (s)**	**Min (s)**	**Max (s)**
RRT*+Smoother	26.92 ± 28.77	3.78	81.26
SMAC Lattice	0.30 ± 0.16	0.20	1.01

Table 5.16 summarizes planning-time statistics over the successful runs in the 100 randomized environments. RRT*+Smoother exhibits large run-to-run variability, with a mean of 26.92 s and a standard deviation of 28.77 s, together with a wide spread between its minimum and maximum observed runtimes (3.78 s to 81.26 s). This spread is consistent with a sampling-based planner whose exploration effort and downstream smoothing difficulty depend strongly on the particular obstacle geometry and the random sampling process, leading to highly variable convergence time across environments.

In contrast, SMAC shows a much tighter runtime range, with a mean of 0.30 s and a standard deviation of 0.16 s, and a comparatively narrow min–max interval (0.20 s to 1.01 s). This concentration reflects the deterministic structure of graph search on a fixed motion-primitive lattice, for which runtime tends to be more predictable from environment complexity. In practical terms, these results suggest that the SMAC runtime is more reliable to provision for, whereas RRT*+Smoother can exhibit substantial runtime fluctuations across scenarios, even when both ultimately return feasible solutions.

5.6. Summary

This chapter evaluated the terrain-aware motion planning framework by validating the sampling strategy for kinodynamic RRT*, comparing planner behaviors in hand-crafted scenarios, assessing generalization through randomized evaluation with both within-planner (baseline vs. terrain-aware) and cross-planner (terrain-aware vs. terrain-aware) comparisons, and characterizing runtime performance.

For kinodynamic RRT*, the empirical steering-feasibility study enabled practical rejection bounds that increased the steering success rate from 3% (uniform sampling) and 6% (potential field sampling) to 56%. Adding corridor-based sampling further improved convergence reliability, raising the probability of reaching the goal within 500 iterations from 85% to 99%, which confirms that structured sampling can both accelerate exploration and reduce wasted infeasible steering attempts.

Across the five hand-crafted scenarios, terrain-aware costs generally reduced wheel-level curvature and steering effort in rough terrain, but the underlying mechanism depended on the planner. RRT*+Smoother typically preserved the route topology induced by the RRT* warm-start and achieved improvements through local refinement of maneuver geometry and motion profiles, often by mitigating short-duration curvature spikes. In contrast, SMAC lattice could change primitive sequences and route topology to defer turning until leaving the rough patch, which produced large reductions in curvature and steering but required substantial path-length increases when low-curvature detours were available. The scenarios also exposed important limitations. In Scenario 4, SMAC produced identical baseline and terrain-aware solutions when the baseline already consisted of predominantly straight primitives. In Scenario 5, the terrain-aware configuration of RRT*+Smoother reduced mean curvature but increased the steering-based metrics, illustrating that its terrain-aware formulation can sometimes degrade performance.

The randomized evaluation over 100 environments reinforces and contextualizes the trends observed in the scenario-based analysis. Within-planner comparisons (Table 5.14 and Figure 5.8) show that SMAC lattice maintains a 100% success rate under both baseline and terrain-aware configurations and exhibits consistent improvements when terrain-aware costs are enabled, as reflected by positive median reductions in both mean absolute curvature and normalized cumulative steering, at the expense of increased path length. In contrast, RRT*+Smoother shows lower and less consistent reliability, with success rates dropping from 71% under the baseline objective to 58% when terrain-aware costs are activated. This reduction stems from the curvature terms in the cost formulation making the smoothing NLP more numerically fragile at low speeds, where curvature-dependent terms can become poorly conditioned. On the subset of scenarios where both baseline and terrain-aware RRT*+Smoother configurations succeed, terrain-aware optimization generally reduces curvature and normalized steering; however, the corresponding distributions exhibit substantially larger dispersion and include negative outliers, indicating that these improvements are not uniformly realized across environments.

Cross-planner comparison is therefore performed only on the 58 scenarios where both terrain-aware planners successfully converge (Table 5.15 and Figure 5.9). On this common subset, RRT*+Smoother typically produces shorter paths on average, while the SMAC lattice planner consistently achieves lower wheel-level curvature and lower normalized cumulative steering. Together, these results highlight a consistent trade-off; RRT*+Smoother exhibits higher variability and reduced robustness under terrain-aware costs, whereas SMAC delivers more reliable and uniform reductions in steering-related metrics at the cost of longer paths.

Finally, the runtime analysis highlighted a practical contrast between planning paradigms. SMAC exhibited low and predictable planning times, while RRT*+Smoother showed substantially higher and more variable runtimes, reflecting the stochastic nature of sampling-based exploration and environment-dependent convergence difficulty.

Overall, terrain-aware costs reduce wheel-level turning demand in many environments, but the magnitude, reliability, and associated path-length trade-offs depend strongly on whether the planner can alter route topology (SMAC) or is limited to local refinement around a warm-start

solution (RRT*+Smoother).

6. Conclusion

This chapter summarizes the contributions of this thesis, discusses the limitations of the proposed approaches, and outlines directions for future research.

6.1. Summary of Contributions

This thesis investigated motion planning strategies for planetary rovers that reduce steering-induced wheel wear by incorporating terrain-aware cost formulations into the planning process. The work addressed a gap in existing approaches, which optimize for geometric feasibility, collision avoidance, and energy efficiency but do not account for the mechanical consequences of steering maneuvers on abrasive terrain.

The primary contributions are summarized as follows:

Curvature-Based Terrain-Aware Cost Model. A cost formulation was developed that penalizes high-curvature maneuvers in proportion to local terrain roughness. Path curvature and its rate of change serve as computationally efficient proxies for steering effort, capturing the kinematic relationship between rover body motion and wheel-level steering demands. This formulation shifts wear mitigation from the execution layer—where operational strategies such as modified driving primitives have previously been applied—to the deliberative planning layer, enabling the planner to proactively avoid wear-inducing maneuvers.

Sampling-Based Kinodynamic Planner with Terrain-Aware Smoothing. A two-stage planning framework was developed combining kinodynamic RRT* with nonlinear trajectory optimization. A hybrid rejection-corridor sampling strategy was introduced that improves steering feasibility from 3% (uniform sampling) to 56% (rejection sampling) and accelerates goal convergence from 85% to 99% success rate. The smoothing stage incorporates terrain-aware curvature penalties enabling fine-grained control over steering smoothness.

Extension of Lattice-Based Planner with Terrain-Aware Edge Costs. The Nav2 SMAC lattice planner was extended with terrain-dependent edge costs that penalize high-curvature motion primitives on rough terrain. The edge cost formulation integrates curvature and curvature rate penalties weighted by terrain roughness, enabling the graph search to favor primitive sequences that defer turning until leaving rough terrain regions.

Experimental Evaluation. Both planners were evaluated using the Lightweight Rover Unit (LRU) kinematic model across five hand-crafted scenarios and 100 randomly generated environments. The evaluation demonstrated that terrain-aware costs generally reduce wheel-level curvature and cumulative steering effort in rough terrain, though the mechanism and consistency differ by planning paradigm. The lattice-based planner achieved consistent improvements across most scenarios by selecting alternative motion primitive sequences, at the cost of increased path length. The sampling-based planner achieved curvature reductions with minimal path extension but exhibited higher variability due to its reliance on local refinement within a fixed route topology.

6.2. Limitations

Several limitations of the proposed approaches should be acknowledged:

Geometric Proxy for Wear. The curvature-based cost model provides a computationally efficient proxy for steering effort but does not capture the full complexity of wheel–terrain interaction. Factors such as wheel sinkage, soil cohesion, contact pressure distribution, and wheel slip are not modeled. While the proxy captures the kinematic component of steering demand, it may not accurately predict wear rates across all terrain types and soil conditions.

Binary Terrain Representation. The roughness map uses a binary classification (rough or smooth), which simplifies the terrain model but does not capture gradations in surface abrasiveness. Real planetary terrain exhibits continuous variation in roughness, and a multi-level or continuous roughness representation could enable more realistic modeling.

Evaluation Metrics. The evaluation metrics—mean absolute curvature, cumulative steering, and normalized cumulative steering—capture geometric steering effort but do not account for temporal aspects such as time under load or velocity during steering. A trajectory with concentrated high-curvature maneuvers executed quickly may impose different mechanical stresses than a longer trajectory with distributed steering maneuvers, but both would yield similar metric values.

Simulation-Only Validation. All experiments were conducted in simulation without physical hardware validation. While the kinematic model is based on the LRU rover platform, real-world factors such as wheel slip, sensor noise, localization error, and terrain estimation uncertainty were not considered.

Computational Cost of Sampling-Based Planner. The RRT*+Smoother planner exhibited mean runtime of 26.92 seconds with high variability (standard deviation 28.77 seconds), which may preclude real-time onboard deployment. The comparison with SMAC with mean of 0.30 seconds is confounded by implementation differences (Python versus C++), but the inherent computational complexity of nonlinear trajectory optimization remains a practical limitation.

Reduced Reliability Under Terrain-Aware Costs. The terrain-aware configuration of RRT*+Smoother exhibited reduced success rates compared to the baseline (58% versus 71%), attributed to numerical conditioning issues in the smoothing NLP when curvature-dependent terms are activated at low speeds. This trade-off between solution quality and reliability needs further investigation.

6.3. Future Work

Several directions for future research emerge from this work, with three primary areas warranting particular attention:

Hardware Validation on LRU. All experiments in this thesis were conducted in simulation. Validation on the physical Lightweight Rover Unit (LRU) platform would assess the practical effectiveness of terrain-aware planning under real-world conditions. Hardware experiments could quantify the actual reduction in steering actuator loads and wheel forces achieved by terrain-aware trajectories compared to baseline paths. Long-term trials on analog planetary terrain with instrumented wheels could provide direct measurement of wear rates, enabling calibration of the curvature-based cost model and validation of the geometric proxy against physical degradation. Such experiments would also reveal the impact of factors not captured in simulation, including wheel slip, terrain deformation, and localization uncertainty.

Integration with Perception-Based Traversability Assessment. The terrain representations used in this work—binary roughness maps and obstacle grids—were manually constructed for experimental evaluation. A natural extension is integration with a perception pipeline that estimates terrain properties from onboard sensors. Visual texture analysis, elevation mapping from stereo or LiDAR data, and proprioceptive feedback from wheel encoders and inertial measurements could provide continuous roughness estimates in real time. Such integration would enable the terrain-aware cost formulation to operate on learned or estimated terrain properties rather than handcrafted maps, making the approach applicable to autonomous navigation in unknown environments. The cost formulation developed in this thesis is agnostic to the source of terrain information and could directly incorporate roughness values from a traversability analysis module.

Improved Evaluation Metrics. The evaluation metrics employed—mean absolute curvature, cumulative steering, and normalized cumulative steering—capture geometric steering effort but have notable limitations. They do not account for temporal aspects such as time under load or velocity during steering. Future work should develop metrics that incorporate time-weighted steering effort, integrate velocity profiles, or directly estimate mechanical stress from kinematic and dynamic quantities. Correlation studies between geometric metrics and measured actuator loads or wear rates from hardware experiments would inform the design of more physically meaningful evaluation criteria.

Additional Directions. Beyond these primary areas, several other extensions are worth consideration. Extending the binary roughness representation to a continuous or multi-level model would enable more realistic terrain characterization that reflects gradations in surface abrasiveness. Addressing the numerical conditioning issues that reduce RRT*+Smoother reliability under terrain-aware costs—through reformulated cost terms, regularization, or alternative solvers—would improve the robustness of the sampling-based approach. Finally, formulating the planning problem as a multi-objective optimization trading off path length, energy consumption, and steering wear could provide mission planners with a range of solutions suited to different operational priorities.

6.4. Closing Remarks

This thesis demonstrated that terrain-aware steering penalties can meaningfully influence rover trajectory generation without requiring high-fidelity terramechanics models. By encoding steering effort as a curvature-based cost and integrating this cost into both sampling-based and search-based planning frameworks, the work provides a practical approach for embedding wear-conscious objectives into the deliberative planning layer. While limitations remain—particularly in the fidelity of the wear proxy and the computational cost of optimization-based planning—the results establish that proactive avoidance of wear-inducing maneuvers is achievable within existing motion planning paradigms. As planetary exploration missions extend to increasingly challenging terrain with longer operational lifetimes, such wear-aware planning strategies may contribute to preserving rover mobility and extending mission success.

A. Motion Primitives Construction Algorithms

A.1. Minimal Primitives Set Generation Algorithm

Algorithm 2 Minimal Primitives Set Generation

Input: Grid resolution Δ_{xy}, min. turning radius $R_{\min}$, heading count n_θ, stopping threshold N_{stop}
Output: Motion primitive set $\mathcal{M}$
$\mathcal{M} \leftarrow \emptyset$
$L_{\min} \leftarrow R_{\min} \cdot \min_m |\theta_{m+1} - \theta_m|$ ▷ Minimum arc length
$w_{\text{start}} \leftarrow \lceil L_{\min}/\Delta_{xy} \rceil$ ▷ Starting wavefront index
for all $\theta_{\text{start}} \in \{\theta_m : 0 \le \theta_m \le \pi/2\}$ **do** ▷ First quadrant only
    $\mathcal{E} \leftarrow \emptyset$
    $w \leftarrow w_{\text{start}}$; $n_{\text{empty}} \leftarrow 0$
    while $n_{\text{empty}} < N_{\text{stop}}$ **do**
        found $\leftarrow$ **false**
        for all $(x, y) \in$ WavefrontPoints(w, Δ_{xy}) **do**
            for all θ_{end} such that $|\theta_{\text{end}} - \theta_{\text{start}}| \le \pi/2$ **do**
                $\mathcal{P} \leftarrow$ GeneratePath$((x, y), \theta_{\text{start}}, \theta_{\text{end}}, R_{\min})$
                if $\mathcal{P} \neq \emptyset$ **and** IsMinimal$(\mathcal{P}, \mathcal{E})$ **then**
                    Add $\mathcal{P}$ to $\mathcal{M}[\theta_{\text{start}}]$
                    Insert endpoint of $\mathcal{P}$ into $\mathcal{E}$
                    found $\leftarrow$ **true**
                end if
            end for
        end for
        $n_{\text{empty}} \leftarrow$ **if** found **then** 0 **else** $n_{\text{empty}} + 1$
        $w \leftarrow w + 1$
    end while
end for
$\mathcal{M} \leftarrow$ ReflectAcrossAxes$(\mathcal{M})$ ▷ Exploit symmetry
$\mathcal{M} \leftarrow$ AddInPlaceRotations$(\mathcal{M})$ ▷ Add rotation in place primitives
return $\mathcal{M}$

A.2. Geometric Primitive Construction Algorithm

Algorithm 3 GeneratePath

Input: Endpoint $(x_{\text{end}}, y_{\text{end}})$, start angle θ_{start}, end angle θ_{end}, minimum turning radius $R_{\min}$
Output: Primitive path $\mathcal{P}$ or $\emptyset$
Construct directed line ℓ_1 through $(0, 0)$ with heading θ_{start}
Construct directed line ℓ_2 through $(x_{\text{end}}, y_{\text{end}})$ with heading θ_{end}
if $\ell_1 \parallel \ell_2$ **then**
    if $\ell_1 \equiv \ell_2$ **then**
        return straight-line path ▷ Case 1: Collinear lines
    else
        return $\emptyset$ ▷ Case 2: Offset parallel lines
    end if
end if
$I \leftarrow \ell_1 \cap \ell_2$ ▷ Intersection point
if I does not lie in the forward direction of ℓ_1 and backward direction of ℓ_2 **then**
    return $\emptyset$ ▷ Case 3: Backward intersection
end if
$d_1 \leftarrow \|I - (0, 0)\|, \quad d_2 \leftarrow \|I - (x_{\text{end}}, y_{\text{end}})\|$
$\phi \leftarrow \pi - |\theta_{\text{end}} - \theta_{\text{start}}|$ ▷ Interior angle between ℓ_1 and ℓ_2
$d_{\min} \leftarrow R_{\min} / \tan(\phi/2)$ ▷ Minimum tangent length for valid turning radius
if $d_1 < d_{\min}$ **or** $d_2 < d_{\min}$ **then**
    return $\emptyset$ ▷ Case 4: Sharp turn, insufficient tangent length
end if
$d \leftarrow \min(d_1, d_2)$
Determine tangent points at distance d from I along each line
Compute circle center C as the intersection of the perpendiculars to ℓ_1 and ℓ_2 at the respective tangent points
$R \leftarrow \|C - \text{tangent point}\|$ ▷ Resulting arc radius
if $R < R_{\min}$ **then**
    return $\emptyset$
end if
Construct path: $(0, 0) \rightarrow$ tangent $\rightarrow$ arc around $C \rightarrow$ tangent $\rightarrow (x_{\text{end}}, y_{\text{end}})$
return primitive path $\mathcal{P}$

B. Calculation of Wheel Kinematics and Evaluation Metrics from Path Geometry

This appendix describes how wheel steering angles, wheel curvatures, and the reported evaluation metrics are computed directly from a discrete planned path. The procedure is purely geometric and treats the planned motion as a sequence of poses $\mathbf{q}_i = [x_i,\ y_i,\ \theta_i]^T$ sampled along the trajectory. All metrics in this work are computed over the portion of the path that lies in rough terrain.

B.1. Wheel Kinematics

Let the path be a sequence of n poses $\{\mathbf{q}_i\}_{i=0}^{n-1}$ with

$$\mathbf{q}_i = \begin{bmatrix} x_i & y_i & \theta_i \end{bmatrix}^T.$$

Each wheel w has a fixed location in the rover body frame given by

$$\mathbf{p}_w = \begin{bmatrix} x_w \\ y_w \end{bmatrix},$$

where x_w points forward and y_w points left.

For each transition $i \to i+1$, the path increment is

$$ds_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \qquad d\theta_i = \theta_{i+1} - \theta_i.$$

A wheel displacement vector is then computed from the rigid-body geometric increment as

$$\Delta\mathbf{p}_{w,i} = \begin{bmatrix} \Delta x_{w,i} \\ \Delta y_{w,i} \end{bmatrix} = \begin{bmatrix} ds_i - d\theta_i\, y_w \\ d\theta_i\, x_w \end{bmatrix},$$

with magnitude

$$ds_{w,i} = \|\Delta\mathbf{p}_{w,i}\|_2 = \sqrt{(ds_i - d\theta_i\, y_w)^2 + (d\theta_i\, x_w)^2}.$$

The wheel steering angle is taken as the direction of the wheel displacement,

$$\varphi_{w,i} = \operatorname{atan2}(\Delta y_{w,i},\ \Delta x_{w,i}).$$

Steering change between consecutive transitions is computed as the signed angular difference between $\varphi_{w,i}$ and $\varphi_{w,i-1}$, and the magnitude of steering effort is measured using the absolute value of that difference.

To handle transitions near standstill, a wheel displacement threshold ε is used. If the wheel displacement satisfies $ds_{w,i} < \varepsilon$, the wheel is considered stationary and its steering angle remains unchanged at $\varphi_{w,i} = \varphi_{w,i-1}$, while the wheel's curvature for this transition is set to zero.

For non-standstill transitions, wheel curvature is computed as the geometric curvature that accounts for both the rover's body rotation and the wheel's steering rotation,

$$\kappa_{w,i} = \frac{d\theta_i + d\varphi_{w,i}}{ds_{w,i}},$$

where $d\varphi_{w,i}$ denotes the signed wheel steering change between transitions. This definition captures how the wheel's heading evolves due to the combination of chassis yaw change and steering actuation, normalized by the wheel's traveled distance.

B.2. Evaluation Metrics

All metrics below are computed from the sequence of wheel steering angles and wheel curvatures over the part of the path that lies in rough terrain. Path length is reported as the traveled distance along the path segment under evaluation,

$$L = \sum_{i=0}^{n-2} ds_i.$$

Mean absolute wheel curvature. For each wheel, the mean absolute curvature is computed as the average of $|\kappa_{w,i}|$ over the evaluated path segment and is reported in m^{-1}. This value reflects typical smoothness of the wheel's motion.

Cumulative steering. For each wheel, cumulative steering is computed as the sum of absolute steering changes along the evaluated path segment,

$$S_w = \sum_i |d\varphi_{w,i}|,$$

and is reported in degrees. This quantity measures how much the wheel steering angle varies along the path, independent of time parameterization.

Normalized cumulative steering. To provide a fair comparison across paths of different lengths, cumulative steering is normalized by the path length,

$$S_w^{\text{norm}} = \frac{S_w}{L},$$

and is reported in degrees per meter. This value measures steering variation per unit distance.

Aggregation across wheels. Wheel-level metrics are aggregated across the four wheels by reporting the mean and standard deviation across wheels.

C. Supplementary Velocity Profiles

This appendix shows the optimized body velocity profiles for each benchmark scenario in the scenario-based evaluation Section 5.3 and the edge cases in Section 5.4 for the RRT*+Smoother planner. For each scenario, the rover's angular velocity $w(t)$, linear velocity $v(t)$, and curvature $\kappa(t)$ are reported for both the Baseline and the Terrain-Aware formulations. The rough-terrain intervals are highlighted in red.

Scenario 1

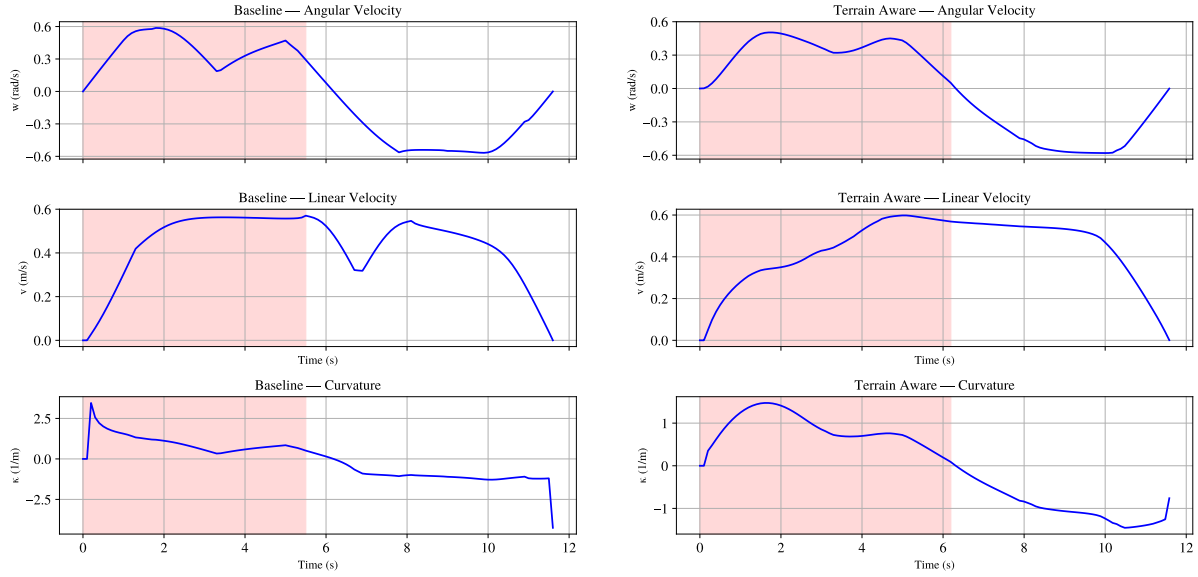


Figure C.1.: Velocity profiles for Scenario 1. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Scenario 2

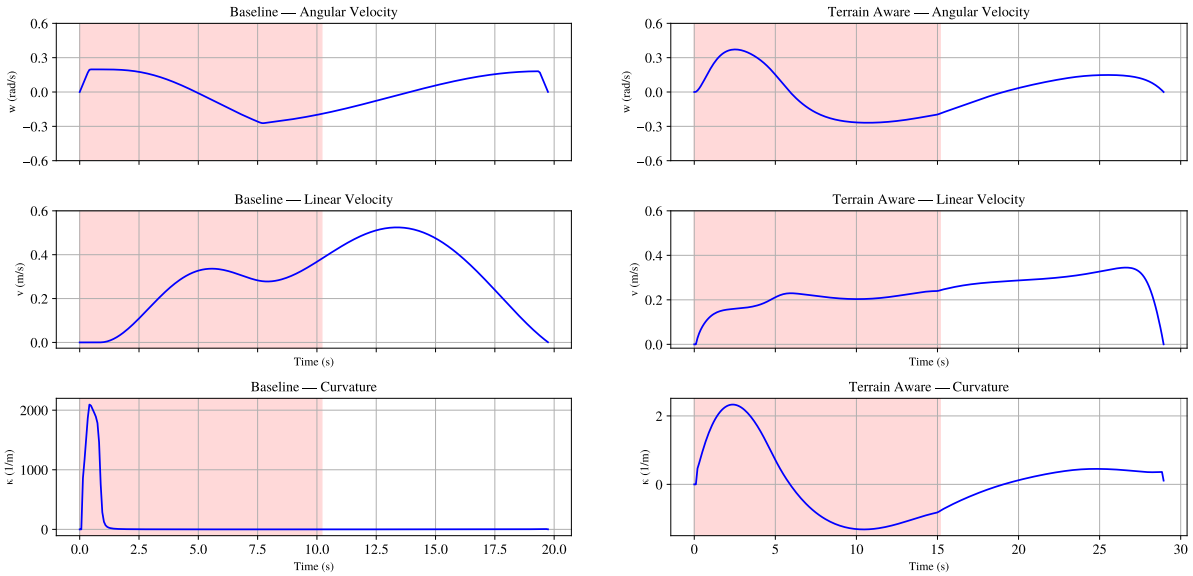


Figure C.2.: Velocity profiles for Scenario 2. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Scenario 3

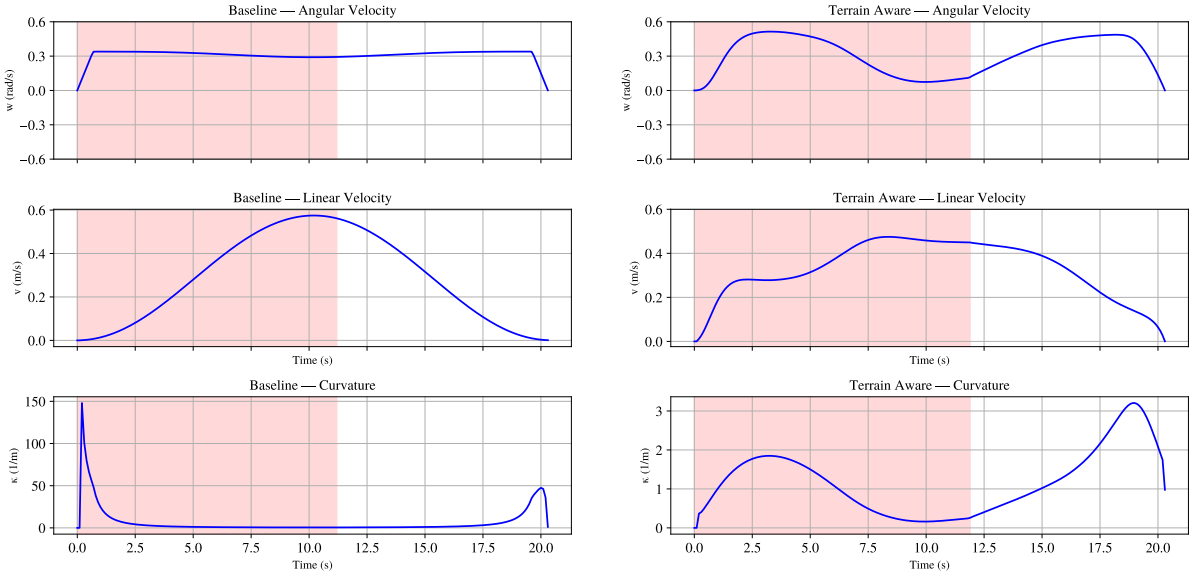


Figure C.3.: Velocity profiles for Scenario 3. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Scenario 4

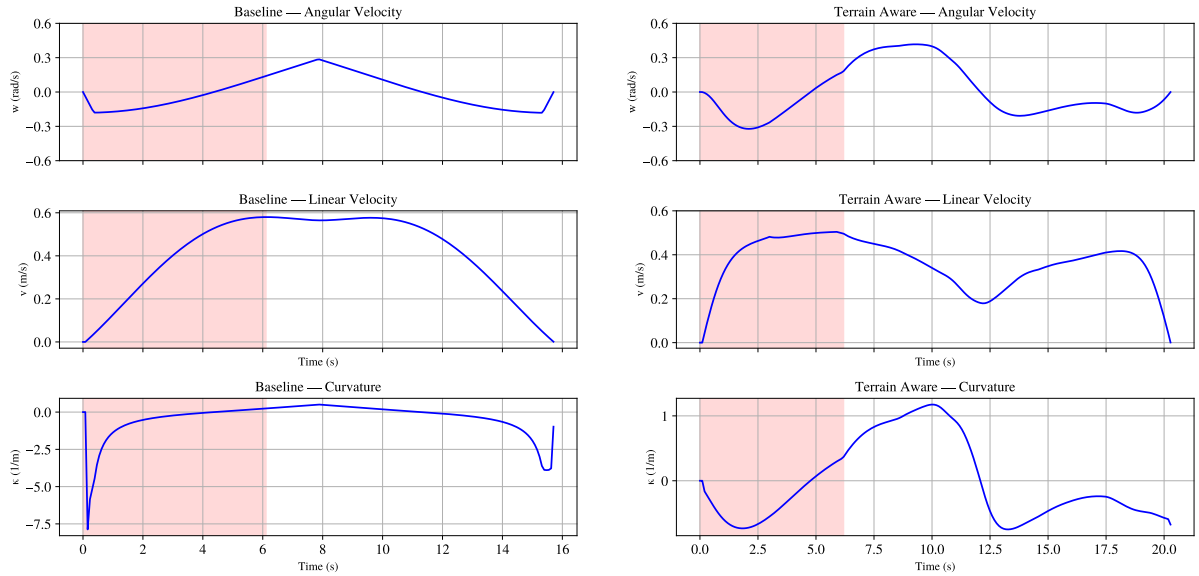


Figure C.4.: Velocity profiles for Scenario 4. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Scenario 5

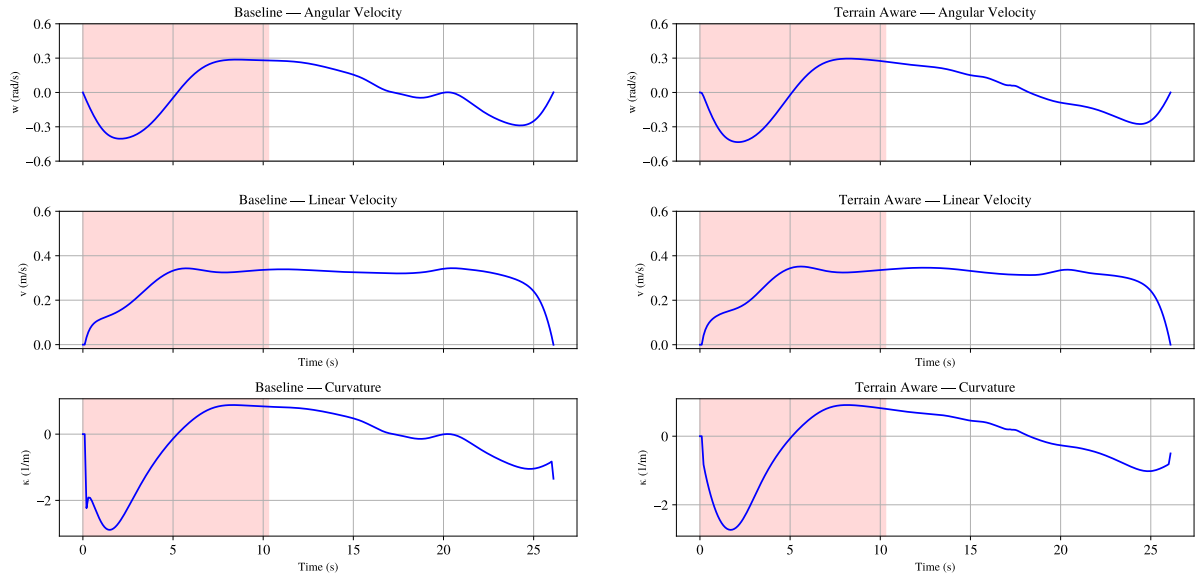


Figure C.5.: Velocity profiles for Scenario 5. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Edge Case 1

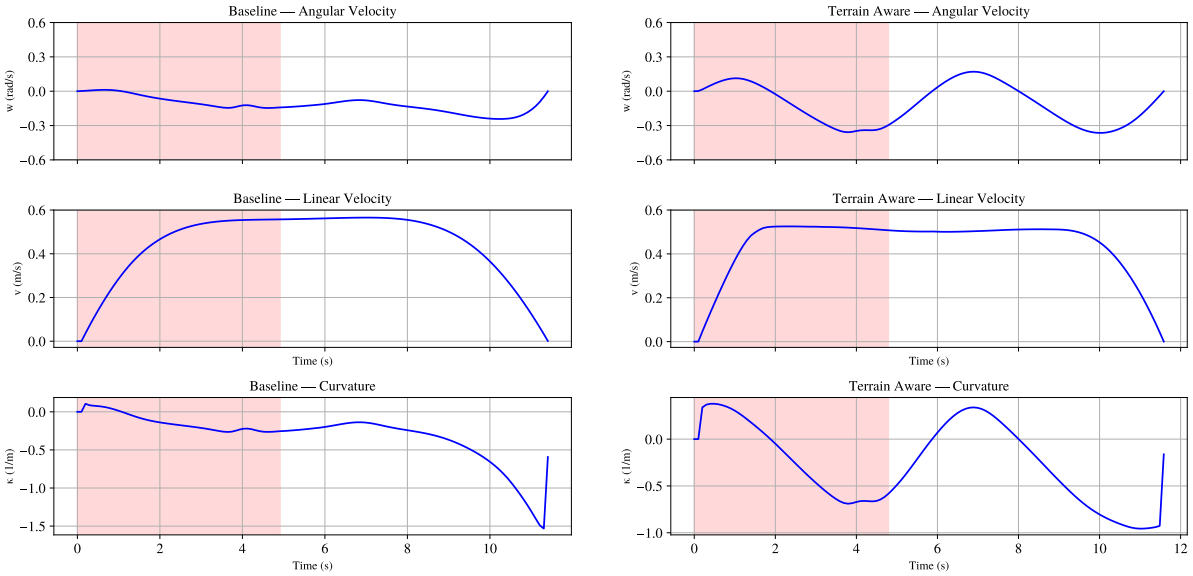


Figure C.6.: Velocity profiles for Edge Case 1. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Edge Case 2

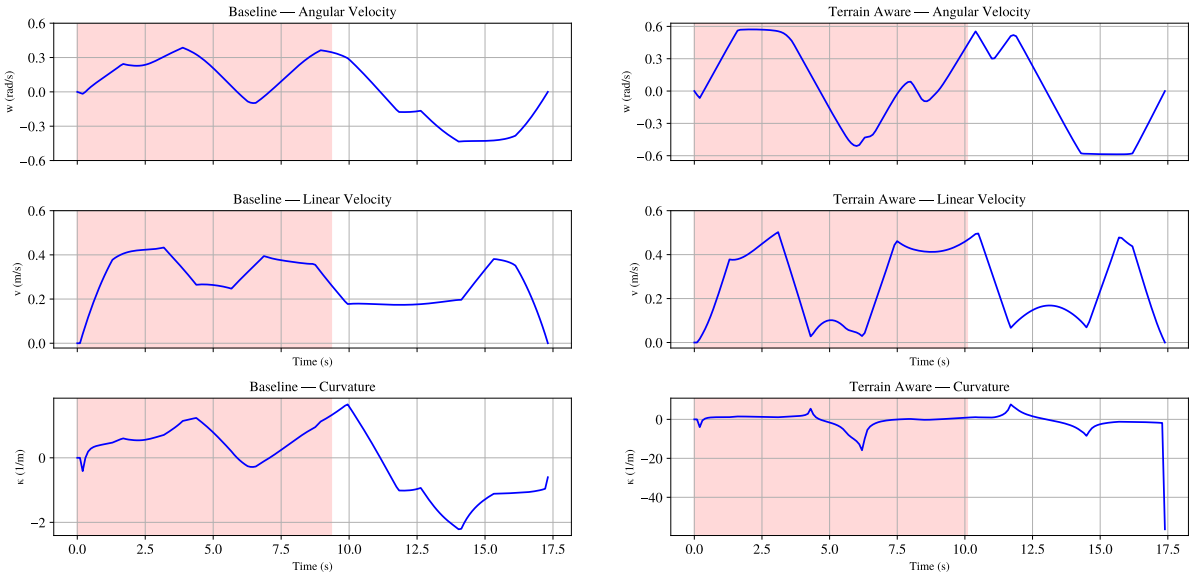


Figure C.7.: Velocity profiles for Edge Case 2. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Edge Case 3

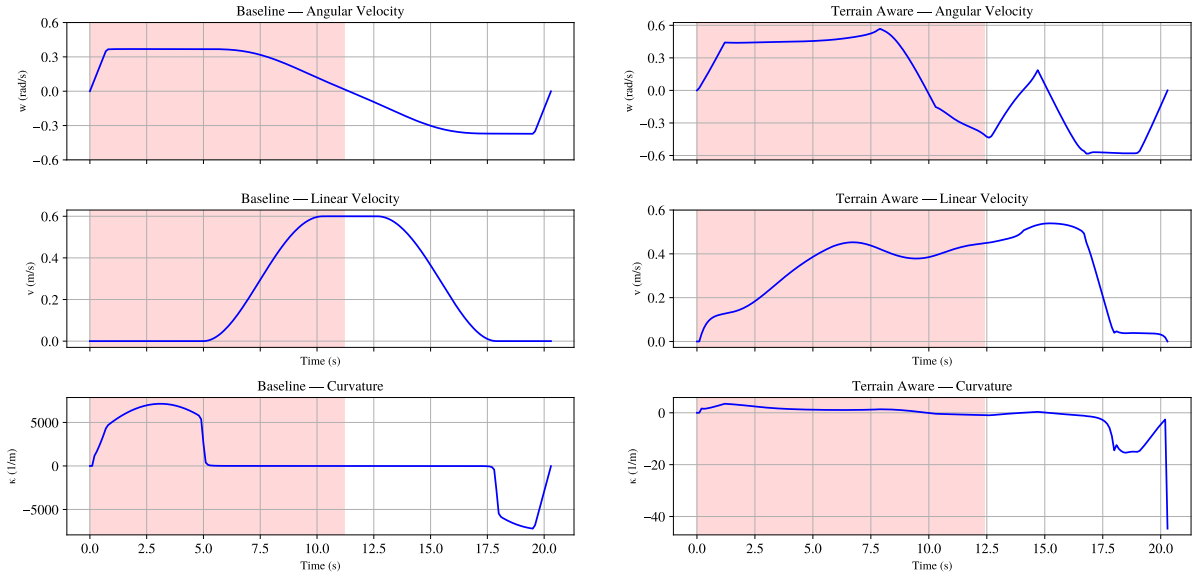


Figure C.8.: Velocity profiles for Edge Case 3. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Edge Case 4

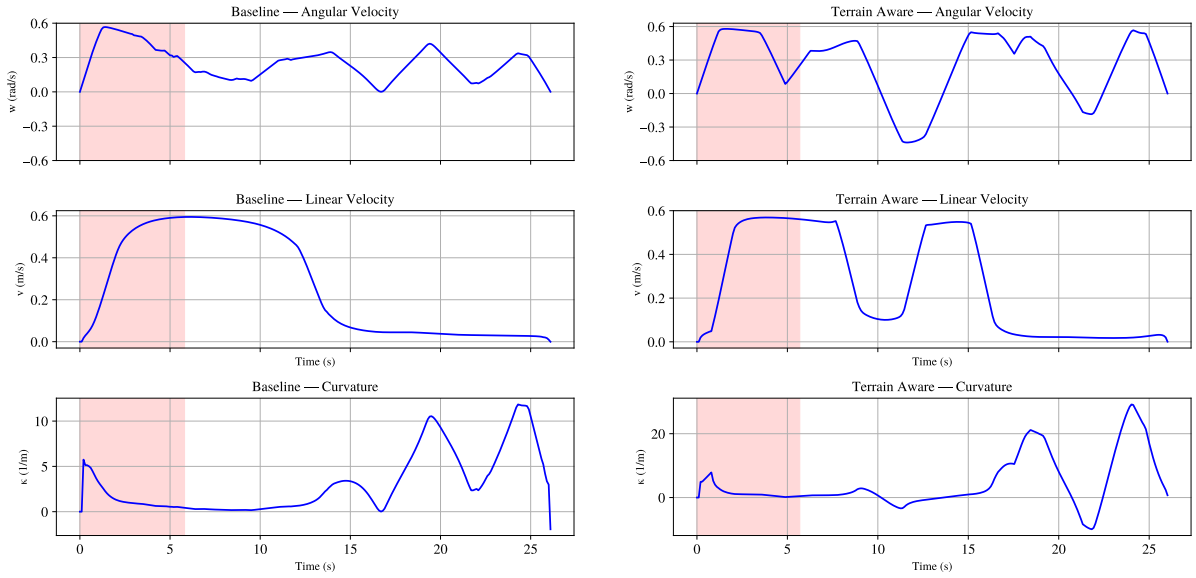


Figure C.9.: Velocity profiles for Edge Case 4. Top subplot: $w(t)$, middle subplot: $v(t)$, bottom subplot: $\kappa(t)$. Rough-terrain intervals are shaded in red.

Bibliography

[1] Arturo Rankin, Nikunj Patel, Evan Graser, Jiun-Kai Freddy Wang, and Kimberly Rink. Assessing mars curiosity rover wheel damage. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–19, 2022. doi: 10.1109/AERO53065.2022.9843634.

[2] Olivier Toupet, Jeffrey J. Biesiadecki, Arturo L. Rankin, Amanda Steffy, Gareth Meirion-Griffith, Dan Levine, Maximilian Schadegg, and Mark W. Maimone. Terrain-adaptive wheel speed control on the curiosity mars rover: Algorithm and flight results. *Journal of Field Robotics*, 37:699 – 728, 2019. URL https://api.semanticscholar.org/CorpusID:203029106.

[3] Mark Maimone, Neil Abcouwer, PJ Rollins, Evan Hilgemann, Freddy Wang, Nikunj Patel, and Alexandra Holloway. These wheels are made for arc-ing: Two new mobility commands to improve wheel wear outcomes. In *2023 IEEE Aerospace Conference*, pages 1–12, 2023. doi: 10.1109/AERO55745.2023.10115731.

[4] NASA Science. Perseverance rover components — wheels and legs. NASA public website, 2020. URL https://science.nasa.gov/mission/mars-2020-perseverance/rover-components/#wheels. Accessed: 2025-12-03.

[5] Joseph Carsten, Arturo Rankin, Dave Ferguson, and Anthony Stentz. Global path planning on board the mars exploration rovers. In *2007 IEEE Aerospace Conference*, pages 1–11, 2007. doi: 10.1109/AERO.2007.352683.

[6] Mikhail Pivtoraiko and Alonzo Kelly. Fast and feasible deliberative motion planner for dynamic environments, May 2009.

[7] William Reid, Robert Fitch, Ali H. Göktoğan, and Salah Sukkarieh. Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover. *Journal of Field Robotics*, 37(5):786–811, 2020. doi: https://doi.org/10.1002/rob.21894. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21894.

[8] Liang Ding, Zongquan Deng, Haibo Gao, Keiji Nagatani, and Kazuya Yoshida. Planetary rovers' wheel—soil interaction mechanics: new challenges and applications for wheeled mobile robots. *Intell. Serv. Robot.*, 4(1):17–38, January 2011. ISSN 1861-2776. doi: 10.1007/s11370-010-0080-5. URL https://doi.org/10.1007/s11370-010-0080-5.

[9] Armin Wedler, Bernhard Rebele, Josef Reill, Michael Suppa, Heiko Hirschmüller, Christoph Brand, Martin Schuster, Bernhard Vodermayer, Heinrich Gmeiner, Annika Maier, Bertram Willberg, Kristin Bussmann, Fabian Wappler, Matthias Hellerer, and Roy Lichtenheldt. Lru

- lightweight rover unit. In *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 05 2015.

[10] A. Howard and H. Seraji. Vision-based terrain characterization and traversability assessment. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2756–2761, 2008.

[11] Steve Macenski, Matthew Booker, and Josh Wallace. Open-source, cost-aware kinematically feasible planning for mobile and surface robotics. *Arxiv*, 2024.

[12] G. Sakayori and G. Ishigami. Energy-aware trajectory planning for planetary rovers. *Advanced Robotics*, 35(21-22):1302–1316, 2021. doi: 10.1080/01691864.2021.1959396. URL https://doi.org/10.1080/01691864.2021.1959396.

[13] Michael McHenry, Neil Abcouwer, Jeffrey J. Biesiadecki, Johnny Chang, Tyler Del Sesto, Andrew Johnson, Todd Litwin, Mark Maimone, ‡ JackMorrison, R. W. Rieber, Olivier Toupet, and Philip Twu. Mars 2020 autonomous rover navigation. URL https://api.semanticscholar.org/CorpusID:266602329.

[14] Ryota Takemura and Genya Ishigami. Traversability-based RRT* for planetary rover path planning in rough terrain with lidar point cloud data. *Journal of Robotics and Mechatronics*, 29(5):838–846, 2017. doi: 10.20965/jrm.2017.p0838.

[15] Amanda Steffy Mark Maimone Masahiro Ono, Thoams J. Fuchs and Jeng Yen. Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In *2015 IEEE Aerospace Conference*, pages 1–10, 2015. doi: 10.1109/AERO.2015.7119022.

[16] Thomas M. Howard and Alonzo Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, 26:141 – 166, 2007. URL https://api.semanticscholar.org/CorpusID:69802.

[17] Y. Li, S. Liang, J. Gao, Z. Chen, S. Qiao, and Z. Yin. Trajectory optimization for the nonholonomic space rover in cluttered environments using safe convex corridors. *Aerospace*, 10(8):705, 2023. doi: 10.3390/aerospace10080705. URL https://doi.org/10.3390/aerospace10080705.

[18] X. Yu, P. Wang, and Z. Zhang. Learning-based end-to-end path planning for lunar rovers with safety constraints. *Sensors*, 21(3):796, 2021. doi: 10.3390/s21030796. URL https://doi.org/10.3390/s21030796.

[19] T. Tanaka and H. Malki. A deep learning approach to lunar rover global path planning using environmental constraints and the rover internal resource status. *Sensors*, 24(3):844, 2024. doi: 10.3390/s24030844. URL https://doi.org/10.3390/s24030844.

[20] 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. doi: 10.1109/70.508439.

[21] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998. URL https://api.semanticscholar.org/CorpusID:14744621.

[22] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. doi: 10.1177/0278364911406761.

[23] Matthew P. Kelly. Transcription methods for trajectory optimization: a beginners tutorial, 2017. URL https://arxiv.org/abs/1707.00284.

[24] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006. URL https://api.semanticscholar.org/CorpusID:14183894.

[25] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1 (1):269–271, December 1959. ISSN 0029-599X. doi: 10.1007/BF01386390. URL https://doi.org/10.1007/BF01386390.

[26] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

[27] Mihail Pivtoraiko, Ross A. Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26:308–333, 2009. URL https://api.semanticscholar.org/CorpusID:116579266.

[28] Mihail Pivtoraiko and Alonzo Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3231–3237, 2005. doi: 10.1109/IROS.2005.1545046.