Uncertainty-aware motion planning for ground vehicle in unstructured uneven off-road terrain

by

Ahmed Hamouda

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2024

© Ahmed Hamouda 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Navigating large, unstructured, and uneven off-road environments, such as those encountered in search and rescue missions or planetary exploration, presents significant challenges. These environments are characterized by varying terrain semantics and complex geometries. Furthermore, initial map representations are often uncertain, as they are typically generated from aerial scans or other remote sensing techniques that may provide incomplete or outdated data. Existing algorithms that focus on planning a single path through the environment frequently overlook the opportunity to incorporate future information gathered during navigation, which can be used to reduce the expected traversal cost.

In this thesis, we propose an uncertainty-aware motion planning framework. The framework starts by integrating both geometric and semantic terrain data to assess terrain traversability. We then utilize an unsupervised region clustering algorithm to segment uncertain regions and group grids with similar visual and spatial features. Following this, our approach is structured into three stages: generating a network of pathways, constructing a stochastic graph, and developing an optimal navigation policy. A multi-query sampling-based planner is used to create a comprehensive network of pathways between the start and goal points, efficiently exploring multiple potential routes. These pathways are then converted into a topological stochastic graph representation of the environment, capturing uncertainty through probabilistic edge representations. The stochastic graph is modeled as a Canadian Traveler Problem (CTP), which is a decision-making framework designed for navigating graphs where some edges have a probability of being blocked. To minimize the expected traversal cost, we extend the state-of-the-art CTP solver CAO*, introducing Complete CAO* (CCAO*), which guarantees to produce a navigation policy that minimizes the expected traversal cost, even when no deterministic path exists.

We validate our framework through extensive simulations using real-world off-road data, testing both small and large environments to assess scalability. Results demonstrate that our approach consistently generates compact graph representations, unaffected by uncertain regions that do not impact the robot's movement. These findings highlight the framework's computational efficiency, robustness, and ability to reduce expected traversal costs when compared to traditional baseline methods.

Acknowledgements

I would like to thank my supervisors, Professor Baris Fidan and Professor Stephen Smith, for their continuous support and guidance throughout my master's journey. Their expertise, guidance, and patience have been instrumental in my growth as a researcher, and I have learned alot from them.

I am also deeply thankful to my friends, whose support and companionship made this journey more manageable and enjoyable. Their encouragement during challenging times kept me motivated.

To my beloved family—my mother, father, sister, brother, and in-laws—thank you for your unwavering support, understanding, and love. Their belief in me has been a constant source of strength.

To the love of my life, Nouran, whose unwavering support and belief in me have been my greatest motivation. Without her by my side, I would not have been able to complete this journey.

Lastly, I am thankful to Allah for granting me the opportunity and the strength to finish my master's degree. All praise is due to Him, who made this achievement possible.

Dedication

This is dedicated to my family and to the love of my life, Nouran.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures viii

List of Tables x

1 Introduction 1

1.1 Thesis Contributions 4

1.2 Thesis Organization 4

2 Literature Review 6

2.1 Terrain Traversability Analysis 6

2.2 Path Planning in Unstructured Environments 7

2.3 Uncertain Aware Motion Planning 8

3 Background **12**

3.1 Motion Planning Algorithms . . . 12

3.1.1 Graph Search Planners . . . 12

3.1.2 Sampling Based Planners . . . 13

3.2 Canadian Traveler Problem . . . 16

4 Uncertainty-Aware Path Planning Framework: Formulation and Approach **21**

4.1 Problem Formulation . . . 21

4.2 Solution Approach . . . 23

4.2.1 Approach Overview . . . 24

4.3 Generating A Network of Pathways . . . 26

4.4 Generating A Stochastic Graph . . . 28

4.5 Generating A Policy . . . 31

5 Simulation Environment and Performance Evaluation **34**

5.1 Simulation Environment . . . 34

5.1.1 Generating Traversability Cost Map . . . 35

5.1.2 Uncertain Region Extraction . . . 38

5.2 Baselines . . . 39

5.3 Simulation Scenarios and Policy Evaluation . . . 40

5.3.1 Simulation Scenario A . . . 42

5.3.2 Simulation Scenario B . . . 50

5.3.3 Simulation Scenario C . . . 53

6 Conclusion and Future Work **57**

References **59**

List of Figures

1.1 Off-road terrain park in Michigan, USA 1

1.2 Illustrative example of path planning given an initial uncertain map estimate 3

3.1 Simple undirected graph representing a CTP instance. 16

3.2 Complete AO tree corresponding to the CTP instance on the graph introduced in Figure 3.1 18

4.1 Illustrative figure of the disambiguation region and the chosen disambiguation point given an uncertain region and a path 27

4.2 Illustrative figure showing a scenario where there is no determinisitc path between start and goal points 31

5.1 Aerial view of the Michigan off-Road testing site used for policy evaluation . 34

5.2 Ground truth semantic labels for our environment 35

5.3 Digital Elevation Map (DEM) showing elevation variations in the environment. 36

5.4 Terrain slope derived from the digital elevation map. 36

5.5 Terrain roughness derived from the digital elevation map. 37

5.6 Geometric classification map based on slope and roughness analysis. 38

5.7 Final traversability map combining geometric and semantic classifications. . 38

5.8 Unsupervised superpixel segmentation of the environment. 39

5.9 Gridgraph representation of the environment 41

5.10 Illustration of our approach in generating network of relevant pathways . . . 44

5.11 Generated network of pathways for scenario A 45
5.12 Stochastic graph representation of small environment 46
5.13 (Left) Spatial layout of the proposed policy. (Right) Solution subtree generated by the CCAO* algorithm. 48
5.14 (Left) Spatial layout of the optimistic policy. (Right) Decision tree for the optimistic policy. 49
5.15 (Left) Spatial layout of the conservative policy. (Right) Decision tree for the conservative policy. 49
5.16 Generated stochastic graph for Scenario B 51
5.17 Expanded AO tree using CCAO* with the policy subtree colored in green . 52
5.18 Aerial view of the full environment and the uncertain regions (red regions) used in Scenario C. 53
5.19 Expected cost vs the number of uncertain regions in the environment. 54
5.20 Total computation time vs the number of uncertain regions in the environment. 55
5.21 Total run time vs. number of regions, comparing policy generation with and without pruning. 56

List of Tables

5.1 Graph generation results for Scenario A 47
5.2 Policy generation results for Scenario A 50

Chapter 1

Introduction

Robotic navigation during search and rescue missions [1] or planetary exploration operations [2, 3] faces unique challenges as they require the robot to navigate in large, off-road, unstructured, and uneven environments. These environments often lack defined paths or roadways, with irregular ground surfaces and varying elevations and slopes. The terrain also includes a wide range of semantics, such as grasslands with trees and vegetation, rocky areas with gravel, loose sand, muddy sections, and water bodies, as shown in the example in Figure 1.1.

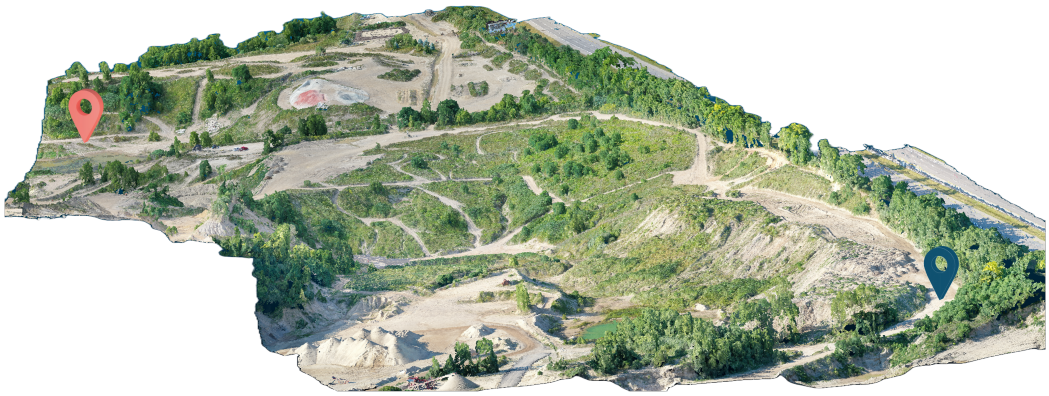


Figure 1.1: Off-road terrain park in Michigan, USA

Simplifications such as assuming movement on a flat surface are often feasible when navigating structured environments. However, additional complexities must be addressed in uneven terrain, including the terrain's geometry and surface characteristics. These

diverse geometric and semantic features cause the traversability cost of each terrain segment to vary considerably. This cost reflects the effort or energy required for a robot to navigate through these segments. For instance, traversing flat terrain is much easier than navigating rough or sloped terrain, resulting in a lower traversability cost for flat surfaces. This concept can also be applied to different terrain types, where asphalt, for example, is more favorable than mud or ice.

Path planners that guide robots across off-road, unstructured, and uneven terrain need to identify a path that is not only short but also traverses the least challenging terrain. Failing to consider the traversability cost could result in selecting a path that, while short, is infeasible or potentially damaging to the robot, leading to mission failure or mechanical breakdown.

Existing traversability analysis methods in the literature differ in their approaches; some focus on geometric aspects like slope and roughness [4, 5, 6], while others use semantic classification to assign costs to different terrain types [7, 8]. However, relying solely on either geometric or semantic features can lead to inaccurate assessments. For instance, while grass may be geometrically rough, it is typically traversable, whereas water bodies, despite appearing smooth, are semantically untraversable. Therefore, integrating both geometric and semantic aspects is essential for accurate terrain traversability assessments, as discussed in [9, 10, 11, 12].

Furthermore, prior knowledge of the environment is typically obtained through maps generated by satellite imagery or an aerial vehicle [13]. However, converting this imagery into usable forms for navigation, such as point clouds [4] or costmaps [14], introduces complexities and inaccuracies in representing the actual environment. These inaccuracies can arise from factors including the resolution of satellite images, atmospheric conditions during image capture, natural dynamic changes in the environment since the images were taken, and the inherent limitations of satellite imagery in assessing terrain.

Failing to account for these uncertainties can lead to two problematic approaches. One might take an overly optimistic view of the map and plan a path from the start to the goal under the assumption that the uncertain regions are actually traversable. However, this optimistic path could either cause the robot to become stuck or require multiple re-planning steps, resulting in a significantly longer route than initially expected. Conversely, adopting a pessimistic perspective may lead to a conservative path that avoids any regions of uncertainty. While this approach reduces the risk of encountering obstacles, it often results in a much longer path to the goal, bypassing more direct routes that could have been navigable. An illustrative example is provided in Figure 1.2.

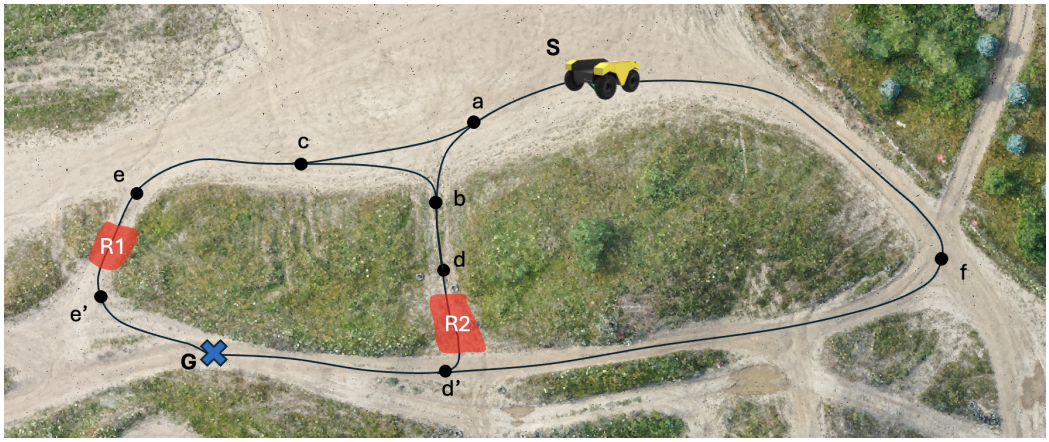


Figure 1.2: Illustrative example of path planning given an initial uncertain map estimate

The figure illustrates a scenario where a robot is positioned at the start point S and needs to navigate to the goal point G. The current map estimate is incomplete, and there are two regions, $R1$ and $R2$, whose true traversability is unknown. These regions could potentially be untraversable, posing significant challenges to the robot's path planning.

If the robot adopts an optimistic approach, it will initially follow the direct path $S \to a \to b \to d \to d' \to G$. However, if $R2$ turns out to be an obstacle, the robot would be forced to stop at point d and backtrack to point b. From there, it would reroute through $b \to c \to e \to e' \to G$. If $R1$ also proves to be untraversable, the robot would have to stop at e and take an even longer detour, following the path $e \to c \to a \to S \to f \to d' \to G$. This extended route results in a much longer and more complex journey than initially anticipated, demonstrating the risks associated with an overly optimistic strategy.

On the other hand, the robot might opt for a more conservative approach, choosing to navigate directly from S to f, then to d', and finally to the goal G. While this path avoids the uncertainty of $R1$ and $R2$, it also misses the opportunity to take a potentially shorter route if $R2$ turns out to be traversable.

Therefore, it is crucial to consider that the robot's understanding of the environment might improve as it progresses through its journey. As the robot collects real-time data, it should dynamically reassess the risks and penalties associated with detouring. The decision to take a riskier or more conservative path should be based on the probability of each scenario and the expected cost of the traversed path. Thus, it is essential to develop a policy that considers all possible outcomes and aims to minimize the expected cost of the traversal, balancing the trade-offs between risk and efficiency.

This thesis addresses the challenge of guiding a ground robot from a starting point to a goal point across large, unstructured, and uneven terrain, given an initial uncertain environment map.

1.1 Thesis Contributions

In this work, we introduce a new problem inspired by the Stochastic Optimal Search Problem (SOSP) but with key modifications to address the challenges of navigating large, unstructured off-road environments. Our problem formulation focuses on a robot navigating through an off-road environment with an uncertain initial map representation—a scenario commonly encountered in real-world applications. Unlike many existing approaches, we do not impose any assumptions regarding obstacle shapes, the guarantee that a path exists from the start point to the goal point, or the maximum number of disambiguations that can be made, making our method more flexible and applicable to a wide range of environments.

Our approach involves utilizing a multi-query planner to generate a network of pathways, encapsulating the various potentially relevant routes between the start and goal points in the environment. These paths are used to generate a compact stochastic graph representation of the environment, which is iteratively updated as the robot traverses the terrain. We leverage this graph to formulate a Canadian Traveler Problem (CTP), a framework for navigating stochastic graphs with uncertain edges, allowing the robot to adjust its decisions as new information is acquired during navigation.

To solve this, we modify the state-of-the-art CTP solver CAO*, introducing Complete CAO* (CCAO*), which guarantees a solution even when no deterministic path exists, thus ensuring algorithm completeness. Finally, we conduct extensive testing using real-world data, demonstrating the scalability and efficiency of our approach compared to existing methods.

1.2 Thesis Organization

The remainder of this thesis is organized into five chapters. Chapter 2 presents a comprehensive Literature Review, focusing on three key areas relevant to this research: terrain traversability analysis, path planning in unstructured environments, and uncertainty-aware motion planning. The chapter reviews state-of-the-art techniques in these domains and highlights the gaps addressed in this work.

In Chapter 3, we provide a comprehensive background on the key motion planning algorithms, including classical graph search methods and sampling-based approaches. Additionally, we delve into the Canadian Traveler Problem (CTP) and its advanced algorithmic solutions, including AO* and CAO*.

Chapter 4 presents a detailed formulation of the uncertainty-aware global motion planning problem in unstructured, uneven off-road terrain. We outline our solution approach, which includes generating a network of pathways, constructing a stochastic graph, and formulating a policy using an enhanced version of the CAO* algorithm.

In Chapter 5, we describe the experimental setups and simulation environments used to validate our approach. This chapter includes the generation of the traversability cost map, extraction of uncertain regions, and the testing results of three simulation scenarios that demonstrate the effectiveness of our uncertainty-aware path planning framework.

Finally, in Chapter 6, we summarize the key contributions of this thesis, discuss the limitations of the proposed method, and suggest potential avenues for future research.

Chapter 2

Literature Review

In this chapter, we provide a literature review of the key research areas relevant to this thesis. We start with traversability analysis, focusing on methods that assess terrain navigability using geometric and semantic features. Next, we review path planning in unstructured environments, highlighting techniques for generating feasible paths in cluttered, off-road terrains. Finally, we examine uncertainty-aware motion planning, discussing strategies to manage environmental uncertainties and enhance the robustness of autonomous navigation systems.

2.1 Terrain Traversability Analysis

As previously discussed, the environment in which the robot operates varies significantly in both its geometric and semantic characteristics, directly impacting the traversability cost of each terrain segment. This cost reflects the effort or energy needed for the robot to navigate these segments. Off-road traversability assessment is a well-researched area in the literature, as it plays a crucial role in off-road robot navigation.

In recent work, several approaches have been developed to classify terrain traversability into different terrain classes based on various criteria. For instance, the study in [8] classifies each point in a newly sensed point cloud into traversable, risky, or untraversable categories based on height and normal vector features. This classification is then projected onto a local traversability cost map for trajectory planning. In [10], a deep neural network is developed to classify local terrains into four cost classes (free, low-cost, medium-cost, and obstacle) using sparse LiDAR inputs. The network outputs a traversability map where each

cell stores a probability distribution over these four class labels. Similarly, in [9], a fully convolutional network classifies input images into 34 different semantic classes, providing a per-pixel probability distribution. Geometric features like step height, slope, and roughness are also calculated and then fused with the output of the fully convolutional network to train another classifier that predicts traversability classes such as safe, risky, or obstacle.

On the other hand, some studies emphasize regression-based approaches that provide a continuous traversability value. For instance, in [15], geometric traversability costs are calculated by combining normalized slope, roughness, and step height values. These values are then integrated with an empirical cost assigned to each semantic class to produce a final traversability score using a weighted sum. Another method, described in [12], sets the geometric cost to 1 (maximally traversable) if both slope and height are below a safe threshold, and 0 (completely untraversable) if they exceed a critical threshold. For intermediate values, the geometric cost is determined as a weighted sum of the normalized slope and height. The final traversability cost is set to 0 if the grid's semantic classification is untraversable (e.g., water or rock), fully traversable if the classification is flat, and to the geometric cost otherwise.

2.2 Path Planning in Unstructured Environments

Krüsi et al. [4] proposed a comprehensive framework for autonomous navigation of ground vehicles in unstructured, off-road 3D environments. Their approach computes kinematically feasible trajectories directly on point cloud maps, where each point is assigned a traversability cost based on terrain roughness and the robot's orientation. The framework employs a hierarchical approach with three stages: initial global path planning with the bidirectional RRT-Connect algorithm, followed by path refinement using RRT* to optimize path length, and finally, trajectory optimization through a gradient-free method.

In [14], Overbye et al. presented an approach for fast online mapping and local planning in off-road environments. The method processes raw point cloud data to generate a local cost map based on obstacle positions, certainty of data, and height gradients. Using this cost map and the distance to the goal as a heuristic, an initial path is generated utilizing the A* algorithm. Subsequently, the system samples from the control space to produce kinematically feasible trajectories, selecting the optimal one based on factors like distance to a target point, heading error, and cumulative grid cost

Zhang et al. [16] introduced humanlike hierarchical motion planning algorithms for mobile robots in unstructured, partially known, dynamic indoor environments. The proposed algorithm consisted of three main layers. First, the global planner uses AD* to

generate a path on a low-resolution map. Second, the sensor-level planner selects target poses by evaluating obstacles, orientation, and distance, using ARA* for path refinement. Finally, the action-level planner selects motion primitives based on curvature and obstacle avoidance, ensuring smooth paths through dense sampling.

In [17], Shen et al. introduced a three-stage hierarchical path planning algorithm for nonholonomic autonomous vehicle navigation in 3D environments. The process begins with the D* Lite algorithm, which guides RRT* sampling. A Gaussian distribution is used for sampling around the reference points generated by the D* Lite algorithm. After RRT* terminates, a local path optimization based on the one presented in [4] by Krusi et al. is applied to smooth out any path windings.

Similar to [17], Brunner et al. introduced a two-phase hierarchical motion planner in [18]. A height map was used to represent the environment. Terrain roughness was calculated by applying a max filter using a squared kernel to the height difference. A weighted graph is then constructed, where the cost of each edge corresponds to the traversal time between two points. A graph search algorithm like A* or Dijkstra is implemented to find an initial path to the goal. Similar to [17], RRT* is then guided by sampling with a Gaussian distribution centered at the initial path.

2.3 Uncertain Aware Motion Planning

Although motion planning for ground vehicles in 3D environments has been extensively studied in the literature [4, 17, 14, 19], effectively managing the uncertainties present in the initial map representations remains a significant challenge. The ability to plan safe and optimal paths while balancing the tradeoff between them continues to be a crucial area of research.

One common strategy to handle uncertainties is to transform the uncertain motion planning problem into a more manageable deterministic framework, which can be achieved through several methods found in the literature. Among these strategies, the optimistic policy has been widely adopted due to its straightforward implementation. This approach involves planning a path assuming the initial map estimate is accurate and then re-planning as needed if obstacles are encountered. Numerous algorithms, such as D*Lite [20] and LPA [21], have been developed to enhance the computational efficiency of this method.

These algorithms are effective when there is a small difference between the initial map estimate and the actual environment, such as navigating around a muddy region that

was not originally mapped. However, this approach becomes less effective when the encountered obstacles require large detouring, which could have been avoided with better initial planning, as illustrated in the example presented in Figure 1.2, where following an optimistic approach might lead to a much longer journey than originally anticipated.

Fan et al. have accounted for uncertainties by assessing their associated risks by calculating the Conditional Value-at-Risk (CVaR) metric [5] and then apply a hierarchical risk-aware planner that finds a feasible trajectory by minimizing the CVaR_α. This risk metric calculates the expected value of the traversability cost in the conditional distribution of cost's upper α-quantile [22]. In [23], Fan et al. proposed a CNN architecture that outputs a CVaR cost map without the need to assume any distribution function.

Alternative approaches in the literature involve generating multiple hypotheses with varying safety levels [24]. First, the algorithm assumes all uncertain vertices are safe and computes an optimistic shortest path; it then iteratively identifies and re-evaluates potentially unsafe vertices, adjusting hypotheses to refine the path. This process yields a set of candidate paths, from which it selects the optimal path based on a balance of safety and distance considerations. Another method is the chance-constrained strategy, which characterizes uncertainty probabilistically by defining the safe boundaries around uncertain obstacles that would guarantee a bounded probability of collision with the obstacles [25, 26, 27].

The aforementioned studies focus on generating a single path that offers a degree of safety that can be adjusted through hyper-parameter tuning. However, this approach presents a significant limitation as it fails to consider the availability of future information sensed by the robot during navigation. In addition, correctly setting these hyper-parameters is challenging as it often results in either overly risky or overly conservative paths, and hence can not be generalized to all scenarios. Furthermore, algorithms that exploit the potential benefit of future disambiguations strictly reduce the expected traversal cost compared with any deterministic shortest path that has zero risk [28].

The concept of incorporating future information availability has been explored in literature in the context of the 'next best view' (NBV) strategy [29, 30]. This method typically involves generating up to two path hypotheses and evaluating NBV poses to reduce uncertainty. The final path is chosen depending on the newly sensed data obtained at the NBV pose. However, this technique is generally applied in a local navigation setting and considers only a single time step.

This work examines a planning framework suitable for navigating terrains with uncertain obstacles, referred to in the literature as the continuous Stochastic Obstacle Scene Problem (continuous SOSP) [31]. This concept was originally introduced in foundational

work on the subject [32]. The SOSP involves navigating from a start point to a goal point in an environment with potential disk-shaped obstacles. Each obstacle is associated with a probability indicating the likelihood of it being an actual obstacle. The objective is to develop a navigation policy that plans a continuous path from the start to the goal, determining the optimal locations and timings for disambiguating uncertain regions to minimize the expected traversal cost.

The computational challenges of the continuous SOSP are well-documented, with Papadimitriou et al. [32] demonstrating the intractability of its various forms. Building upon this, Aksakalli et al. [33] have further discussed the challenges involved, particularly the difficulty in computing optimal disambiguation points for non-trivial instances of the problem. A more approachable discrete variant of the problem is the well-known Canadian Traveller Problem (CTP) [32]. CTP is defined on a stochastic graph where each edge has a certain probability of being blocked. The unique aspect of CTP is the ability to disambiguate these edges at one of their adjacent vertices. The primary objective is to develop a navigation policy that strategically decides which edges to disambiguate and the timing of such actions to minimize the expected traversal cost. A more detailed overview of CTP and some of its algorithms can be found in Section 3.2.

CTP finds practical application in various fields, such as guiding ground robots through uncertain terrains [34], optimizing aircraft routes [35], and aiding autonomous surface vehicles in navigating with the help of historical satellite images [36].

The Canadian Traveler Problem (CTP) can be modeled as both a Markov Decision Process (MDP) and a Deterministic Partially Observable Markov Decision Process (Deterministic POMDP) [37]. However, Fried et al. [38] have shown that the CTP is PSPACE-complete, indicating that the MDP and POMDP formulations will contain an exponential number of states [34]. To find exact solutions to the CTP, the AO* algorithm [39] and its more advanced variations [13, 40, 37] are commonly used due to their efficiency. AO* is a heuristic search algorithm that finds optimal solutions in state-space problems represented by AND/OR trees. For further details on the AO* algorithm, refer to [37].

To formulate our problem as a CTP, we need to represent our environment as a stochastic graph. However, in CTP literature, it is often assumed that the graph is either pre-existing [37] or created by manually guiding the robot through the environment [34]. Both of these assumptions are unrealistic in real-world scenarios. Other strategies include grid discretization of the environment [41, 42] and lattice discretization [33]. While these approaches simplify the problem, they result in a vast state space. For instance, consider an $m \times n$ gridmap where each node can be categorized as traversable, untraversable, or unknown. The resulting state space involves $3^{(m.n)}$ distinct information states [43], making

solving the CTP computationally challenging [44, 31, 45]. This challenge is particularly pronounced in large-scale, cluttered off-road terrains, where a fine grid resolution is necessary for accurate terrain feature estimation [46]. Furthermore, as the discretization resolution increases, the number of vertices and edges in the approximating graph also grows [31], rendering the solution intractable [47].

In much of the existing CTP literature, it is often assumed that at least one path from the start point to the goal point exists that is composed entirely of deterministic edges, thereby ensuring a finite expected traversal length. If no such path is found, the algorithms typically fail to return a policy, resulting in incompleteness. This assumption neglects the scenario where a path to the goal could still be feasible through the disambiguation of uncertain edges, should they prove to be traversable. Our approach challenges this assumption by extending state-of-the-art algorithms to guarantee completeness, allowing for the discovery of a path to the goal even when it necessitates traversing uncertain edges.

Moreover, traditional formulations of the CTP often impose a limit on the number of disambiguations that can be performed. In contrast, our work removes this restriction, permitting the agent to execute an unlimited number of disambiguations as required, thereby making the problem formulation more realistic and applicable to real-world scenarios.

Chapter 3

Background

This chapter provides an essential foundation for the algorithms utilized in this thesis. We introduce multiple motion planning algorithms, including the classical graph search methods, Dijkstra's and A*, as well as sampling-based approaches like PRM*. Additionally, we explore various sampling methods, including Gaussian sampling and the Randomized Bridge Builder (RBB) technique. The chapter also delves into the Canadian Traveler Problem (CTP), offering a detailed exploration of its formulation and the advanced algorithms like AO* and CAO* developed to address its challenges.

3.1 Motion Planning Algorithms

3.1.1 Graph Search Planners

A weighted undirected graph is formally defined as a graph $G = (N, E)$ where N is a set of nodes with cardinality $|N|$, and $E \subseteq \{\{u, v\} \mid u, v \in N, u \neq v\}$ is a set of edges with cardinality $|E|$. Each edge $\{u, v\} \in E$ is an unordered pair of distinct nodes, indicating a connection between them. The cost function $c : E \to \mathbb{R}_{\geq 0}$ associates each edge $e \in E$ with a non-negative cost function indicating the cost of traversal. Since the graph is undirected, the edge $\{u, v\}$ is identical to $\{v, u\}$, with the same cost of traversal.

Dijkstra's algorithm [48] is a well-established algorithm for finding the shortest paths between a source node and every other node in a weighted graph. The algorithm works by repeatedly choosing the unvisited node with the least distance to the start node. The selected node is then used to calculate the distance of its unvisited neighbor nodes and

update their distance if the newly calculated distance is smaller than the currently assigned distance. The node is then marked as visited, and the process continues until all reachable nodes are marked visited.

The A* algorithm [49], on the other hand, enhances Dijkstra's approach by incorporating heuristics that guide the search process. A* chooses nodes based on the sum of the cost from the start node to the current node and an estimated cost from the current node to the goal. This estimated cost must be an admissible heuristic, meaning it is a lower bound on the actual cost to reach the goal to ensure that the path found is optimal. This heuristic-driven approach allows A* to find the shortest path to a goal node more efficiently, often without visiting all nodes in the graph.

3.1.2 Sampling Based Planners

Sampling-based methods are a class of path-planning algorithms that are particularly useful for finding feasible paths in high-dimensional spaces. Unlike grid-based search methods, which discretize the space into a fixed grid structure, sampling-based methods randomly sample points in the free space, making them more efficient and scalable for large and complex environments. Two commonly used sampling-based methods are Rapidly-exploring Random Trees (RRT) [50] and Probabilistic Roadmaps (PRM) [51]. RRT is particularly effective for single-query problems, rapidly expanding the search tree towards unexplored regions, while PRM is well-suited for multi-query scenarios, constructing a roadmap of the environment that can be reused for different queries.

PRM motion planning algorithm consists of two phases: a graph construction phase followed by a query phase. During the construction phase, the algorithm randomly samples points in the free space and adds them to the graph. For each newly sampled point, the algorithm attempts to connect it to existing nodes within a specified radius. This connection can be made using a simple straight-line path or a more complex local planner, such as a spline. If the proposed edge is collision-free, it is added to the graph. This process is repeated for a predetermined number of nodes. In the query phase, the start and goal nodes are connected to the graph, and a path is found using a graph search algorithm like Dijkstra's [48] or A* [49].

PRM* [52] is an enhanced version of the PRM algorithm that is probabilistically complete. This means that the probability of finding a path (if one exists) approaches one as the number of sampled points increases. Instead of a constant connection radius, in PRM* for each new sampled node, the nearest neighbors are selected using an r-ball selection strategy, with the connection radius $r(i) = O((log(i)/i)^{(1/d)})$, where $r(i)$ is the radius of

the ball, i represents the number of samples, and d is the dimensionality of the configuration space. The pseudocode for the PRM* algorithm is presented in Algorithm 1. This approach ensures that the search radius starts large, allowing for broad exploration in the early stages, and gradually decreases as more nodes are sampled, refining the connections and improving the chances of finding an optimal path as the graph becomes denser.

Algorithm 1 PRM*(s, g, N)

```
Initialize G with start and goal configurations
for i ← 1 to N do
    q ← Sample()
    Add vertex q to G
    for all v ∈ Neighbors(G, q, i) do
        if IsVisible?(q, v) then
            Add edge (q, v) to G
        end if
    end for
end for
return ShortestPath(G, s, g)
```

Sampling is a fundamental aspect of sampling-based planners, enabling them to extend and explore the free space within a given environment. When PRM and RRT were first introduced, they utilized uniform sampling techniques. However, this approach has a significant limitation: the planner is more likely to sample nodes in wide, open areas rather than in narrow, constrained spaces. This occurs because each configuration in the environment has an equal chance of being selected, and since narrow regions contain fewer free configurations, they are less likely to be sampled, making exploring these regions more challenging. The following sampling strategies have been proposed as methods to address these limitations.

Gaussian sampling [53] is designed to bias the sampling process towards areas close to obstacles. In this method, a random node q is first generated, followed by the selection of a distance d based on a normal Gaussian distribution. A second node q' is then sampled at this distance from q. The technique then evaluates the feasibility of both q and q'. If q is feasible and q' is not, then q is added to the graph G. Conversely, if q is not feasible and q' is, then q' is added to the graph. This method ensures that the sampled nodes remain close to obstacles, which gives better coverage of the cluttered regions of the environment.

The Randomized Bridge Builder (RBB) [54] technique on the other hand is particularly

Algorithm 2 Gaussian Sampling

```
q ← UniformSample()
d ← a distance chosen according to a normal distribution
q′ ← a random conf. at distance d from q
if IsFeasible(q) and not IsFeasible(q′) then
    add q to the G
else
    if not IsFeasible(q) and IsFeasible(q′) then
        add q′ to the G
    end if
end if
```

effective for sampling in narrow passages. It begins by generating a random node q using a uniform distribution. If this configuration is not feasible, meaning it lies in an obstacle, the algorithm proceeds to select a distance d based on a suitable probability distribution. A second random node q' is then sampled at this distance d from q. If both q and q' are infeasible, the midpoint q_{mid} of the line segment connecting q and q' is calculated. If q_{mid} is feasible, it is inserted into the graph G as a new node. This technique is particularly useful in identifying and sampling within narrow passages that are otherwise difficult to explore.

Algorithm 3 Randomized Bridge Builder (RBB)

```
q ← UniformSample()
if not IsFeasible(q) then
    d ← a distance chosen according to a according to a probability density λ_x.
    q′ ← a random node at distance d from q
    if not IsFeasible(q′) then
        q_mid ← midpoint of line segment qq′.
        if IsFeasible(q_mid) then
            Insert q_mid into G as a new node.
        end if
    end if
end if
```

Other sampling methods include sampling around a Voronoi graph [55], sampling around an initial path [17], sampling in visibility regions [56], or a hybrid sampling technique that combines two sampling strategies [57, 58].

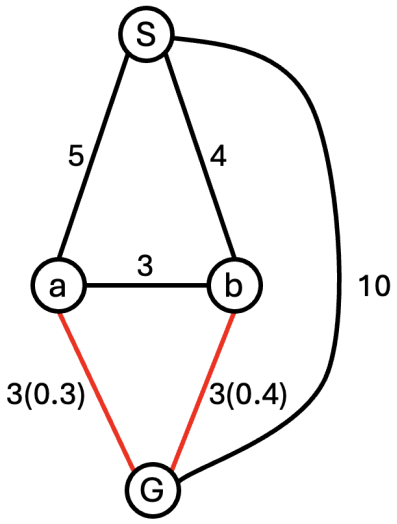


Figure 3.1: Simple undirected graph representing a CTP instance.

3.2 Canadian Traveler Problem

The Canadian Traveler Problem (CTP) can be formally described as follows: Consider an undirected stochastic graph $\mathcal{G} = (N, E)$, where each edge $e \in E$ is associated with a blocking probability function $p : E \to [0, 1]$ that represents the probability that e is not traversable. For any edge $e \in E'$, where E' is a subset of edges where $0 < p(e) < 1$, when the agent reaches one of its endpoints, they have the option to disambiguate e and determine whether e is traversable or not. An edge cannot be traversed until its traversability is confirmed, and the true state of each edge is assumed to be static throughout the traversal. The agent aims to find an optimal policy that minimizes the expected traversal cost from node s to node g in G. The problem of finding such an optimal policy is known as the CTP.

Figure 3.1 illustrates an instance of the Canadian Traveler Problem (CTP) on a simple undirected stochastic graph $\mathcal{G}$. The graph consists of four nodes: S (start), a, b, and G (goal). The edges $e \in E$ connecting these nodes have associated weights representing traversal costs. Among these edges, two are uncertain, shown in red, with associated blocking probability functions $p(e)$ such that $0 < p(e) < 1$. These uncertain edges belong to the subset E', and their true state (traversable or untraversable) remains unknown until the agent reaches one of their endpoints a or b and disambiguates them. The agent's objective is to determine the shortest traversal path from S to G while minimizing the expected traversal length under the uncertainty of these edges.

AO* is a heuristic search algorithm that finds the least-cost solution in state-space problems represented by AND/OR trees. The output of AO* is a policy represented as a solution graph within the larger AND/OR tree. AND/OR (AO) trees are used to model decision-making processes in uncertain environments, like the stochastic graphs in the CTP. The AO tree node set consists of two types of nodes. OR nodes correspond to possible disambiguation outcomes, while AND nodes serve as either leaf nodes for direct traversal to termination or a disambiguation point that has two successor OR nodes representing the traversable or untraversable disambiguation outcomes. In a CTP scenario, the tree is built with nodes representing states as (n, L), where 'n' is the agent's location, and L = {"A", "T", "U"}$^{\text{k}}$, which represents a knowledge vector about 'k' uncertain edges, with 'A' for ambiguous, 'T' for traversable, and 'U' for untraversable. The root of the AO tree is an OR node starting with all 'A's, followed by AND nodes for action choices like traversing to each reachable node that can disambiguate an uncertain edge or reach the goal node through a deterministic path if it exists. AND nodes branch into OR nodes for each possible outcome, and this pattern repeats.

Figure 3.2 shows the complete AO tree corresponding to the CTP instance on the graph introduced in 3.1. This AO tree captures all possible decisions, disambiguations, and outcomes as the agent navigates from the start node S to the goal node G under uncertainty of edge blockage. In the shown AO tree, OR nodes are represented as squares, AND nodes are represented as circles, and bold AND nodes represent terminal nodes.

The cost-to-go function $f : N \to \mathbb{R}_{\geq 0}$ assigns each node in the tree a non-negative value representing the expected cost of traversing from that node to the goal while following the optimal policy. This cost-to-go value for a node n is also known as the node's true label. AO* guides the AO tree expansion by incorporating admissible lower bounds $h_{\text{lower}} : N \to \mathbb{R}_{\geq 0}$, known as heuristic labels. These labels are carefully designed to never overestimate the true label of any node. Similar to A*, AO* is guaranteed to produce an optimal policy as long as the heuristic labels are admissible. A natural admissible lower bound is a deterministic shortest path from node n to the goal in a graph realization where all uncertain edges are considered traversable.

The AO tree is formed by iteratively performing a process that consists of two steps: expansion and propagation. In the expansion, the tree grows by finding the non-terminal node with the lowest heuristic label $h_{\text{lower}}(n)$ and adding its successor to the tree. Each successor is associated with a heuristic cost as the lower bound to their true label. In the propagation step, the label estimate of the expanded node is updated as follows:

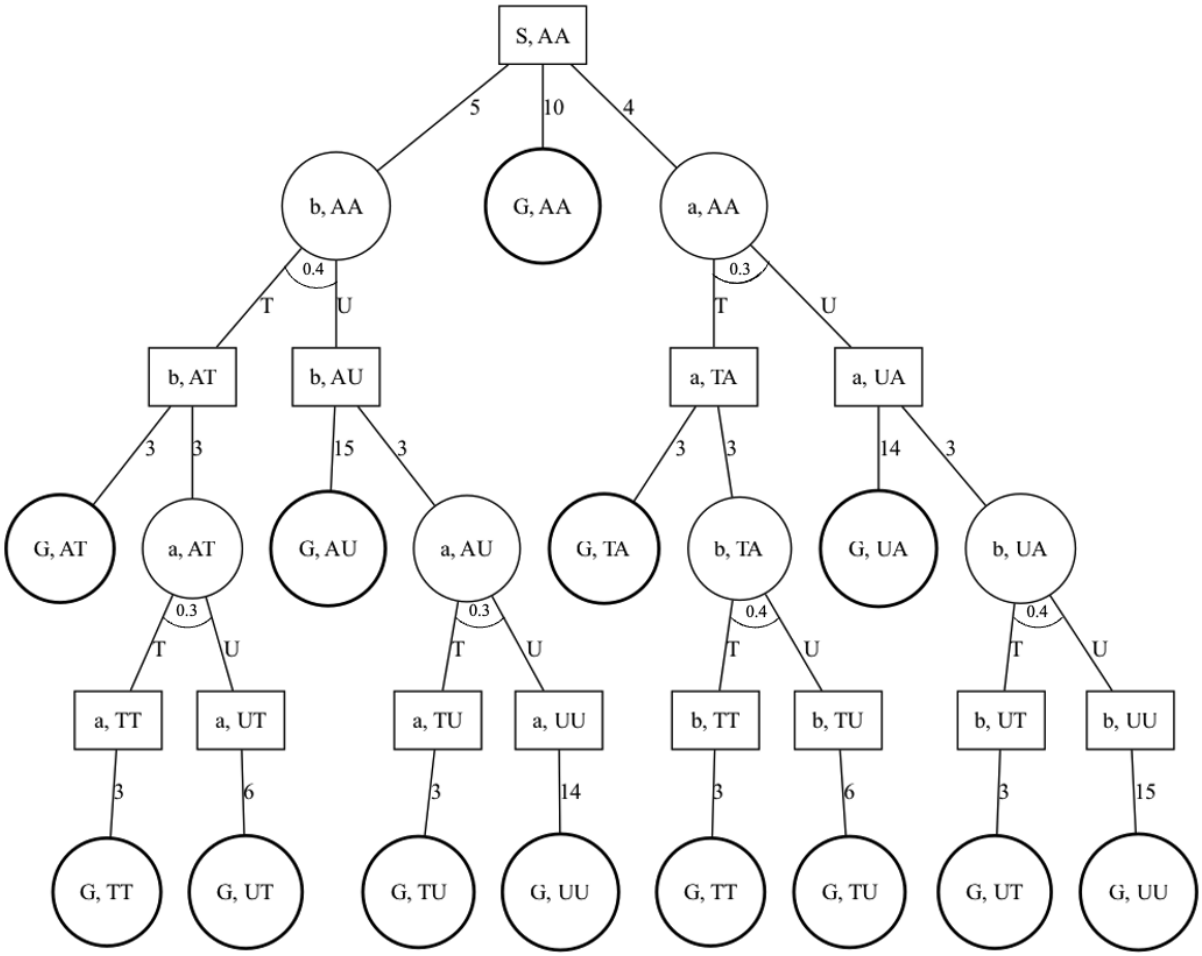


Figure 3.2: Complete AO tree corresponding to the CTP instance on the graph introduced in Figure 3.1

- If n is OR node, the cost is the minimum over all successors:

$$h_{\text{lower}}(n) = \min_{n' \in S(n)} \left(l(n, n') + h_{\text{lower}}(n') \right),$$

- If n is AND node, the cost is a weighted sum of its successors, considering the probability of each disambiguation outcome:

$$h_{\text{lower}}(n) = \sum_{n' \in S(n)} p(n, n') \cdot \left(l(n, n') + h_{\text{lower}}(n') \right),$$

where $S(n)$ is successor set of n, $l(n, n')$ is the traversal cost from n to n', and $p(n, n')$ is the probability of transitioning to successor n'. This cost update is propagated up the AO tree until the root node is reached or until it reaches a node where its label estimate is not affected.

The algorithm terminates when the root node is labeled as solved, meaning a complete solution has been found or if it is marked as unsolvable, indicating no possible solution. A node is marked as solved when the true label of that node is identified. For OR nodes, the true label is considered found when the true label of the best successor is found, and for AND nodes, it is when all successors are solved, indicating the best combined action to the goal state. Each node in the solution graph represents a state, and the edges denote the actions taken, with the policy guiding the agent to the goal with minimal expected cost. For more information about AO*, the reader is advised to check [37].

A more recent and advanced version of AO* is AO* with caching (CAO*) [37]. In CAO*, they employ two properties of the CTP and are used to significantly reduce the computation time needed to compute an exact solution for the CTP. First is the admissible upper bound of a node label, which is an estimate that never underestimates the true label of node n. A natural admissible upper bound is a deterministic shortest path from node n to the goal in a graph realization where all uncertain edges are considered untraversable. The cost of such a pessimistic path is an admissible upper bound on the label of node n. The second feature is the fact that a state (n, L) might be reached multiple times after following different sequences of states. By taking advantage of these two properties, Aksakalli et al. proposed the CAO* algorithm that has three main new improvements on AO* search that dramatically shorten the running time of the algorithm.

The first improvement is made during the expansion step, where an AND node is expanded only if $l(n_o, n_a) + \hat{h}_{\text{lower}}(n_a) \leq \hat{h}_{\text{upper}}(n_o)$; otherwise, it is discarded. Second, CAO* uses a hash map to keep track of all AND nodes added to the AO tree. When expanding the tree, it checks this cache to determine if a new AND node already exists.

If it does, the node is linked to the current OR node being expanded, and the OR node is added to the existing AND node's parent list. If the AND node is not found in the cache, a new one is created, connected to the current OR node, and then added to the cache. The third improvement involves dynamically pruning suboptimal AND nodes during the propagation phase. This is done by using updated labels to identify and remove "bad" AND nodes. A "bad" AND node n'_a is defined as a node for which the following condition holds:

$$l(\mathrm{no}, n'_a) + \hat{h}_{\mathrm{lower}}(n'_a) > \min_{n_a} \left[l(\mathrm{no}, n_a) + \hat{h}_{\mathrm{upper}}(n_a) \right].$$

Results presented in [37] show that by applying these three features, CAO* significantly shortens the computational time compared to the AO* algorithm [39] and its other advanced variations [13, 40].

Chapter 4

Uncertainty-Aware Path Planning Framework: Formulation and Approach

4.1 Problem Formulation

We define our unstructured, off-road environment as a closed and bounded 3D space, denoted by $\mathcal{M} \subseteq \mathbb{R}^3$. In our study, we assume that traversability analysis on $\mathcal{M}$ has been conducted using both geometric and semantic features, following the approach outlined in [9, 10]. This analysis results in the discretization of the environment into a 2D grid map, represented as $\hat{\mathcal{M}}_0 \in \hat{\boldsymbol{\mathcal{M}}}$, where $\hat{\boldsymbol{\mathcal{M}}}$ denotes the set of possible 2D gridmap realizations of $\mathcal{M}$. Each grid cell $i \in \hat{\mathcal{M}}_0$ is assigned a terrain class, each associated with a specific traversability cost.

All robot poses and paths in this study are evaluated in a continuous 2D projection of the original 3D environment, denoted by $\mathcal{M}_{2D}$. Although analyzed in 2D, these poses and paths correspond to the robot's actual movement within the 3D space, $\mathcal{M}$. The notation $\mathcal{M}_{2D}$ is used to emphasize this 2D projection as distinct from the 3D environment.

Let us define $\mathcal{S} = \{s_1, s_2, \ldots, s_n, s_{\text{untrav}}\}$ as the set of all possible terrain classes within this gridmap, where $\mathcal{S}_T = \{s_1, s_2, \ldots, s_n\}$ denotes the subset of traversable terrain classes. Each traversable class $s_j \in \mathcal{S}_T$ has a corresponding traversal difficulty cost, $D(s_j)$. While the class $s_{\text{untrav}} = \mathcal{S} \setminus \mathcal{S}_T$ is reserved to represent untraversable cells within the gridmap with a significantly high cost $D(s_{\text{untrav}})$ for traversing it, making those cells effectively impossible to cross.

The estimates of terrain's traversability classification at time step t, denoted as $\hat{\mathcal{M}}_t$, are probabilistic in nature. Each cell i, at time step t, holds a probability distribution $P_{i,t}(s_j)$ for $j = \{1, \cdots, n\}$, indicating the likelihood of being traversable terrain class s_j, and $P_{i,t}(s_{\text{untrav}})$ for being untraversable. The sum of probabilities for all classes equating to one: $\sum_j P_{i,t}(s_j) + P_{i,t}(s_{\text{untrav}}) = 1$.

As the robot navigates $\mathcal{M}$, it employs its laser scanner and camera sensors to effectively observe its surroundings. This enables the robot to continuously update the map estimate, denoted as $\hat{\mathcal{M}}_t$ at time step t. The observed set $\mathcal{O}_t \subseteq \hat{\mathcal{M}}_t$ is defined as the set of cells within the field of view of the robot's sensors at time step t. We assume the robot's sensors determine the true terrain class for each cell i in $\mathcal{O}_t$, denoted as $s_{\text{true}}(i)$. The map estimate is updated such that for each cell i in $\mathcal{O}_t$, $P_{i,t}(s_j) = 1$ if $s_j = s_{\text{true}}(i)$ and 0 otherwise, reflecting the robot's real-time observations.

Definition 4.1.1. (Cell certainty) A cell i is considered **certain** at time t if the maximum probability of it belonging to any terrain class, $\max_{s \in \mathcal{S}}(P_{i,t}(s))$, exceeds a specified threshold P_{certain}. If this condition is not met, the cell is classified as **uncertain**.

A path $\mathcal{P}_{uv}$ is defined as an ordered sequence of points that connects point $u \in \mathcal{M}_{2D}$ to point $v \in \mathcal{M}_{2D}$. The **path footprint**, denoted as $\Omega(\mathcal{P}_{uv}) \subseteq \hat{\mathcal{M}}_t$, represents the set of grid cells that the robot occupies while traversing along the path $\mathcal{P}_{uv}$.

Definition 4.1.2. (Traversable path) A path $\mathcal{P}_{uv}$ is considered **traversable** if, for every cell $i \in \Omega(\mathcal{P}_{uv})$, the true classification $s_{\text{true}}(i) \neq s_{\text{untrav}}$.

The traversability of a path $\mathcal{P}_{uv}$ can be either certain or uncertain. It is certain if all cells in the path footprint are certain, i.e., $\forall i \in \Omega(\mathcal{P}_{uv}),\ \max_{s \in \mathcal{S}}(P_{i,t}(s)) \geq P_{\text{certain}}$. Otherwise, the path traversability is uncertain.

A **certainly traversable path** given a map estimate $\hat{\mathcal{M}}_t$ is both traversable and certain, meaning all cells $i \in \Omega(\mathcal{P}_{uv})$ are certain and $s_{\text{true}}(i) \neq s_{\text{untrav}}$. The set of certainly traversable paths, $\mathcal{R}(u, \hat{\mathcal{M}}_t)$, includes all paths from u that satisfy these conditions.

The cost of a path $\mathcal{P}_{uv}$ from point u to point v is represented as $C(\mathcal{P}_{uv})$. This cost comprises two main components: the traversability cost, denoted as $\mathcal{T}(\mathcal{P}_{uv})$, and the length of the path, denoted as $\mathcal{L}(\mathcal{P}_{uv})$. The total path cost is a weighted sum of these two components, where α is a parameter that controls the tradeoff between traversability and path length. Hence, the cost of the path $\mathcal{P}_{uv}$ is formulated as:

$$C(\mathcal{P}_{uv}) = \alpha \cdot \mathcal{T}(\mathcal{P}_{uv}) + (1 - \alpha) \cdot \mathcal{L}(\mathcal{P}_{uv}) \tag{4.1}$$

The state of the robot at time t is defined as $z_t = \{r_t, \hat{\mathcal{M}}_t\}$, where $r_t \in \mathcal{M}_{2D}$ represents the x and y components of the robot pose at time t, and $\hat{\mathcal{M}}_t \in \hat{\boldsymbol{\mathcal{M}}}$ is the 2D terrain estimation of our 3D environment at time t. A policy π is defined as a function that takes that current state z_t as input and outputs an action $a \in \mathcal{R}(r_t, \hat{\mathcal{M}}_t)$. Each action corresponds to a certainly feasible path $\mathcal{P}_{r_t r_{t+1}}$ from point r_t to r_{t+1}. Importantly, the probability of reaching state r_{t+1} from r_t following action a is 1. In other words, if action $a = \mathcal{P}_{r_t r_{t+1}}$ is taken, the robot is guaranteed to transition to r_{t+1}. However, the next state z_{t+1} remains probabilistic as the map estimate $\hat{\mathcal{M}}_{t+1}$ at time $t+1$ is uncertain due to the uncertainty of the true terrain classification. The probability transition function $P_a(z, z')$ describes the probability that action a in state z at time t will lead to state z' at time $t+1$: $P_a(z, z') = \mathbb{P}(z_{t+1} = z' \mid z_t = z, a_t = a)$.

The cost function $C_{a_t}(z_t, z_{t+1})$ represents the cost of transitioning from state z_t to state z_{t+1} due to action a_t and is the equal of cost $C(\mathcal{P}_{r_t r_{t+1}})$ introduced in Eq. 4.1. For a given initial state $z_0 = \{r_0, \hat{\mathcal{M}}_0\}$ and a policy π, the total cost of adhering to a policy π is calculated as:

$$J_\pi(z_0) = \mathbb{E}\left[\sum_{k=0}^{N} C_{a_t}(z_t, z_{t+1}) \mid z_0 = \{r_0, \hat{\mathcal{M}}_0\}\right] \tag{4.2}$$

where we choose $a_t = \pi(z_t)$ and the expectation is taken over $z_{t+1} \sim P_{a_t}(z_t, z_{t+1})$. The aim is to find the optimal policy π^*, from the set of all policies Π, that minimizes the expected cost of traversal described in Eq. 4.2. Formally, it can be written as:

$$\pi^* = \arg\min_{\pi \in \Pi} J_\pi(z_0) \tag{4.3}$$

Problem 1. *Given a start point $u \in \mathcal{M}_{2D}$, a goal point $g \in \mathcal{M}_{2D}$, an initial probabilistic map estimate $\hat{\mathcal{M}}_0 \in \hat{\boldsymbol{\mathcal{M}}}$, and an agent equipped with sensors that updates the map estimate $\hat{\mathcal{M}}_t$ at each time step t, determine an optimal policy π^* that minimizes the expected traversal cost $J_\pi(z_0)$, where the initial state is $z_0 = \{u, \hat{\mathcal{M}}_0\}$.*

4.2 Solution Approach

Before introducing our approach, we outline a key assumption that forms the foundation of our methodology. Treating every cell in the environment as uncertain would necessitate considering all possible outcomes for each cell, alongside their associated costs and probabilities. This results in exponential computational complexity, making it infeasible for large environments. Even in simple cases, the state space becomes prohibitively large.

While individual cells exhibit uncertainty, this uncertainty is often spatially correlated with nearby cells due to shared environmental features. Therefore, we assume that uncertainties are not independent but can be grouped into regions, denoted as Q, where each region $q \in Q$ contains cells with dependent uncertainties. This assumption allows us to treat uncertainty disambiguation as a regional event rather than an isolated one. Specifically, if any cell i in region Q_d is observed, then the probability distribution for all other cells $k \in Q_d$ is updated accordingly. Formally, if i is observed at time t, then for all $k \in Q_d$, the following holds:

$$P_{k,t}(s_j) = P_{i,t}(s_j), \quad \forall k \in Q_d \setminus \{i\}, \quad \text{given that } O(i) = 1, \tag{4.4}$$

where $O(i) = 1$ is an indicator variable representing that cell i has been observed, and $s_j \in \mathcal{S}$ is any terrain class in the set of possible terrain classifications. We generate these uncertain regions by performing a preprocessing step on the initial map, allowing us to concentrate computational efforts on the most needed areas. For a more detailed explanation of how these regions are generated, please refer to Section 5.1.1.

4.2.1 Approach Overview

Our approach to solving Problem 1 is divided into three stages, each building upon the previous to create an efficient solution for navigating uncertain large off-road environments. The three stages of our approach are as follows:

1. **Pathway Network Generation:** The first step involves creating a network of paths that forms the foundational structure for the subsequent steps. This network encapsulates the various potential routes between the start and goal points within the environment.

2. **Stochastic Graph Generation:** Based on the created paths, a stochastic graph $\mathcal{G}$ is generated. This graph captures the inherent uncertainties within the environment, providing a framework for the robot's navigation decisions.

3. **Policy Generation:** Finally, given the stochastic graph $\mathcal{G}$, we formulate a Canadian Traveler Problem (CTP) and generate a policy using a modified version of the state-of-the-art CAO* algorithm, which we refer to as Complete CAO* (CCAO*). This policy guides the robot through the environment while handling uncertainties and minimizing the expected traversal cost.

The core of our approach involves generating a topological graph that focuses on the most relevant pathways and uncertain regions likely to be considered during the robot's navigation. Our strategy targets only relevant areas instead of addressing the entire map and its uncertainties. Given an initial uncertain map representation $\hat{\mathcal{M}}_0$, a start point s, a goal point g, and a set of uncertain regions Q, we generate a stochastic graph $\mathcal{G}$.

This graph $\mathcal{G} = (N, E)$ is constructed by converting the uncertain environment into a topological representation, where each node $n \in N$ corresponds to a specific position $w \in \mathcal{M}_{2D}$, representing its location within the environment. Each edge $e = (u, v) \in E$ corresponds to a path $\mathcal{P}_{uv}$ in the environment, with the cost of the edge reflecting the cost of the respective path. The graph nodes are strategically positioned to represent critical decision points in the environment. The generated graph adheres to the following constraints:

1. Includes both start and goal points within the node set N.
2. If a certain traversable path exists between the start and end points in the initial map $\mathcal{M}_0$, at least one such path is represented in the graph.
3. Nodes are not placed in regions that are certainly untraversable.
4. Edges does not represent paths that are certainly untraversable.

Each step is elaborated upon in Sections 4.3, 4.4, and 4.5, respectively. The complete methodology is encapsulated in Algorithm 4, which outlines the overall process of our approach.

Algorithm 4 Uncertainty Aware Motion Planning

Input: $\hat{\mathcal{M}}_0$, s, g, Q
Output: π^*
$\Gamma \leftarrow \text{GenerateNetworkOfPathways}(\hat{\mathcal{M}}_0, s, g, Q)$
$\mathcal{G} \leftarrow \text{GenerateStochasticGraph}(\Gamma)$
$\pi^* \leftarrow \text{CCAO}^*(\mathcal{G}, s, g)$
Return π^*

4.3 Generating A Network of Pathways

This section outlines the process for generating a network of potentially relevant pathways for the robot's navigation in a large, cluttered, and unstructured environment. The key challenge in such environments is the presence of multiple uncertain regions, leading to numerous potential pathways and infinite possible routes to the goal. Our approach focuses on identifying the most relevant paths that facilitate efficient robot navigation from the start point to the goal.

Our approach includes multiple planning iterations within the same environment, always targeting the same goal point but initiating from different start locations. We employ a novel PRM-based multi-query planner to generate the pathway network. The pseudocode for the proposed algorithm is presented in Algorithm 5.

The algorithm we propose works iteratively within a loop, utilizing a stack to manage the sequence of planning requests. It continues to process planning requests until the stack is empty, ensuring that all relevant scenarios are explored (line 6). The first planning request added to the stack assumes that all regions are uncertain, allowing the algorithm to start with the most general case (line 4). This initial request forms the foundation for generating subsequent, more specific requests as the algorithm progresses. Each planning request operates on a distinct map realization (line 8), where both the traversable and uncertain regions are assumed to be traversable while others are treated as untraversable.

For the initial planning request, the PRM* algorithm is utilized to construct a base graph (line 9 to 11), where the graph is generated with the assumption that all uncertain regions are traversable, allowing the base graph to represent the possible traversal through these uncertain regions. The graph created using PRM* in the first planning request is then used as a base graph in subsequent requests. In later requests, we apply the same PRM* algorithm (line 14), but before we run the other planning requests, we update the base graph to accommodate a different map realization than the optimistic one considered in the first planning request (line 13). We modify the base graph by removing nodes and edges associated with untraversable regions in the current realization.

The sampling of new configurations in PRM* employs a hybrid sampling technique [59] that combines uniform random sampling, Gaussian sampling [53] to sample points near obstacles, and narrow passage sampling [54] using a bridge test sampler. The hybrid sampling technique was used as it combines the strengths of multiple sampling distributions to build a roadmap with good coverage and efficiently captures the free space's connectivity. The combination of the sampling strategies is achieved through the weighting of each sampling distribution.



Figure 4.1: Illustrative figure of the disambiguation region and the chosen disambiguation point given an uncertain region and a path

After obtaining the newly computed path, we evaluate whether this path intersects any uncertain regions (line 17). If it does not, the path is guaranteed to be traversable, and no further consideration of alternative realizations sourced from the current one is needed. However, if the path passes through uncertain regions, these regions become critical to consider, as their traversability directly impacts the feasibility of the path.

When a path intersects multiple uncertain regions, the algorithm proceeds to handle each region systematically. The regions are first ordered based on the sequence in which the path passes through them. For each uncertain region, the algorithm generates new planning requests by making specific assumptions: it assumes that the current region (the ith region in the sequence) is untraversable, all previous regions (from 1 to $i-1$) are traversable, and the remaining regions are ambiguous (lines 21 to 23). The logic behind this approach is that in order to reach region i, the robot must have successfully traversed all prior regions, making this realization relevant for consideration.

For each of these scenarios, two new planning requests are generated and added to the stack. The first request involves re-planning from the start point with the updated assumptions, and the second request re-plans from the first disambiguation point along the path where the robot can observe and resolve the uncertainty for the ith region. A disambiguation point for region q is simply a point $u \in R(q)$. As shown in Figure 4.1, the uncertain region R2 is highlighted in red, while its disambiguation region is represented in blue. The disambiguation point is selected as the first location on the path where the robot can observe R2. At this point, the robot's sensors can gather sufficient information

to determine whether R2 is traversable or should be avoided.

To further optimize this process, we implement a pruning mechanism before adding any new planning request to the stack. The *PruneRequest* function (line 27) checks whether a given planning request is redundant. Specifically, it determines if the path corresponding to this request has already been computed or if it duplicates efforts from previous requests. This check is crucial as it prevents the algorithm from re-exploring already evaluated paths, thereby saving computational resources and time. If a request is identified as redundant, it is pruned and not added to the stack, ensuring that only unique requests are added to the stack. If they are not pruned, these requests are added to the *pathCache* and then pushed onto the stack, allowing the algorithm to explore these alternative paths in subsequent iterations (lines 28 and 29). To prevent the base graph from growing uncontrollably, we selectively update it by only adding new edges that are used in the path, thereby preserving computational effort while maintaining manageability across planning requests (line 35).

In summary, the process of generating a network of pathways, as outlined above, leverages PRM* algorithm to explore the vast search space within uncertain environments efficiently. By iteratively refining the assumptions about uncertain regions and dynamically updating the base graph, the algorithm ensures that only the most relevant and feasible pathways are considered. The integration of the pruning mechanism further optimizes the process by eliminating redundant requests and focusing computational resources on exploring unique routes. The resulting list of paths serves as the basis for the next step, which involves generating a stochastic graph representation of the environment.

4.4 Generating A Stochastic Graph

After generating the network of connected paths, we aim to generate a stochastic graph $\mathcal{G}$. Let $\mathcal{G} = (N, E)$ be a weighted undirected graph, where N denotes the set of nodes with cardinality $|N|$, and E denotes the set of edges with cardinality $|E|$. Each edge $e \in E$ is associated with a non-negative cost $c(e) \in \mathbb{R}_{\geq 0}$, indicating the cost of traversal, and a blocking probability $p(e) \in [0, 1]$ that represents the probability that e is not traversable.

The method processes the list of paths generated in the first step, adding nodes at merge points between paths and at disambiguation points. Edges between nodes $u, v \in N$ represent the paths connecting points u and v. Edges with $p(e) = 0$ correspond to paths that do not pass through any uncertain region. In contrast, uncertain edges with $0 < p(e) < 1$ represent paths from a disambiguation point within an uncertain region to the first point on the path outside that region. For these edges, the probability $p(e)$ equals the probability

Algorithm 5 Generating Network Of Pathways Algorithm

Input: $\hat{\mathcal{M}}_0$, s, g, Q
Output: Γ:
FirstRun $\leftarrow$ True
Stack $\leftarrow \{(s, Q, \varnothing, \varnothing)\}$
pathCache $\leftarrow \varnothing$
while Stack $\neq \varnothing$ **do**
    $(s', Q'_A, Q'_U, Q'_T) \leftarrow$ Pop(Stack)
    $M' \leftarrow$ GetRealization$(Q'_A \cup Q'_T, Q'_U)$
    if FirstRun **then**
        $\mathcal{P}_{s'g}, G_{base} \leftarrow$ PRM*(s', g, M')
        FirstRun $\leftarrow$ False
    else
        $G \leftarrow$ UpdateGraphs(G_{base}, Q'_U)
        $\mathcal{P}_{s'g} \leftarrow$ PRM*(G, s', g, M')
    end if
    $\Gamma \leftarrow \Gamma \cup \mathcal{P}_{s'g}$
    if Intersects$(\mathcal{P}_{s'g}, Q'_A)$ **then**
        $Q_{crit} \leftarrow$ IdentifyCriticalRegions$(\mathcal{P}_{s'g}, Q'_A)$
        $Q_{prev} \leftarrow \varnothing$
        for each $(i, q) \in$ enumerate(Q_{crit}) **do**
            $Q''_T \leftarrow Q'_T \cup Q_{prev}$
            $Q''_U \leftarrow Q'_U \cup \{q\}$
            $Q''_A \leftarrow Q'_A \setminus (Q_{prev} \cup \{q\})$
            $u \leftarrow$ FindDisambiguationPoint$(\mathcal{P}_{s'g}, q)$
            for each v in $\{u, s\}$ **do**
                NewRequest $\leftarrow (v, Q''_A, Q''_U, Q''_T)$
                if not PruneRequest(NewRequest) **then**
                    Push(pathCache, NewRequest)
                    Push(Stack, NewRequest)
                end if
            end for
            $Q_{prev} \leftarrow Q_{prev} \cup \{q\}$
        end for
    end if
    $G_{base} \leftarrow$ UpdateBaseGraph$(\mathcal{P}_{s'g})$
end while
Return Γ

that the uncertain region they pass through is classified as untraversable. The developed algorithm iterates through each path, adding nodes and edges accordingly, and returns the constructed graph $\mathcal{G}$.

The cost of a path is calculated based on the formulation in Equation (4.1), where α is chosen empirically to balance traversability and path length. The traversability cost for a path segment is calculated by summing the costs of each cell in the path's footprint. The cost of a path is calculated given the following equation:

$$C(\mathcal{P}_{ij}) = \alpha \cdot \left(\sum_{k \in \Omega(\mathcal{P}_{ij})} C(k) \right) + (1 - \alpha) \cdot L(\mathcal{P}_{ij}) \tag{4.5}$$

where $C(k)$ represents the cost for cell k. For certain paths, $C(k)$ is simply the traversal difficulty $D(s_j)$ based on the known terrain class $s_j \in \mathcal{S}$. However, this cost is not applicable if the path passes through uncertain cells as the cost of cell k, $C(k)$, is now a random variable instead. Let $I(i)$ be an indicator random variable and is introduced for each cell i, where $I(i) = 1$ if the cell is traversable and $I(i) = 0$ for untraversable cells. Let $C(k)$ be a random variable that indicates the estimated cost for traversing uncertain cell k. The expected cost for traversing a cell given that it is traversable is then given by:

$$\mathbb{E}[C(k)|I(k) = 1] = \frac{\sum_{j=1}^{n} P_i(s_j) \cdot D(s_j)}{1 - P_k(s_{\text{untrav}})} \tag{4.6}$$

Hence, the expected cost of an uncertain path $\mathcal{P}_{ij}$ is calculated as follows:

$$C(\mathcal{P}_{ij}) = \alpha \cdot \left(\sum_{k \in \Omega(\mathcal{P}_{ij})} \mathbb{E}[C(k)|I(k) = 1] \right) + (1 - \alpha) \cdot L(\mathcal{P}_{ij}) \tag{4.7}$$

This formulation ensures that the cost of both the deterministic and stochastic regions in the environment are accurately captured. The generated graph represents a comprehensive integration of the pathways identified in Step 1. It achieves this by adding nodes at locations where multiple pathways intersect, allowing it to connect segments from different paths seamlessly and, hence, present new paths that were not explicitly generated initially. This approach not only enhances the topological representation of the environment but also optimizes the computational time needed to generate such a graph by reducing the number of paths required to represent the possible routes effectively.

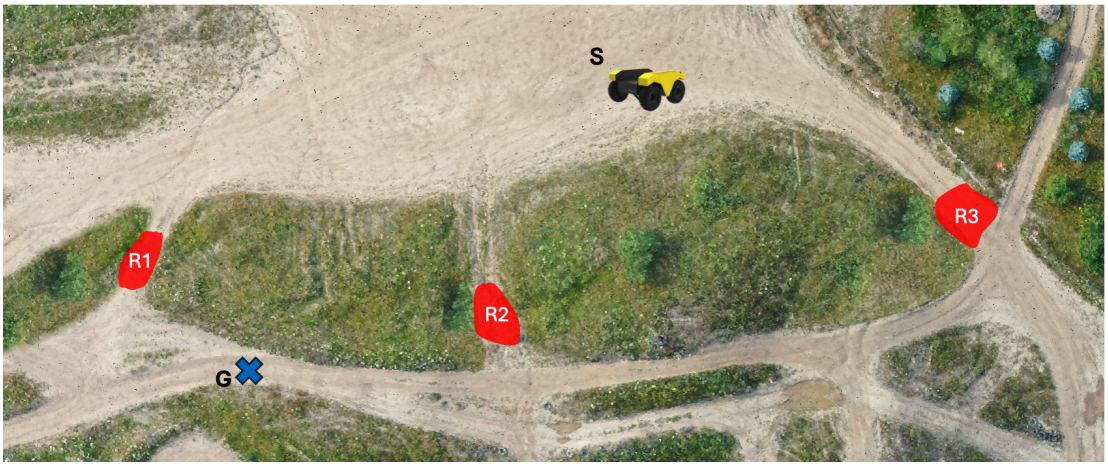


Figure 4.2: Illustrative figure showing a scenario where there is no determinisitc path between start and goal points

4.5 Generating A Policy

In this section, we introduced the CCAO* algorithm, an extension of the state-of-the-art CAO* algorithm that addresses the issue of algorithm completeness. In the original CAO*, a solution is returned only when a deterministic path, which can be prohibitively expensive, exists to the goal. Otherwise, the algorithm fails to provide a navigation policy. Furthermore, in CTP literature, AO* and its variants are usually tested on graphs that have a deterministic path to the goal in order to avoid infinite costs [34], [37]. However, the absence of a deterministic path does not necessarily imply the impossibility of reaching the goal.

To address the limitation of returning no solution in such cases, we made several key modifications to the original CAO* algorithm. The pseudocode for CCAO* is presented in Algorithm 7. In the original CAO*, an infinite cost was used to indicate that no path existed from a node to the goal. However, this use of infinity caused the value to propagate through the tree during the backpropagation step, corrupting the cost estimation by labeling entire subtrees with infinite costs and leading to the failure to generate valid solutions. To address this issue, we replaced the infinite cost with a *NoPathPenalty* cost—a finite value set higher than any feasible path cost. This approach discourages selecting these paths without disrupting cost propagation, enabling accurate backpropagation and proper node label updates.

Additionally, If no path to the goal is found after disambiguating all critical regions

that might have a path to the goal, we introduce the option to return to the starting point but with a substantial penalty. This penalty is designed to be more severe than any other possible path from start to goal if one exists. This ensures that a solution can be found even when a deterministic path is not guaranteed. The core of this extension lies in modifying the *getFront* method used to identify the non-terminal leaf node with the lowest heuristic value when expanding the tree. In the original *getFront* method, the algorithm starts at the root node and traverses down the tree by selecting the most promising valid child. A valid child is a node that has not yet been solved and has a path to the goal. The algorithm continues this process until it reaches a leaf node. However, if the current most promising node has children but none are valid, the algorithm terminates with no solution. This approach relies on the assumption that a deterministic path exists, which limits its ability to handle situations where such a path cannot be found.

In the modified method, shown in Algorithm 6, if the current most promising node has children but none of them are valid, i.e. the most promising unsolved child doesn't have a path to the goal, the method still extends this child by connecting it back to the start node. This triggers a return to the start with the added *ReturnToStartPenalty*, a penalty set to be more severe than any feasible path to the goal.

By implementing these modifications, our algorithm will still output a policy even if there is no deterministic path to the goal. This policy will exhaust all possible disambiguation options before guaranteeing that there is no path to the goal and, hence, returns to the start.

Algorithm 6 getFront (New)

```
1: n ← root
2: while n has successors do
3:     s ← getMinCostUnsolved(n)
4:     if s is None then
5:         s ← unsolvedChild(n)
6:         Return s, False
7:     end if
8:     n ← s
9: end while
10: Return n, True
```

Algorithm 7 CCAO* Algorithm

Input: $\mathcal{G} = (N, E)$, s, g
Output: π^*
$T \leftarrow$ InitializeTree(s)
while s is not terminal **do**
    n_e, expandable $\leftarrow$ getFront(T)
    if not expandable **then**
        $n_s \leftarrow$ connectToStart(T, n_e)
    else
        $n_s \leftarrow$ nextStates(T, n_e)
    end if
    for each $n \in n_s$ **do**
        if n is not redundant **then**
            AddNewNodeToTree(T, n_e, n)
        end if
    end for
    backpropagate(T, n_e)
end while
$\pi^* \leftarrow$ ExtractOptimalPolicy(T, s)
Return π^*

Chapter 5

Simulation Environment and Performance Evaluation

5.1 Simulation Environment

In this study, we conducted our experiments in a custom-built simulation environment. The vehicle used in the simulation can be bounded by a sphere with a radius of 1.0 meters, which adequately represents the dimensions of an off-road vehicle. For testing purposes, we simulated an off-road terrain testing site in Michigan, USA. The testing site covers an area of 1000 by 325 meters. The resolution of the gridmap used in the simulation is 0.5 meters per cell. This resolution was chosen because it is sufficient to accurately represent the terrain features of the environment while avoiding the creation of a huge gridmap representation. As a result, the gridmap consists of 2000 by 650 cells, balancing detail with computational efficiency. A top-view image of the testing site is shown in Figure 5.2, illustrating the scale and complexity of the terrain the vehicle needs to navigate.



Figure 5.1: Aerial view of the Michigan off-Road testing site used for policy evaluation

5.1.1 Generating Traversability Cost Map

In autonomous off-road navigation, the ability to assess the traversability of different terrains is critical for safe and efficient path planning. The traversability cost map is a key component in this process, combining geometric and semantic information to comprehensively assess the environment. In this section, we describe how the map $\mathcal{M}$ is processed to generate a detailed traversability analysis.

Semantic Classification

The semantic classification of the map is manually annotated into distinct terrain classes. These classes include Background, Asphalt, Bushes, Obstacle, Rock, Trees, Water, Sand, and Grass, as illustrated in Figure 5.2. This classification allows us to distinguish between different types of terrain based on their physical characteristics and potential impact on navigation.

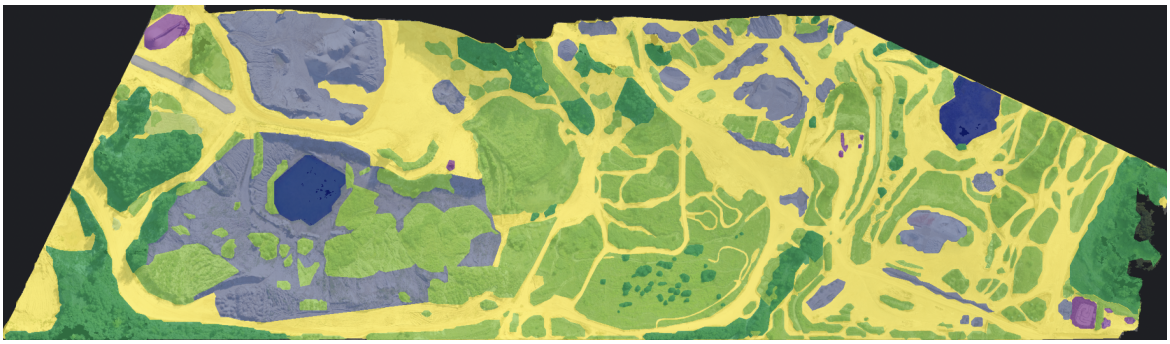


Figure 5.2: Ground truth semantic labels: Background, Asphalt, Bushes, Obstacle, Rock, Trees, Water, Sand, Grass

Geometric Features Extraction

The terrain within the environment is modeled using Digital Elevation Models (DEMs) derived from point cloud data of the Michigan testing site. This allows the simulation to accurately replicate the elevation changes and surface irregularities present in the real-world location. As shown in Figure 5.3, the DEM reveals a range of elevations across the environment, which are crucial for further geometric analysis.

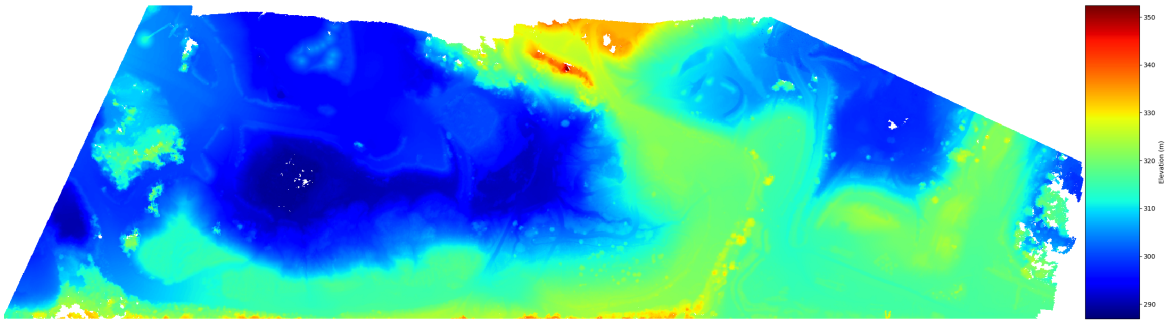


Figure 5.3: Digital Elevation Map (DEM) showing elevation variations in the environment.

The slope and roughness of the terrain are derived from the DEM, as shown in Figures 5.4 and 5.5. The slope represents the gradient of each pixel relative to its neighbors, providing an indication of how steep the terrain is [60]. Roughness measures the cumulative differences in elevation between a central pixel and its surrounding pixels, giving an indication of how uneven the terrain is [61]. These features are calculated using the xDEM open-source Python library [62].

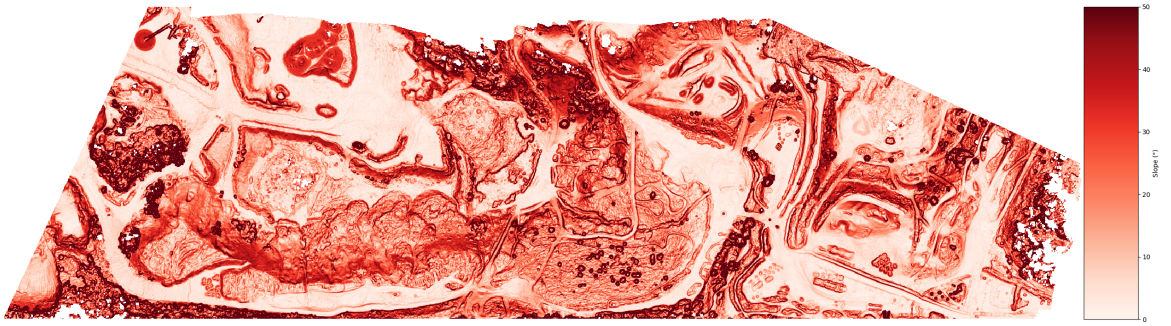


Figure 5.4: Terrain slope derived from the digital elevation map.

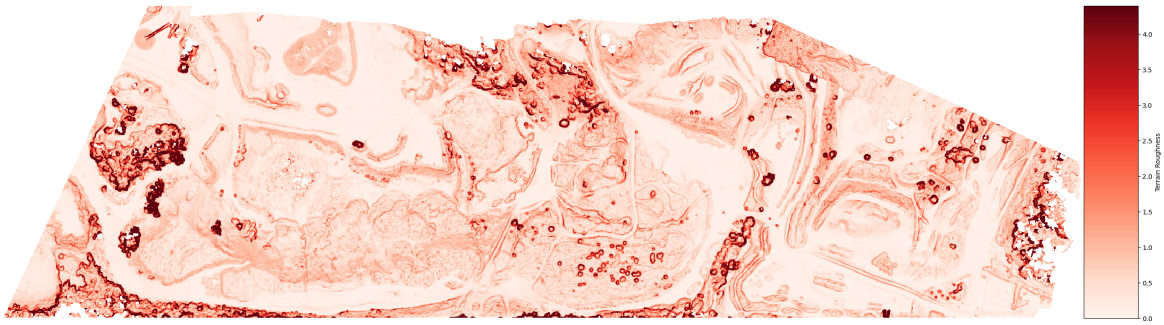


Figure 5.5: Terrain roughness derived from the digital elevation map.

Geometric Classification

The terrain classification of each cell i within the gridmap $\hat{\mathcal{M}}_0$ is determined based on its slope θ_i and roughness ρ_i. The classification assigns each cell i to one of the following classes: free (s_1), low-cost (s_2), medium-cost (s_3), or obstacle (s_{untrav}). This classification is defined as:

$$s(i) = \begin{cases} s_{\text{untrav}} & \text{if } \theta_i \geq \theta_{\text{high}} \text{ or } \rho_i \geq \rho_{\text{high}}, \\ s_3 & \text{if } \theta_{\text{mid}} \leq \theta_i < \theta_{\text{high}} \text{ or } \rho_{\text{mid}} \leq \rho_i < \rho_{\text{high}}, \\ s_2 & \text{if } \theta_{\text{low}} \leq \theta_i < \theta_{\text{mid}} \text{ or } \rho_{\text{low}} \leq \rho_i < \rho_{\text{mid}}, \\ s_1 & \text{if } \theta_i < \theta_{\text{low}} \text{ and } \rho_i < \rho_{\text{low}}. \end{cases} \tag{5.1}$$

In this equation, θ_i represents the slope, and ρ_i represents the roughness of terrain at cell i. The slope and roughness thresholds θ_{low}, θ_{mid}, θ_{high} and ρ_{low}, ρ_{mid}, ρ_{high} define the boundaries between different terrain classes. Cells are classified as free (s_1) if both slope and roughness are low, as low-cost (s_2) if either slope or roughness is moderate, as medium-cost (s_3) if either slope or roughness is high, and as obstacles (s_{untrav}) if either slope or roughness exceeds the highest thresholds. The geometric classification is visualized in Figure 5.6, which categorizes the terrain based on the defined thresholds.

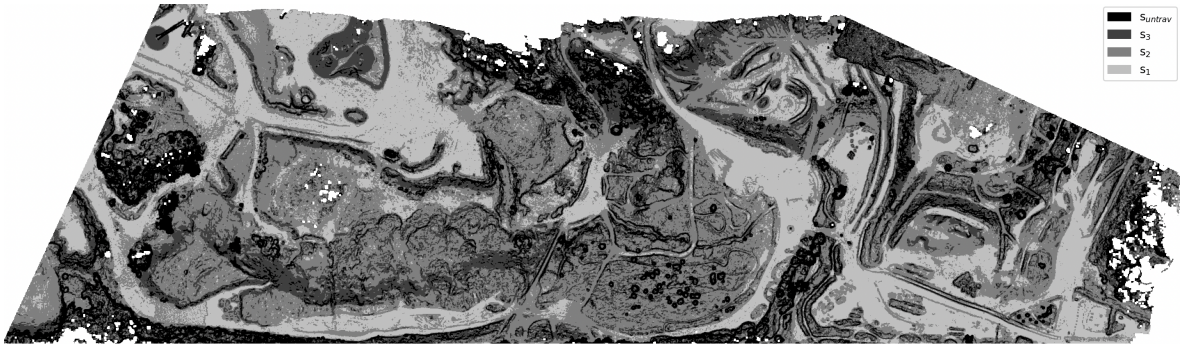


Figure 5.6: Geometric classification map based on slope and roughness analysis.

Final Traversability Map

The final traversability map, shown in Figure 5.7, combines these geometric labels with the semantic information. If a cell's semantic label indicates an untraversable feature (e.g., water or dense vegetation), the cell is marked as untraversable, overriding the geometric classification. Otherwise, the geometric classification is retained, following the approach used in [12]. This integrated map provides a robust basis for navigation planning in complex environments.

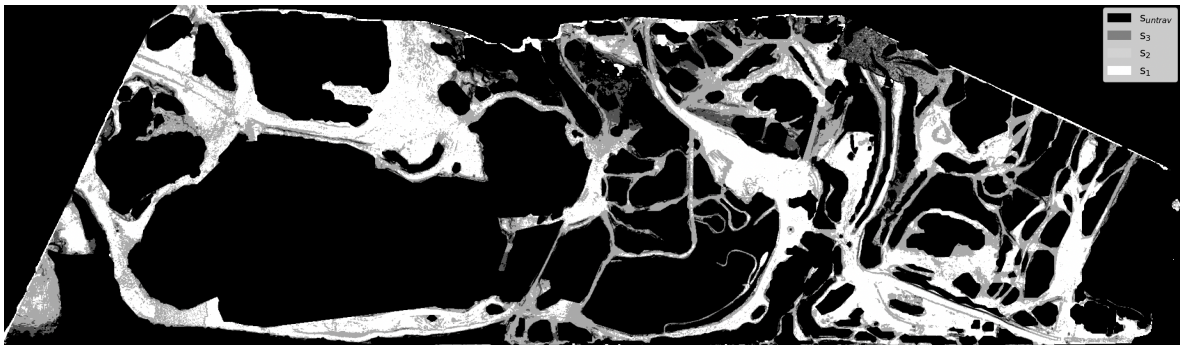


Figure 5.7: Final traversability map combining geometric and semantic classifications.

5.1.2 Uncertain Region Extraction

Instead of assigning uncertainty values to individual pixels, we use unsupervised clustering to group grids with similar visual and spatial features. Specifically, we employ the SLIC

(Simple Linear Iterative Clustering) algorithm [63], which performs K-means clustering in a 5D space that includes color and spatial location. This approach captures regional similarities, leading to more meaningful, context-aware clusters.

Figure 5.8 shows the result of this clustering, where visually similar grids are grouped into distinct clusters. This reduces the complexity of managing pixel-wise uncertainty and ensures consistent treatment of similar regions. Some of the regions created during the clustering process are then designated as uncertain regions. This designation is due to factors such as obstructions, ambiguous terrain types, or unclear boundaries resulting from limitations in the gridmap resolution used. In this work, each uncertain region q is assigned an arbitrary probability of being untraversable, denoted $p(q)$, as this aspect is not the focus of our study. However, the probability distribution of possible terrain classes could be obtained as a byproduct of the region-based semantic classification process, as detailed in [64].

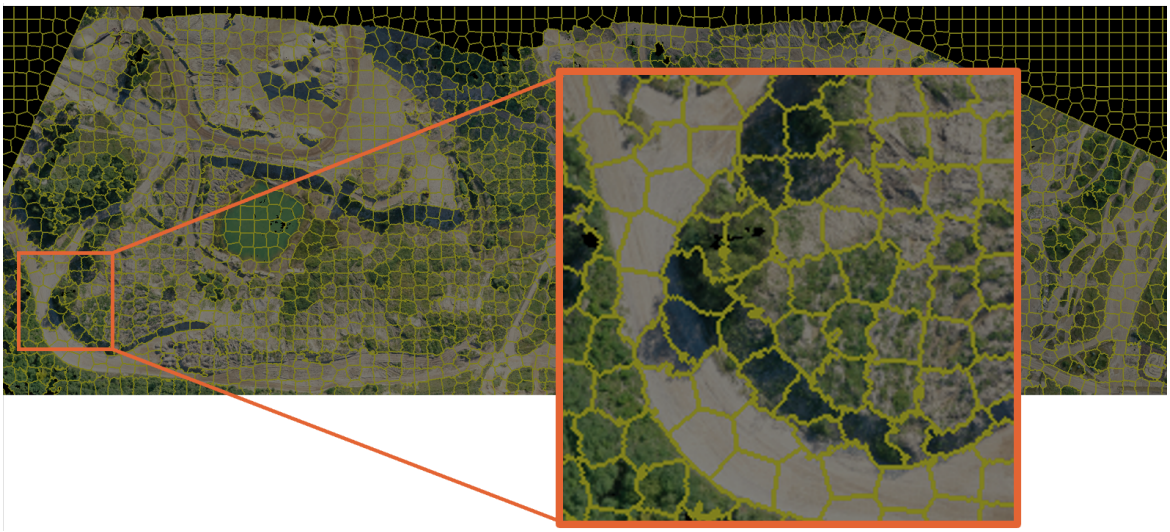


Figure 5.8: Unsupervised superpixel segmentation of the environment.

5.2 Baselines

We evaluate our algorithm by comparing it against three baseline approaches. The first baseline is a greedy method that initially assumes all regions are fully traversable and plans an optimistic path to the goal. As the robot progresses along the path, it updates its map

based on sensor feedback. Upon encountering uncertain regions, it performs disambiguation, and if a region is found to be untraversable, the robot replans a new path from the point of disambiguation to the goal.

The second baseline we compare against is a more conservative planner that assumes all uncertain regions are untraversable from the beginning. This planner takes a cautious approach by avoiding any regions labeled as uncertain, planning its path solely through regions that are confirmed to be traversable. This ensures that the robot avoids potential obstacles, but it may result in longer paths or even failure to find a feasible path if there is no deterministic path to the goal.

Finally, we compare against formulating the problem as D-SOSP, a grid graph discretization of continuous-space SOSP. A graph representation of the environment is created by positioning a node at the center of each grid cell, where they are interconnected by edges that extend in eight directions—to adjacent and diagonal nodes. When these edges intersect the boundaries of the uncertain regions, they become stochastic, with their probability of being traversable determined by the probability of the region being traversable. Otherwise, the edges are considered deterministic. Similar to our approach, the same CAO* algorithm is applied to the generated stochastic graph, with the goal of developing a policy that minimizes the expected length of the traversal by effective exploitation of the disambiguation capability. D-SOSP is essentially a variant of CTP on grid graphs with probabilistic dependency among groups of edges. A figure of the D-SOSP on our testing site is shown in figure 5.9.

To further assess the effectiveness of grid graph approach, we evaluate it across different grid resolutions, analyzing the trade-off between computation time and path optimality. By varying the grid resolution, we can observe how finer or coarser discretizations impact the performance of the algorithm, particularly in terms of achieving optimal paths while managing computational resources. This comparison sheds light on the scalability of the existing approaches in the literature.

5.3 Simulation Scenarios and Policy Evaluation

In this section, we present and analyze the results of our uncertain-aware path planning algorithm across three distinct simulation scenarios, each designed to evaluate the algorithm's performance under varying conditions of environmental complexity and uncertainty.

Simulation Scenario A focuses on a small section of the environment, where the start and goal points are placed such that a deterministic path exists between them. This

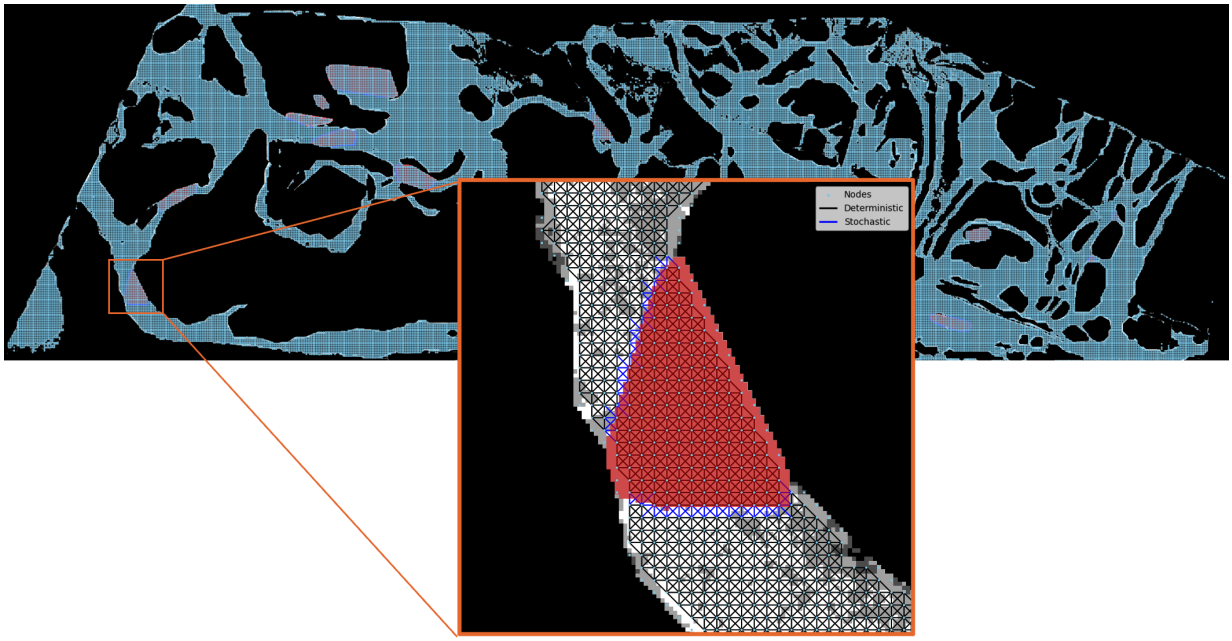


Figure 5.9: Gridgraph representation of the environment

scenario serves as a preliminary test to observe how the algorithm manages uncertainty within a limited, well-defined region. Simulation Scenario B also focuses on the same small section of the environment, but the uncertain regions are positioned such that no deterministic path exists between the start and goal points. This scenario challenges the algorithm to navigate through uncertain regions and adapt its planning strategy in the absence of a clear, deterministic route, testing the algorithm's completeness in a compact area. Simulation Scenario C extends the analysis to the entire environment. This scenario is designed to test the scalability of the algorithm, assessing how well it performs in a larger, more complex environment.

In all scenarios, we compare our algorithm's performance against three baselines: the greedy optimistic approach, the conservative pessimistic planner, and the D-SOSP method, which is a grid graph discretization of the continuous-space Stochastic Obstacle Scene Problem (SOSP). By examining these different strategies, we aim to demonstrate the advantages and trade-offs of our approach in navigating uncertain, unstructured, uneven off-road environments.

5.3.1 Simulation Scenario A

In Simulation Scenario A, we focus on a small section of the environment, which spans 125 by 54 meters and is represented by a grid of 250 by 108 cells. Within this environment, there are two uncertain regions, R1 and R2, with associated blocking probabilities. Specifically, the blocking probability for R2 is 0.4, meaning there is an 40% chance that R2 is untraversable, while for R1, the blocking probability is 0.75, indicating a 75% chance of it being untraversable. The start point S and goal point G are positioned such that a deterministic path is available between them.

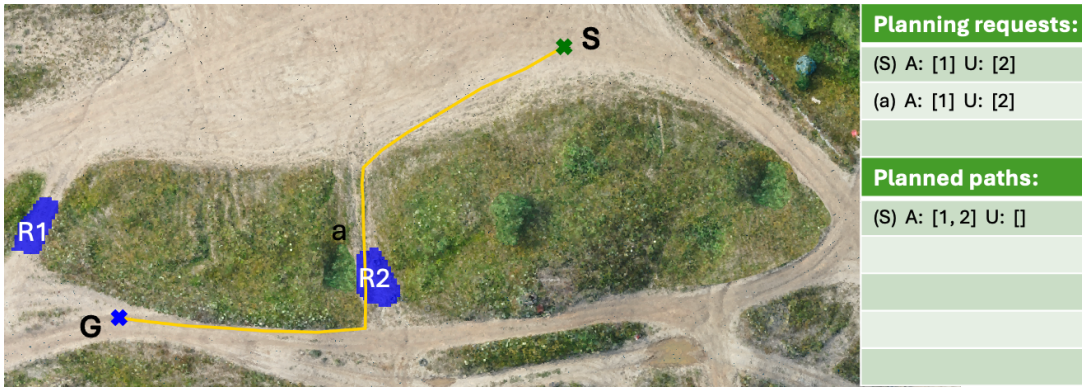
Generation of relevant pathways

The six images in Figure 5.10 illustrate an iterative methodology for generating a network of pathways in an unstructured environment while considering uncertain regions (R1, R2) as shown in Figure 5.10a.

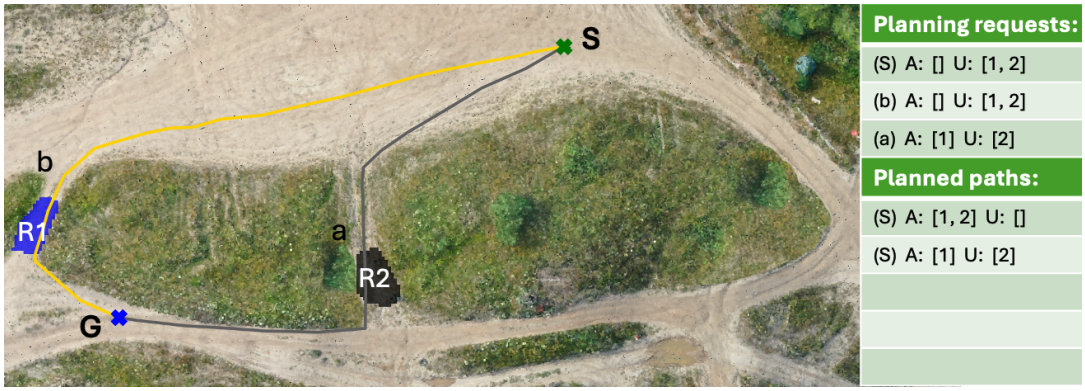
The process begins with the initial planning request starting at S made with the assumption that both uncertain regions are traversable. This request is processed, resulting in the path shown in Figure 5.10b. However, because this path passes through uncertain region R2, two new planning requests are added to the stack: one from the start point and another from the disambiguation point of R2 denoted as a, both targeting the goal point G, under the assumption that R2 is untraversable.



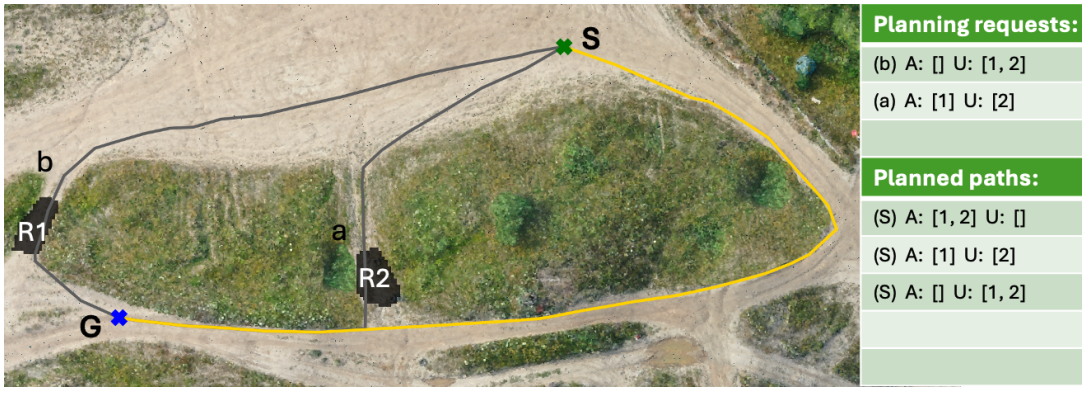
(a)


S
R1
a
R2
G
Planning requests:
(S) A: [1] U: [2]
(a) A: [1] U: [2]
Planned paths:
(S) A: [1, 2] U: []

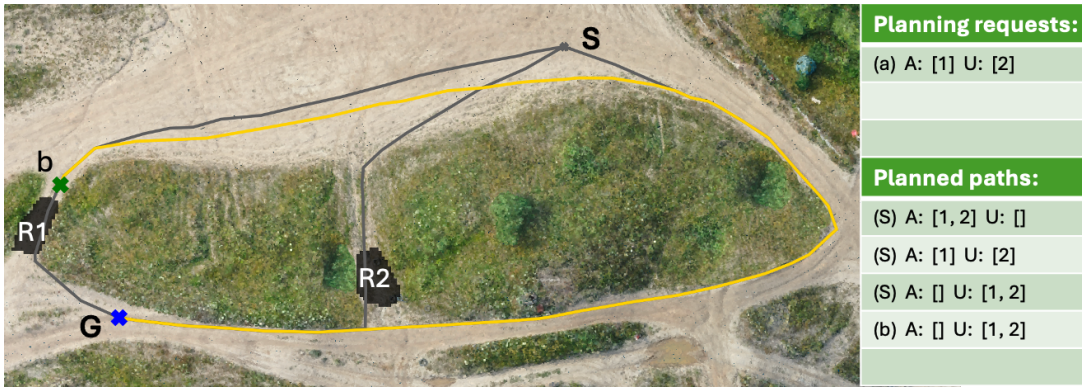
(b)


S
b
R1
a
R2
G
Planning requests:
(S) A: [] U: [1, 2]
(b) A: [] U: [1, 2]
(a) A: [1] U: [2]
Planned paths:
(S) A: [1, 2] U: []
(S) A: [1] U: [2]

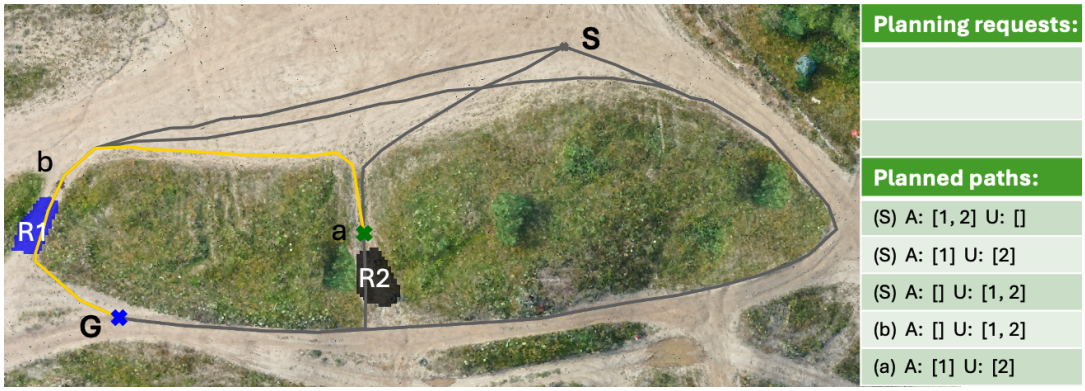
(c)


S
b
R1
a
R2
G
Planning requests:
(b) A: [] U: [1, 2]
(a) A: [1] U: [2]
Planned paths:
(S) A: [1, 2] U: []
(S) A: [1] U: [2]
(S) A: [] U: [1, 2]

(d)



(e)



(f)

Figure 5.10: Illustration of our approach in generating network of relevant pathways

As the planning process continues, requests are iteratively popped from the stack and planned. If any planned path intersects another uncertain region, additional planning requests are again pushed onto the stack. The process continues, and new planning requests are handled, and others are added to the *pathCache* as shown in Figures 5.10c, 5.10d, and 5.10e. In figure 5.10f, the last remaining planning request in the stack is performed, and it plans a path starting at a with the realization that R2 is untraversable while R1 is traversable. The path generated passes through the uncertain region R1, which requires adding two new planning requests, ((S) A: [] U:[1, 2]) and ((b) A: [] U: [1, 2]), however, as these two requests are already stored in the *pathCache*, they are pruned and not added to the planning request stack, which leaves the stack empty and hence the termination of the algorithm.

The outcome of this iterative planning process is a network of pathways that effectively navigates the small environment while accounting for the uncertainties present in regions R1 and R2. This network, illustrated in Figure 5.11, shows the five generated paths: three originating from S, one from a, and one from b, all targeting G. Each of these paths highlights an important route in this environment. Out of the infinite possible paths, our proposed method selects these to represent the environment. With this step completed, the network of pathways will now be used in the next phase of the process, which generates the stochastic graph representation of the environment.

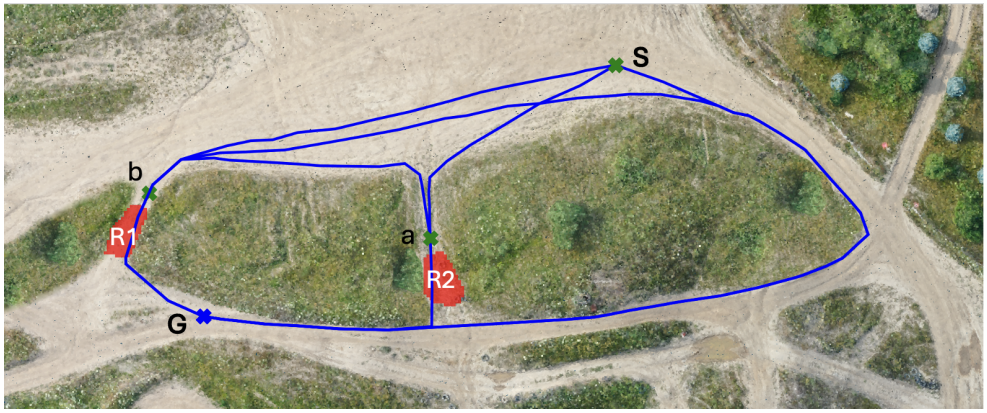


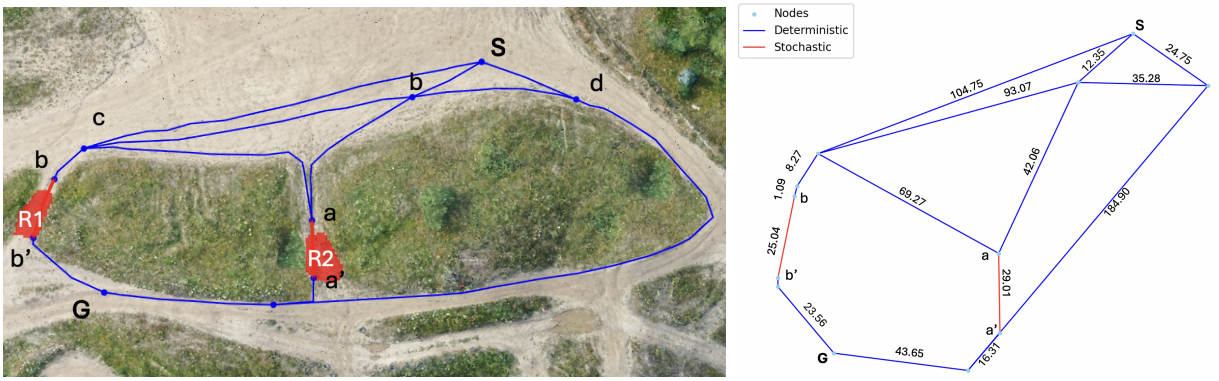
Figure 5.11: Generated network of pathways for scenario A

Stochastic graph generation

After establishing the network of connected pathways in the environment, the next step is to generate a stochastic graph $\mathcal{G}$. Figures 5.12a and 5.12b illustrate the stochastic graph generated for Scenario A. Figure 5.12a shows the spatial layout of the graph overlaid on the environment, highlighting the added nodes as well as the deterministic and stochastic edges. Figure 5.12b offers a simplified view of the graph structure, showing the cost associated with each edge.

The construction of $\mathcal{G}$ involves analyzing the previously generated paths and identifying key nodes at critical junctions. These include merge points between different paths (such as nodes b, c, and d), disambiguation points within uncertain regions (nodes a and b), and exit points from those regions (nodes a' and b'). Edges are then established between these nodes, representing the possible routes that the vehicle can follow.

By strategically placing nodes at these locations, we can introduce new paths that were not explicitly generated in the initial pathway network. For instance, in the graph,



(a) (b)

Figure 5.12: Stochastic graph representation of small environment

a deterministic path from a to G can be followed by the sequence $a \to b \to d \to G$, even though this specific path was not part of the initially generated paths in Step 1. This approach not only enriches the topological representation of the environment but also optimizes the computational effort required to generate the graph by reducing the number of pathways needed to effectively represent all possible routes. As a result, we achieve a more efficient and comprehensive model of the environment's navigable paths, balancing detail with computational efficiency.

Table 5.1 presents the results of the graph generation for Scenario A, comparing our method against five grid-based approaches, referred to as GridGraph 1 to 5. These grid-based methods differ in the step size, which is the number of grid cells between one node and another. The comparison focuses on the number of nodes, the number of deterministic and stochastic edges, and the time taken to generate the graph.

As the step size increases from 1 to 5, the number of nodes and edges in the resulting graph decreases, leading to reduced computational time for graph generation. This is evident in the table, where GridGraph 1, with the smallest step size, results in the largest graph with 12,689 nodes and 41,131 deterministic edges, taking 1.4118 seconds to generate. On the other end of the spectrum, GridGraph 5, with the largest step size, results in a significantly smaller graph with 518 nodes and 1,334 deterministic edges and the shortest generation time of 0.1957 seconds. In contrast, our method, while taking the longest time to generate the stochastic graph at 2.9532 seconds, achieves this with a drastically reduced number of nodes at 10 and edges at 10 deterministic and 2 stochastic. This reduction in the graph size is crucial for managing the computational load in subsequent steps.

Table 5.1: Graph generation results for Scenario A

Method	# Nodes	# Deterministic Edges	# Stochastic Edges	Graph generation time (s)
GridGraph(1)	12,689	41,131	82	1.4118
GridGraph(2)	3,222	9,698	37	0.5365
GridGraph(3)	1,443	4,077	24	0.3507
GridGraph(4)	815	2,148	14	0.2374
GridGraph(5)	518	1,334	11	0.1957
Our approach	10	10	2	2.9532

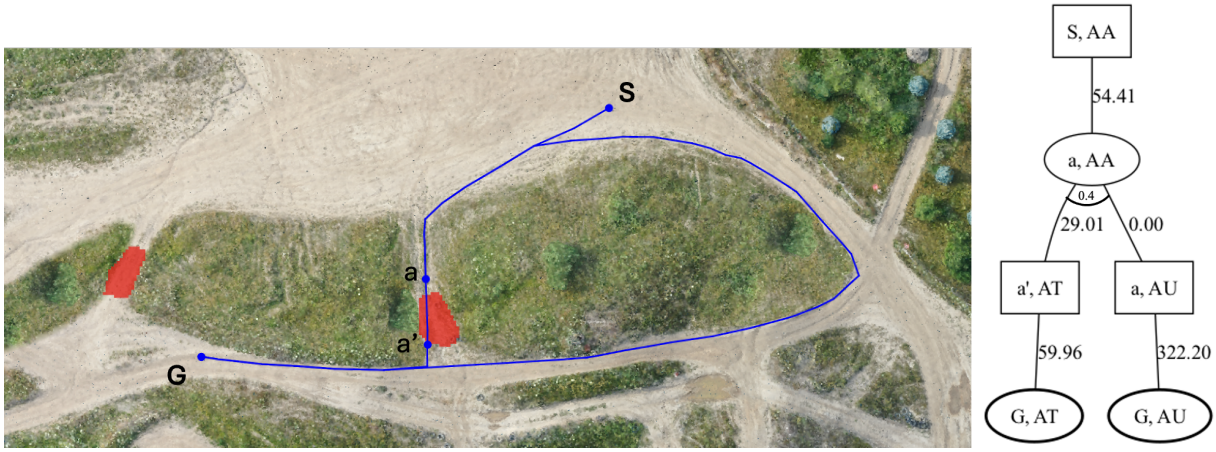
Policy generation

In this final step of our approach, we generate policies that guide the vehicle from the start point S to the goal point G, while minimizing the expected cost. We compare our method against the optimistic and conservative policies, as well as the grid-based approach with five different step sizes. The policies are evaluated based on expected cost, best-case cost (when all regions are traversable), worst-case cost (when all regions are untraversable), the computation time for policy generation, and the computation time for the entire approach.

Figure 5.13a shows the spatial layout of the policy generated by our method, visualizing the algorithm's navigation through uncertain regions R1 and R2 within the actual environment. Figure 5.13b displays the output of the CAO* algorithm, which is a solution subtree from the AO tree. In this graph, the circular nodes represent AND nodes, which are either disambiguation points, such as point a, or terminal nodes leading to the goal, such as G, which is also bolded to indicate its status as a goal node. Each edge in the graph is associated with a traversal cost and, where applicable, a probability that represents the outcome of a disambiguation.

Our policy chooses to first disambiguate R2, and if it is found to be untraversable, the robot will bypass R1 entirely and proceed directly to the goal. It avoids disambiguating R1 in this case because if R1 also turns out to be untraversable, the robot would have to take a significantly longer detour. The probability of regions being untraversable plays a crucial role in determining which regions are disambiguated. Since R1 has a 75% chance of being blocked, it is not worth the risk. Depending on the likelihood of each region being traversable and the associated path costs for both traversable and untraversable cases, the algorithm selects the path that minimizes the expected cost.

The optimistic policy, shown in Figure 5.14, ignores the probabilities of untraversability



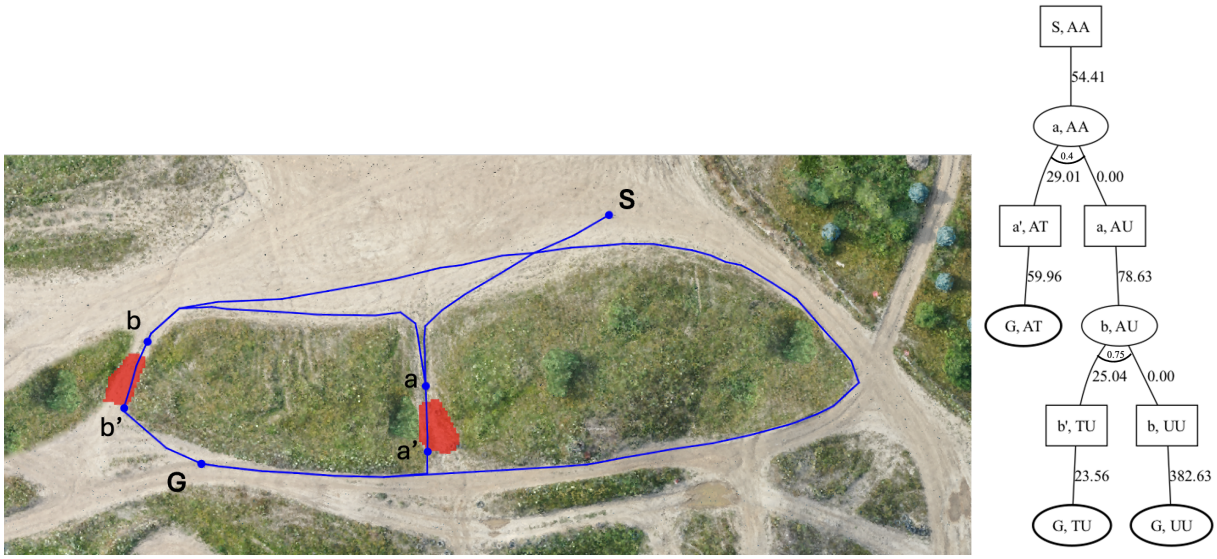
(a) (b)

Figure 5.13: (Left) Spatial layout of the proposed policy. (Right) Solution subtree generated by the CCAO* algorithm.

and directly tries to reach G through R2, the shortest path. If R2 is untraversable, it then attempts R1 and finally the last available path. This approach risks higher costs if the initially chosen uncertain region turns out to be untraversable. The conservative policy, on the other hand, avoids any uncertain regions entirely, as shown in Figure 5.15. It directs the vehicle along a path that does not pass through any uncertain regions, thus ensuring a deterministic route to the goal but at the cost of a higher expected traversal cost due to the longer and potentially suboptimal path.

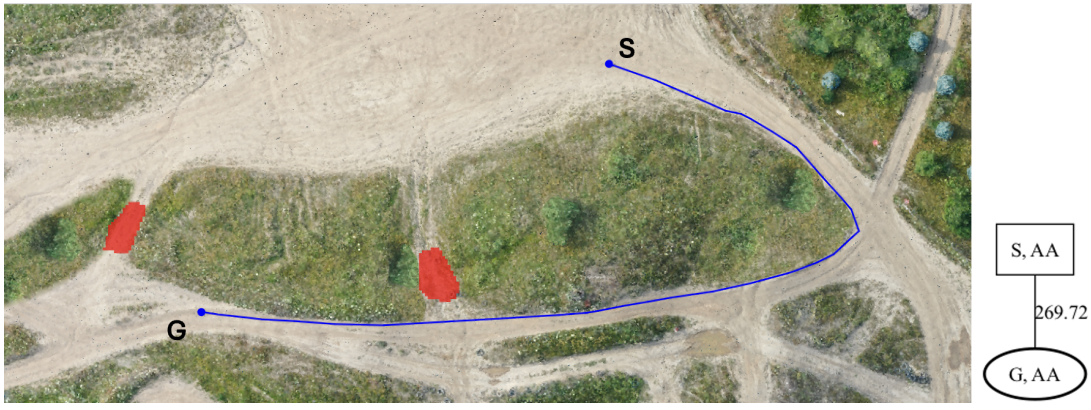
Table 5.2 presents a comparative analysis of the policy generation results across the different methods. As the step size increases in grid-based methods, the number of nodes and edges decreases, leading to faster computation times but less detailed graphs. For instance, GridGraph(1) has the lowest expected cost at 252.34 but also the longest computation time at 2.179 seconds. In contrast, GridGraph(5) has the highest expected cost of 293.12 with the shortest computation time of 0.205 seconds. Notably, for step sizes 4 and 5, the path through R2 is not represented in the graph due to the low graph resolution. This omission explains the significant increase in expected cost as the policy defaults to the deterministic path and does not disambiguate R1, thereby reducing the worst-case cost.

Our approach demonstrates considerable advantages, achieving the lowest expected cost of 236.67, a 6.2% improvement over GridGraph(1), while maintaining a competitive total time of 2.955 seconds. This improvement is primarily because our sampling-based approach allows nodes to be placed in more strategically optimal locations, leading to



(a) (b)

Figure 5.14: (Left) Spatial layout of the optimistic policy. (Right) Decision tree for the optimistic policy.



(a) (b)

Figure 5.15: (Left) Spatial layout of the conservative policy. (Right) Decision tree for the conservative policy.

shorter and more direct paths than the grid-based approach. It also achieves the best-case cost of 143.38, matching that of the optimistic policy. Compared to GridGraph(5), our method lowers the expected cost by 19.2%, despite GridGraph(5) having a faster total computation time.

In comparison to the optimistic policy, our approach reduces the expected cost by 8.6% and significantly cuts the total computation time by 35%. The optimistic policy, while achieving a low best-case cost, suffers from high worst-case cost (515.67) and longer overall computation time. Against the conservative policy, our method lowers the expected cost by 12.2%, though with a slight increase in total computation time, providing a more balanced and efficient approach to managing uncertainties. Although this analysis is based on a relatively small region, Scenario C will demonstrate how our algorithm scales to larger and more complex environments.

Table 5.2: Policy generation results for Scenario A

Method	Expected cost	Best-case cost	Worst-case cost	Policy generation time(s)	Total time(s)
GridGraph(1)	252.34	147.84	409.08	0.767	2.179
GridGraph(2)	255.23	149.85	413.33	0.148	0.685
GridGraph(3)	261.47	152.67	424.67	0.0638	0.415
GridGraph(4)	293.87	293.86	293.86	0.0137	0.251
GridGraph(5)	293.12	293.12	293.12	0.00912	**0.205**
Optimistic	258.89	**143.38**	515.67	-	4.562
Conservative	269.72	269.72	**269.72**	-	2.466
Our approach	**236.67**	**143.38**	376.61	**0.00192**	2.955

5.3.2 Simulation Scenario B

Scenario B is similar to Scenario A, with the key difference being the addition of a third uncertain region, and the blocking probability of all regions is set to 0.5. This extra region means there is no deterministic path to the goal, increasing the complexity of the path planning process and requiring the vehicle to navigate entirely through uncertain regions.

As shown in figure 5.16, the stochastic graph generated for Scenario B features stochastic edges in red and deterministic edges in blue. The graph contains 13 nodes, 14 deterministic edges, and 3 stochastic edges. This graph was created from 5 paths, with 5 planning requests pruned during the process. Pruning these requests significantly reduced the graph generation time from 12.738 seconds to 5.397 seconds, a 57.6% improvement in efficiency.

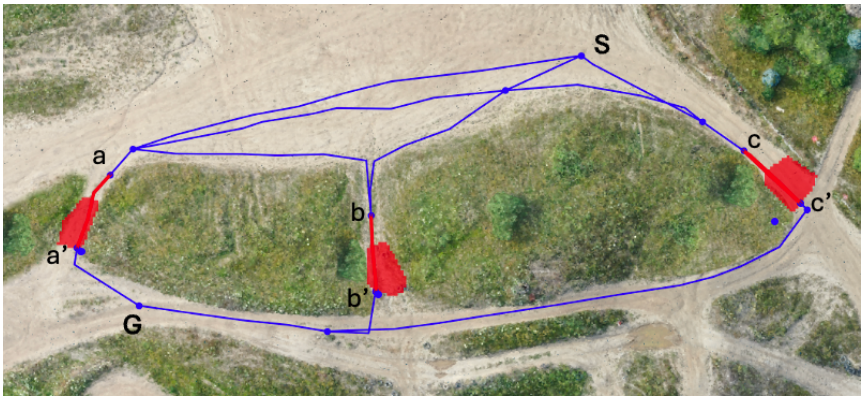


Figure 5.16: Generated stochastic graph for Scenario B

Figure 5.17 illustrates the expanded AO tree generated using our CCAO* algorithm for Scenario B, with the optimal policy highlighted in green. AND nodes are represented by ellipses and OR nodes by squares, each labeled with its cost-to-go estimate on the top left. The edges between nodes are labeled with their traversal costs, and for edges emerging from AND nodes, the probability of the disambiguation outcome is indicated in brackets.

As shown in the figure, some OR nodes, like $b, (\text{UUA})$ and $c, (\text{UAU})$, have multiple parents due to a feature in CAO* that uses a hash map to track all AND nodes added to the AO tree. This cache is checked during tree expansion to determine if a newly expanded AND node already exists. If it does, the node is linked to the current OR node being expanded, and the OR node is added to the existing AND node's parent list.

As mentioned in Section 4.5, we introduced two modifications to extend the original CAO* algorithm, making it complete and capable of outputting a policy even if there is no deterministic path to the goal. First, we replaced the use of infinite costs when no valid path exists to the goal with the NoPathPenalty, which is a finite value set higher than any other penalty or path cost in the tree. This change prevents the propagation of infinite costs, which could otherwise ruin entire subtrees during backpropagation, leading to incorrect or non-existent solutions. We set this penalty to 100,000, and as shown in the figure, it is applied to nodes $b, (\text{UUU})$ and $a, (\text{UUU})$, where at these positions, there is no path to the goal, effectively discouraging paths through these nodes while still allowing for cost propagation.

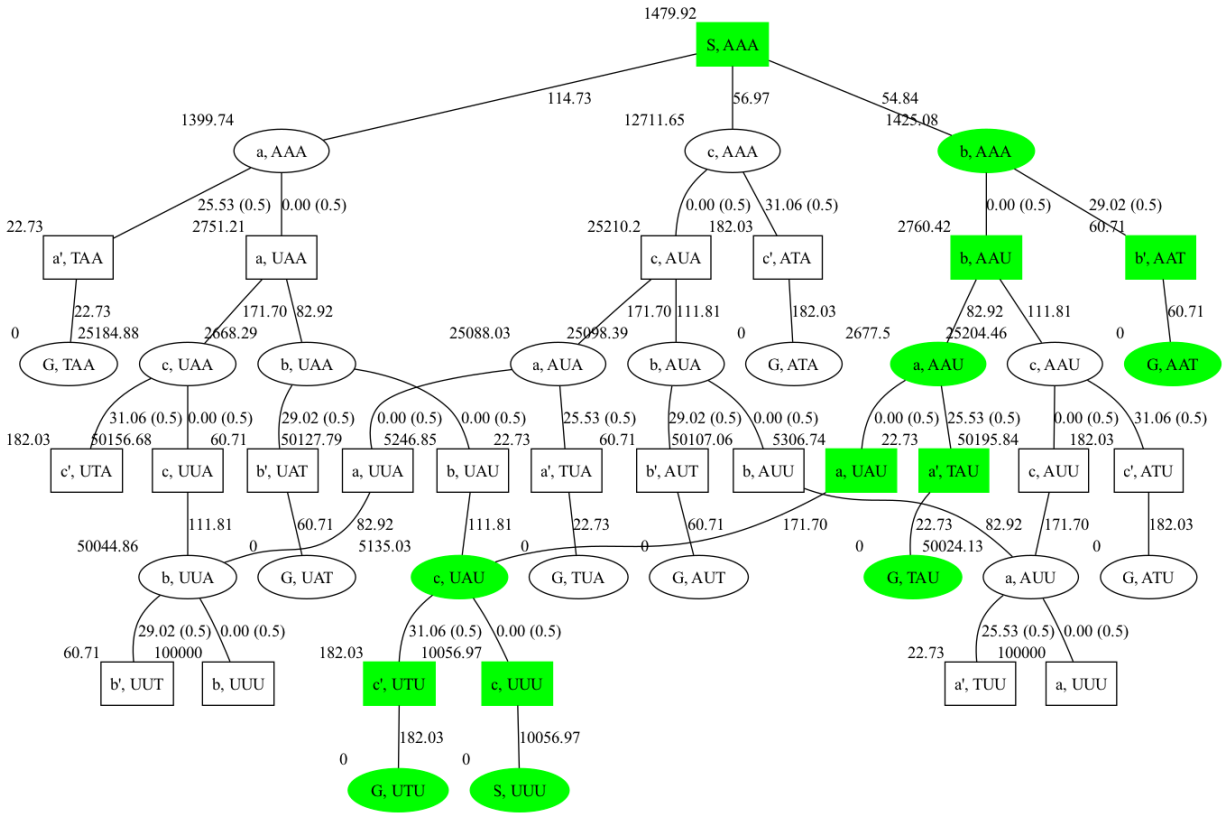


Figure 5.17: Expanded AO tree using CCAO* with the policy subtree colored in green

The second modification is triggered during the expansion step. If the most promising node, such as $c, (\text{UAU})$, has children but none of them are valid—$c', (\text{UTU})$ is solved and $c, (\text{UUU})$ doesn't have a path to the goal—the unsolved child, node $c, (\text{UUU})$, is connected back to the start with the addition of a ReturnToStartPenalty. We set this penalty to 10,000, which is higher than any possible path cost in the environment but lower than the NoPathPenalty. As shown in the figure, it is applied to the connection from node $c, (\text{UUU})$ back to the start node $S, (\text{UUU})$. This strategy ensures that a solution can still be reached, even when a deterministic path to the goal is unavailable.

The policy generated by the algorithm, highlighted in green in the figure, directs the vehicle from the start node S to disambiguate regions R2, R1, R3 in sequence. If all three uncertain regions turn out to be untraversable, leading to the state UUU, the policy then returns the vehicle to the start node.

5.3.3 Simulation Scenario C

In this scenario, we expand our experiments to cover the entire environment, as shown in Figure 5.18. The start point S is located at the bottom right, while the goal point G is positioned at the top left, requiring the vehicle to navigate across the entire terrain. This environment introduces a total of 20 uncertain regions, where we begin with a single uncertain region and incrementally add up to 20 regions, prioritizing their addition from the most critical to the least critical for the robot's navigation. Each uncertain region is highlighted in red in the figure and assigned a blocking probability of 0.5. Notably, after the third region is added, no deterministic path to the goal remains, necessitating more sophisticated path planning strategies. For computational evaluation, each approach is allocated a one-hour limit to generate a policy; if an approach exceeds this time, the algorithm is terminated and the method is deemed intractable. The presence of up to 20 uncertain regions presents a significant challenge to our approach, as well as to the grid-based and baseline methods. We will analyze how each method performs in this large-scale scenario and examine how the increasing number of uncertain regions impacts the overall expected costs and computational times. By comparing the results across methods, we aim to assess the scalability of our approach and its ability to handle environments with a high degree of uncertainty, particularly when multiple uncertain regions obstruct the path between the start and the goal.



Figure 5.18: Aerial view of the full environment and the uncertain regions (red regions) used in Scenario C.

As shown in Figure 5.19, our approach demonstrates clear advantages in terms of expected cost over other baselines. It consistently achieves the lowest expected cost as the number of uncertain regions increases, showing substantial improvements over grid-based

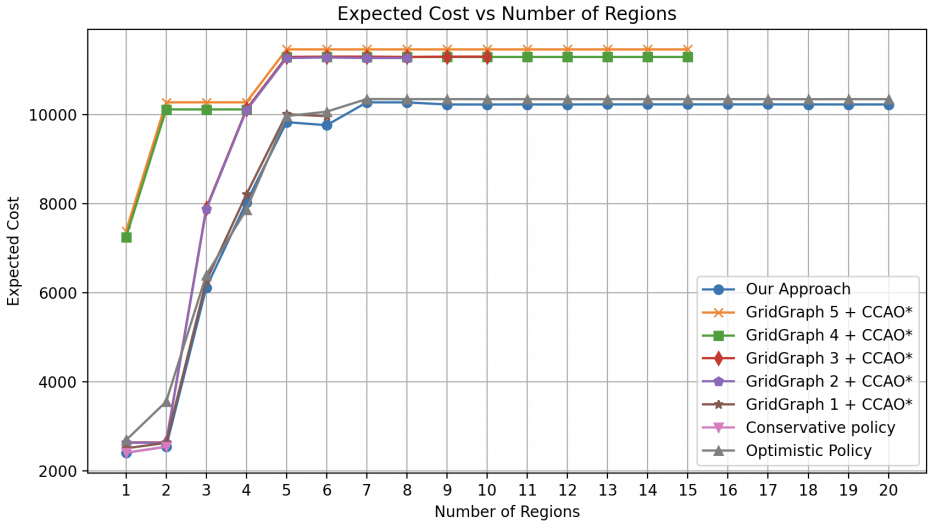


Figure 5.19: Expected cost vs the number of uncertain regions in the environment.

methods. For example, our approach yields a 2.26% lower expected cost corresponding to 149.9 meters compared to GridGraph 1 and a 22% reduction compared to GridGraph 5, while remaining scalable and computationally tractable as the number of uncertain regions grows. GridGraph 4 and 5, with their higher sampling resolution, fail to represent the deterministic path when one or two regions are present, leading to significantly higher costs compared to other methods. The conservative policy, which avoids all uncertain regions, ceases to function after 3rd region is introduced, as no deterministic path to the goal remains.

When comparing our approach to the optimistic policy, we see a 3.04% improvement in expected cost averaged across all regions. This improvement is more pronounced with 10.54% when one region is present in the environment and 28.52% when two regions are present. Additionally, when seven or more regions exist in the environment, our policy's expected cost is 120 meters lower than that of the optimistic policy. The optimistic policy disregards uncertainties and directly chooses the shortest path, often leading to higher expected costs by not accounting for potential risks. In contrast, our approach balances risk and cost more effectively, resulting in better overall performance.

When analyzing runtime, the grid-based methods begin to struggle as more regions are added. Their computation time exceeded the one-hour limit when the 7th, 9th, 11th, 16th, and 16th regions were added for GridGraphs 1 through 5, respectively. This is particularly inefficient since after the 7th region is added, the additional regions do not

critically affect the robot's ability to traverse the environment. Despite these regions being irrelevant, the grid-based methods continue to experience drastic increases in computation time. In contrast, our approach remains efficient, with computation time stabilizing after the 7th region is added. This is because the non-critical regions are not represented in the generated stochastic graph, allowing our method to avoid unnecessary computations. By focusing on relevant regions and managing uncertainty more effectively, our approach maintains a stable runtime.

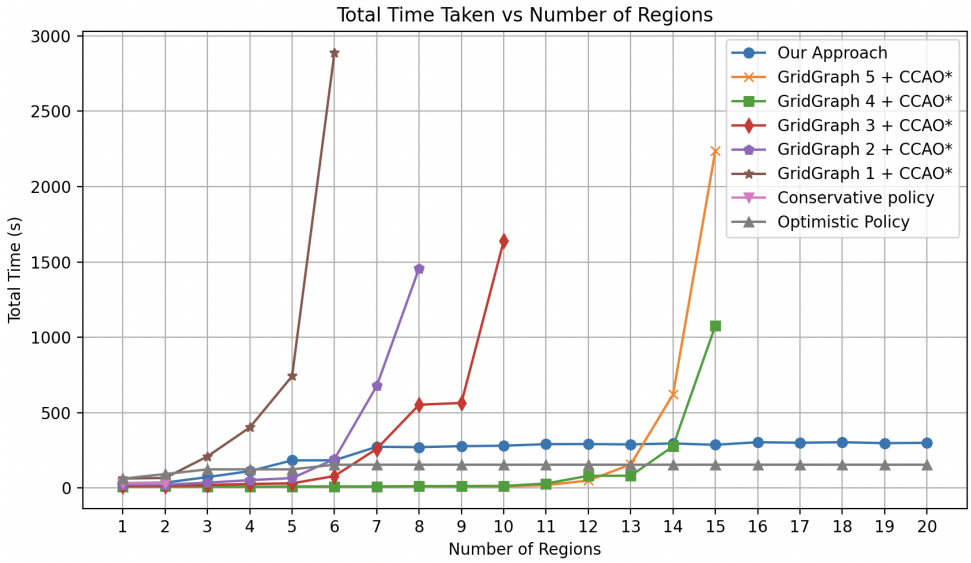


Figure 5.20: Total computation time vs the number of uncertain regions in the environment.

Finally, Figure 5.21 illustrates the total computation time for our approach with and without pruning as the number of uncertain regions increases. Up to two regions, the time difference between both methods is minimal. However, from the third region onward, pruning significantly reduces computation time. For example, when seven regions are added, the pruned approach reduces the time by 74%, from 1,054 seconds to 273 seconds. This trend highlights the efficiency of pruning, especially in larger environments with numerous uncertain regions. By reducing redundant planning requests, the pruning feature significantly improves computational performance, allowing the algorithm to scale effectively.

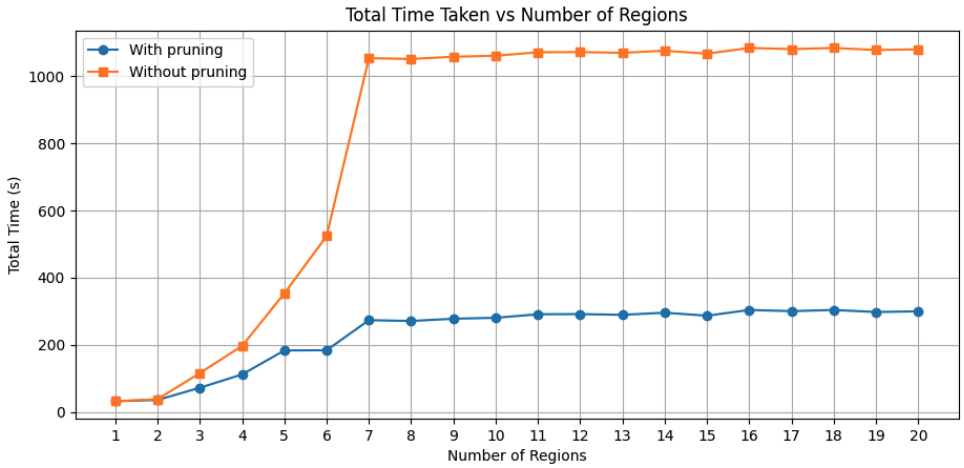


Figure 5.21: Total run time vs. number of regions, comparing policy generation with and without pruning.

Chapter 6

Conclusion and Future Work

In this thesis, we addressed the challenge of navigating ground robots through large, unstructured, and uneven off-road environments, given an initial uncertain representation of the environment. The method was built upon three key steps: generating a network of pathways, constructing a topological stochastic graph representation of the environment, and employing a novel CTP solver (CCAO*) to generate a navigation policy that minimizes the expected traversal cost. Extensive testing using real environment data demonstrated that our approach consistently provided the lowest expected traversal cost compared to traditional methods. Furthermore, the results highlighted the scalability of our method, proving its effectiveness in large and complex regions while maintaining computational efficiency.

In this work, we have made several key assumptions regarding uncertain region extraction. While our current approach utilizes unsupervised clustering to handle uncertainty by grouping grids with similar visual and spatial features, the probability of a region being blocked is assumed to be known, and an arbitrary probability is assigned to each region. One promising direction for future work is inspired by the approach detailed in [64]. This work introduces a novel method of region-based semantic classification that interprets images as tessellations of learnable regions with homogeneous semantics. Their approach not only improves the efficiency of semantic segmentation but also generates a per-region probability distribution of possible terrain classes as a byproduct of the semantic classification process. Incorporating this method into our framework could significantly enhance the practicality of our uncertainty modeling. Instead of relying on arbitrary probabilities for uncertain regions, we could use the per-region probability distributions provided by the region-based semantic classification. This would allow for a more data-driven and context-aware assignment of uncertainty, potentially leading to improved decision-making in path

planning and navigation tasks.

Furthermore, in this thesis, we employed a modified version of CAO* to minimize the expected traversal cost given a stochastic graph. Nevertheless, several alternative objectives merit exploration. One such objective is maximizing the stochastic on-time arrival (SOTA) probability [65], which aims to identify a path with the highest likelihood of reaching the destination within a predefined deadline. Another potential direction is minimizing travel time while adhering to a high-confidence constraint (α-RSP) [66], ensuring that the path's travel time remains below a specified threshold with a probability greater than α. Additionally, optimizing the minimal combination of mean and standard deviation in travel time, or alternatively, minimizing mean travel time subject to variance constraints [67], presents another promising area for future research. Moreover, incorporating the conditional value at risk (CVaR) as an objective could be valuable for developing more sophisticated, risk-averse strategies in path planning [22].

Finally, we could extend this approach to generate a stochastic graph representation of the environment that encompasses routes between multiple points of interest rather than just between only the start and goal points. This would enable more flexible and comprehensive navigation strategies, supporting scenarios where the robot must visit multiple key locations efficiently while accounting for uncertainty in the environment [36].

References

[1] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer, "Distributed search and rescue with robot and sensor teams," in *Field and Service Robotics*. Springer, 2003, pp. 529–538.

[2] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Whittaker, "Ambler: An autonomous rover for planetary exploration," *Computer*, vol. 22, no. 6, pp. 18–26, 1989.

[3] M. Massari, G. Giardini, and F. Bernelli-Zazzera, "Autonomous navigation system for planetary exploration rover based on artificial potential fields," in *Proceedings of Dynamics and Control of Systems and Structures in Space (DCSSS) 6th Conference, 2004*, 2004, pp. 153–162.

[4] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments," *J. Field Robotics*, vol. 34, pp. 940–984, 2017.

[5] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, "STEP: Stochastic traversability evaluation and planning for risk-aware off-road navigation," in *Proc. Robot.: Sci. Syst., 2021, p. 21*, 2021.

[6] T. Overbye and S. Saripalli, "G-VOM: A GPU Accelerated Voxel Off-Road Mapping System," *IEEE Intelligent Vehicles Symposium (IV), Aachen, Germany*, pp. 1480–1486, 2022.

[7] T. Cortinhal, G. Tzelepis, and E. Aksoy, "SalsaNext: Fast semantic segmentation of lidar point clouds for autonomous driving," *ArXiv*, vol. 2003, p. 03653, 2020.

[8] J. Xie, Y. Qian, B. Wang, C. Wang, and M. Yang, "Off-load semantic terrain classification using 3d lidar data," *China Automation Congress (CAC), Xiamen, China*, pp. 5960–5965, 2022.

[9] F. Schilling, X. Chen, J. Folkesson, and P. Jensfelt, "Geometric and visual terrain classification for autonomous mobile navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada*, 2017, pp. 2678–2684.

[10] A. Shaban, X. Meng, J. Lee, B. Boots, and D. Fox, "Semantic terrain classification for off-road autonomous driving," in *Conference On Robot Learning*, 2021.

[11] X. Meng, N. Hatch, A. Lambert, A. Li, N. Wagener, M. Schmittle, J. Lee, W. Yuan, Z. Chen, S. Deng, G. Okopal, D. Fox, B. Boots, and A. Shaban, "TerrainNet: Visual modeling of complex terrain for high-speed, off-road navigation," *Robotics: Science And Systems XIX, Daegu, Republic Of Korea, July*, vol. 10-14, 2023.

[12] T. Guan, Z. He, R. Song, D. Manocha, and L. Zhang, "TNS: Terrain traversability mapping and navigation system for autonomous excavators," *Robotics: Science And Systems XVIII, New York City, NY, USA, June*, vol. 27 - 1, July 2022.

[13] D. Ferguson, A. Stentz, and S. Thrun, "PAO for planning with hidden state," in *IEEE International Conference on Robotics and Automation.*, vol. 3, 2004, pp. 2840–2847 Vol.3.

[14] T. Overbye and S. Saripalli, "Fast local planning and mapping in unknown off-road terrain," in *IEEE International Conference on Robotics and Automation (ICRA), Paris, France*, 2020, pp. 5912–5918.

[15] D. I. T. H. Y. Leung and A. Zolotas, "Hybrid terrain traversability analysis in off-road environments," in *8th International Conference on Automation, Robotics and Applications (ICARA)*, 2022, pp. 50–56.

[16] J. W. X. Zhang, Y. Fang, and J. Yuan, "Multilevel humanlike motion planning for mobile robots in complex indoor environments," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1244–1258, 2019.

[17] C. Shen, S. Yu, and T. Ersal, "A Three-Phase Framework for Global Path Planning for Nonholonomic Autonomous Vehicles on 3D Terrains," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 160–165, 2021.

[18] B. B. M. Brunner and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 5539–5544.

[19] F. Atas, G. Cielniak, and L. Grimstad, "Elevation state-space: Surfel-based navigation in uneven environments for mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan*, 2022, pp. 5715–5721.

[20] S. Koenig and M. Likhachev, "D*lite," in *Eighteenth National Conference on Artificial Intelligence*. USA: American Association for Artificial Intelligence, 2002, p. 476–483.

[21] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong Planning A*," *Artificial Intelligence*, vol. 155, pp. 93–146, 2004.

[22] A. Majumdar and M. Pavone, "How should a robot assess risk? towards an axiomatic theory of risk in robotics," in *Robotics Research*, ser. Springer Proceedings in Advanced Robotics., N. Amato, G. Hager, S. Thomas, and M. Torres-Torriti, Eds. Cham: Springer, 2020, vol. 10.

[23] D. D. Fan, A.-A. Agha-Mohammadi, and E. A. Theodorou, "Learning risk-aware costmaps for traversability in challenging environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 279–286, 2022.

[24] B. H. Wang, B. Asfora, R. Zheng, A. Peng, J. Banfi, and M. Campbell, "Multiple-hypothesis path planning with uncertain object detections," *ArXiv*, vol. 2308, p. 07420, 2023.

[25] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, December 2011.

[26] W. Han, A. Jasour, and B. Williams, "Non-Gaussian Risk Bounded Trajectory Optimization for Stochastic Nonlinear Systems in Uncertain Environments," in *International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA*, 2022, pp. 11 044–11 050.

[27] A. Jasour, W. Han, and B. C. Williams, "Convex risk-bounded continuous-time trajectory planning and tube design in uncertain nonconvex environments," *The International Journal of Robotics Research.*, vol. 42, no. 10, pp. 705–728, 2023.

[28] C. E. Priebe, D. E. Fishkind, L. Abrams, and C. D. Piatko, "Random disambiguation paths for traversing a mapped hazard field," *Naval Research Logistics*, vol. 52, pp. 285–292, 2005.

[29] Y. Han, H. Lin, J. Banfi, K. Bala, and M. Campbell, "DeepSemanticHPPC: Hypothesis-based planning over uncertain semantic point clouds," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4252–4258.

[30] J. Banfi, L. Woo, and M. Campbell, "Is it worth to reason about uncertainty in occupancy grid maps during path planning?" in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 11 102–11 108.

[31] D. E. Fishkind, C. E. Priebe, K. E. Giles, L. N. Smith, and V. Aksakalli, "Disambiguation protocols based on risk simulation," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 5, pp. 814–823, September 2007.

[32] C. H. Papadimitriou and M. Yannakakis, "Shortest paths without a map," *Theor. Comput. Sci.*, vol. 84, no. 1, pp. 127–150, 1991.

[33] V. Aksakalli and I. Ari, "Penalty-based algorithms for the stochastic obstacle scene problem," *INFORMS Journal On Computing*, vol. 26, pp. 370–384, 2014.

[34] H. Guo and T. D. Barfoot, "The robust canadian traveler problem applied to robot routing," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5523–5529.

[35] D. Dey, A. Kolobov, R. Caruana, E. Kamar, E. Horvitz, and A. Kapoor, "Gauss meets canadian traveler: Shortest-path problems with correlated natural dynamics," in *Autonomous Agents and Multi-Agent Systems*, 2014.

[36] Y. Huang, H. Dugmag, T. D. Barfoot, and F. Shkurti, "Stochastic planning for asv navigation using satellite images," in *IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom*, 2023, pp. 1055–1061.

[37] V. Aksakalli, O. F. Sahin, and I. Ari, "An AO* based exact algorithm for the canadian traveler problem," *INFORMS J. Comput.*, vol. 28, no. 1, pp. 96–111, 2016.

[38] D. Fried, S. Shimony, A. Benbassat, and C. Wenner, "Complexity of canadian traveler problem variants," *Theoretical Computer Science*, vol. 487, pp. 1–16, 2013.

[39] C. Chang and J. Slagle, "An admissible and optimal algorithm for searching AND/OR graphs," *Artificial Intelligence*, vol. 2, pp. 117–128, 1971.

[40] V. Aksakalli, "The BAO* algorithm for stochastic shortest path problems with dynamic learning," in *46th IEEE Conference on Decision and Control, New Orleans, LA, USA*, 2007, pp. 6003–6008.

[41] X. Ye and C. E. Priebe, "A graph-search based navigation algorithm for traversing a potentially hazardous area with disambiguation," *IJORIS*, vol. 1, no. 3, pp. 14–27, 2010.

[42] W. Y. Kwon, I. H. Suh, and S. Lee, "SSPQL: Stochastic shortest path-based Q-learning," *International Journal of Control, Automation and Systems*, vol. 9, pp. 328–338, 2011.

[43] D. Ferguson and A. Stentz, "Planning with imperfect information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, vol. 2*, 2004, pp. 1926–1931.

[44] G. H. Polychronopoulos and J. N. Tsitsiklis, "Stochastic shortest path problems with recourse," *Networks: An International Journal*, vol. 27, no. 2, pp. 133–143, 1996.

[45] S. Gao and I. Chabini, "Optimal routing policy problems in stochastic time-dependent networks," *Transportation Research Part B: Methodological*, vol. 40, no. 2, pp. 93–122, 2006.

[46] B. Stenning and T. D. Barfoot, "Path planning with variable-fidelity terrain assessment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan*, 2010, pp. 300–306.

[47] J. S. Provan, "A polynomial-time algorithm to find shortest paths with recourse," *Networks*, vol. 41, pp. 115–125, 2003.

[48] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec 1959.

[49] N. N. P. Hars and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[50] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[51] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[52] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research.*, vol. 30, no. 7, pp. 846–894, 2011.

[53] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *IEEE International Conference on Robotics and Automation, Detroit, MI, USA, vol. 2*, 1999, pp. 1018–1023.

[54] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, December 2005.

[55] C. Holleman and L. Kavraki, "A framework for using the workspace medial axis in prm planners," in *Proceedings ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, vol. 2, 2000, pp. 1408–1413 vol.2.

[56] T. S. C. Nissoux and J.-P. Laumond, "Visibility based probabilistic roadmaps," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients*, vol. 3, 1999, pp. 1316–1321 vol.3.

[57] D. Hsu and Z. Sun, "Adaptively combining multiple sampling strategies for probabilistic roadmap planning," in *IEEE Conference on Robotics, Automation and Mechatronics*, vol. 2, 2004, pp. 774–779 vol.2.

[58] X. T. S. Thomas, M. Morales and N. M. Amato, "Biasing samplers to improve motion planning performance," in *Proceedings IEEE International Conference on Robotics and Automation*, 2007, pp. 1625–1630.

[59] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[60] B. Horn, "Hill shading and the reflectance map," *Proceedings of the IEEE*, vol. 69, no. 1, pp. 14–47, 1981.

[61] D. D. S. J. R. Shawn and A. E. Robert, "Terrain ruggedness index that quantifies topographic heterogeneity," *intermountain Journal of Science*, vol. 5, no. 1-4, pp. 23–27, 1999.

[62] R. H. F. K. A. T. A. Dehecq, E. Mannerfelt, "xdem," May 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4809698

[63] K. S. A. L. P. F. R. Achanta, A. Shaji and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[64] B. P. Y. Zhang and C. Lu, "Semantic segmentation by early region proxy," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 1248–1258.

[65] H. Frank, "Shortest paths in probabilistic graphs," *Operations Research*, vol. 17, no. 4, pp. 583–599, 1969.

[66] I. Murthy and S. Sarkar, "Stochastic shortest path problems with piecewise-linear concave utility functions," *Management Science*, vol. 44, no. 11-part-2, pp. S125–S136, 1998.

[67] T. Uchida and Y. Iida, "Risk assignment: A new traffic assignment model considering the risk of travel time variation." *Transportation and Traffic Theory*, 1993.