Motion Planning for UGVs in Terrain Scenarios Based on RRT, RRT* and the Fast Marching Method

University of Oslo

Faculty of Mathematics and Natural Sciences

May 2023

A Thesis Submitted for the Degree of

Master of Science in Informatics: Robotics and Intelligent Systems, Cybernetics and Autonomous Systems

Lars Kristian Bårdevik

Main Supervisor: Marius Thoresen, MSc

Co-Supervisor: Kim Mathiassen, PhD

Abstract

This thesis presents a sampling scheme for traversability mapping and a motion planning approach for terrain scenarios. The sampling scheme uses overlapping minimum enclosing cylinders to analyze a point cloud and to estimate terrain inclinations over a grid map. For motion planning, an objective cost function is defined, that allows penalization of an interval of inclination values based on varying a parameter λ. RRT and RRT* are extended with a motion primitive scheme that ensures an implicit obstacle-clearance by performing collision checking in a minimum enclosing radius from every node in a tree. In the RRT-variant, accumulated costs through branches from the root node of the tree, in a field defined by the objective function, are saved at every node. A path is constructed by picking the node with the lowest accumulated cost from the root within a radius from the goal. The RRT*-variant also utilizes the cost field defined by the objective function, and employs a proposed BVP-rewiring scheme. For comparison, the motion primitive variants are modified to use inclination cost directly.

Results show that the proposed motion primitive scheme produces more traversable paths when higher inclinations are penalized, indicating that the collision-checking approach is the significant contributor to higher path traversability for the sampling-based variants. This finding is supported by the results, as no apparent trend is observed with respect to λ when the motion primitive variants are compared directly, but a clear trend is observed with respect to the ideal benchmarks. Since no apparent trend is observed between the proposed algorithms, this suggests the RRT* optimization scheme does not produce more traversable paths compared to paths produced by the RRT-variants. The finding that collision-checking is the significant contributor to higher path traversability for the sampling-based variants, is also supported by the results from a simple non-holonomic version of FMM, which is included to show that for vehicles without netural-turn and reverse-motion capabilities, feasibility is not guaranteed by FMM alone.

Preface

Motion planning is a truly fascinating research problem that becomes increasingly challenging to solve as the complexity of a planning problem increases. This complexity depends on what a researcher chooses to prioritize when tackling the problem. Whatever the researcher's choices, eventually there will be a trade-off. When one aspect is prioritized to improve the quality of solutions, another aspect may be ignored, and this can potentially result in a degradation of solution quality. For example, if one chooses to use a sophisticated vehicle model, this may result in solution paths that are more feasible than those generated using a simpler model. On the other hand, generating solution paths using an advanced vehicle model may result in a difficult, non-linear optimization problem. Due to the inherent trade-offs in motion planning, it may seem difficult to discern which contributions ensure progress in the field. Yet, it is clear that motion planning research has led to significant advancements in terms of enabling autonomous navigation in a variety of scenarios. With continuing technological advancements that work to simplify and improve aspects of this research, the future of motion planning looks bright.

Acknowledgements

I am deeply grateful to my supervisors, Marius Thoresen and Kim Mathiassen, for asking critical questions and for their invaluable advice. In particular, a more nuanced representation of the terrain was enabled by Dr. Mathiassen's suggestion to save inclination estimates immediately, instead of using them only to determine the separation between obstacles and free space. Marius Thoresen proposed using the inclination map directly in planning. This provided a more transparent evaluation of the algorithms, enabling a deeper understanding of their effectiveness. I attempted to develop a working non-holonomic version of FMM for vehicles without neutral-turn capabilities, but I found this to be a challenging problem. This realization strengthened the motivation for combining FMM with a randomized approach to address motion constraints of certain types of vehicles. My supervisors suggested the inclusion of the results from the NH-FMM method as a means of providing evidence that FMM does not guarantee feasibility for such vehicles. The motion planning approach presented in this thesis is inspired by their work on Traversability Hybrid A*, which produced notable results in a real-world experiment on terrain. I feel fortunate to have had the opportunity to discuss research problems with experts in the field. My dear parents and friends, I would not have been able to complete this thesis without your love and support.

Contents

1 Introduction

Motion planning intends to provide robots with the capability of making autonomous decisions on where to move, so that they are able to perform tasks without explicit instruction. Some applications of the field include space exploration [1], agriculture [2], self-driving in traffic [3] and off-road scenarios [4], and military operations [5]. It is evident that self-driving on terrain poses a number of challenges, such as those related to motion constraints, obstacle avoidance, and traversability, and many of these challenges remain unsolved [6].

The DARPA Grand Challenge was organized with the aim of accelerating technological development of autonomous vehicles [4]. To successfully complete the competition, self-driving vehicles had to navigate a 142 mile course in under 10 hours through desert terrain consisting of steep hills, mountain passes and sharp turns [4]. Stanley, a Stanford University vehicle, won in 2005 [7]. The Carnegie Mellon University car, Boss, won the DARPA Urban Challenge of 2007, successfully navigating in a dynamic traffic scenario [3]. The DARPA competitions have highlighted the difficulty of developing methods that enable effective autonomous navigation in complex and dynamic environments.

Motion planning is often applicable in situations that require alternatives to human operation. The Perseverance Mars Rover requires a significant level of surface autonomy due to the commmunications latency between Earth and Mars [1]. In agriculture, the adoption of motion planning methods can increase productivity [2]. Research into autonomous navigation is focused on increasing safety and efficiency. Therefore, motion planning is also relevant for military applications.

At the Norwegian Defence Research Establishment (FFI), researchers are investigating self-driving on terrain. The motion planning problem is part of this study. In such scenarios the problem is complex due to features of the terrain, and the ability of a UGV to traverse it [8]. In this case, path optimality may depend on characteristics such as inclination and roughness [9]. Due to the complexity of this problem, more sophisticated methods are needed to improve upon the traditional approaches, which rely on a simple division of a world model into obstacles and free space [8]. Mathiassen et al. developed an autonomous UGV platform for the purpose of enabling experiments in outdoor terrain scenarios [10].

1.1 Previous Work

The class of motion planning methods that has been given the most research effort can be divided into grid-based and sampling-based methods [11]. Grid-based planners work by overlaying a grid on the configuration space [12]. These algorithms can be used with nonuniform cost-maps, and they are complete planning methods meaning that, in general, they will find a solution if it exists [8]. Sampling-based methods do not rely on an explicit construction of the configuration space, and provide probabilistically complete solutions (i.e., the probability that an existing solution is found converges to one given enough samples) [13]. Sampling-based motion planning schemes employ a collision-detection module, which enables probabilistic probing of the configuration space [6]. Solutions are constructed from established connectivity, between samples drawn, in the free configuration space. Two fundamental types of sampling-based algorithms are the single-query and multi-query algorithms. Single-query algorithms solve an instances of motion planning problems based on initial-goal configuration pairs, and often explore only a subset of the configuration space relevant for the problem at hand [14]. Multi-query algorithms can accept multiple initial-goal configuration pairs to solve several instances of a motion planning problem [15], based on a preconstructed roadmap that richly represents the connectivity of the free configuration space [14]. Hybrid motion planning methods combine features from grid-based and sampling-based methods [8].

1.1.1 Grid-Based Methods

Edsger W. Dijkstra published an article in 1959 [16], presenting an algorithm capable of finding the shortest path in a graph structure, given positive cost values. The algorithm repeatedly visits closest unvisited nodes, starting at an initial node, until either the goal is found or there is no connection between the starting point and the nodes remaining that are not yet visited, while updating nodes with the least cost (resulting in the shortest path) to the starting point. When representing the search space as a grid, center points of grid cells correspond to nodes. Dijkstra's algorithm was used by the Ben Franklin Racing Team in the DARPA Urban Challenge for recomputing waypoints when their vehicle encountered blocked intersections or lanes [17].

A-star (A*) extends Dijkstra's algorithm by a heuristic function that adds an estimated cost to the goal, to the original cost [18]. Nodes are given priorities based on approximate costs-to-go, which results in a greedy exploration of nodes (as opposed to a repeated exploration of closest, unvisited neighbors). The algorithm is mostly used to search for paths in spaces known a priori to a vehicle [19], and forms a basis for the A* family of algorithms which have been used for various motion planning purposes in mobile robotics. The Anytime Dynamic A* (AD*) algorithm combines features from incremental re-planning with anytime planning to increase efficiency in complex and dynamic environments, resulting in bounded sub-optimal solutions [20]. In the DARPA Urban Challenge, the Carnegie Mellon University team exploited AD*'s ability to improve the quality of an initial sub-optimal solution to achieve the complex maneuvering that was needed for example in the presence of partially blocked intersections [21]. In 1994, Anthony Stentz presented the Dynamic A* (D*) algorithm, which can dynamically update the cost of directed edges (i.e., arcs in a directed graph) to achieve optimal and efficient planning in partially known environments [22]. D* Lite is more efficient than D*, and both are incremental versions of A*, meaning that they recycle information for new tasks instead of solving similar tasks completely from scratch [23]. The combination of features from heuristic search and incremental search makes these algorithms more suitable than A*, for planning problems in unknown environments, as these scenarios require efficient re-planning capabilities [23].

Stentz et al. introduced the Field D* algorithm, attempting to tackle the problem of sub-optimal paths that arise due to the limited number of possible transitions from a given graph vertex when graphs are used to approximate a grid-based environment representation [24]. An A* grid-search is done by first placing vertices at the center of each grid-cell which limits movement to be between these centers, resulting in non-smooth paths [8]. In Field D*, vertices are placed at the corners of the grid-cells and linear interpolation is used to approximate path costs for every point on the boundary of grid cells, which results in smoother paths [24]. The algorithm has been used for global path planning on terrain by NASAs Spirit and Opportunity Mars Exploration rovers due to its ability to provide more direct, low-cost paths while maintaining real-time performance [25].

The Fast Marching Method (FMM) is another grid-based method that computes solutions to the discrete Eikonal equation, a non-linear first order partial differential equation, which

describes the motion of a wavefront [26]. This equation contains both the speed function of the wavefront at a given point (or grid-cell in the discrete case) and the gradient of its time of arrival function to a given point from the source of the wavefront. The wavefront can be reperesented as a curve in 2D or a surface in 3D, and the mathematical model is applicable to higher dimensions. FMM estimates the time of arrival of the wavefront to every cell in a gridmap from the source, and a smooth path is traced by computing and following the maximum gradient direction over cells with stored time of arrival values, starting at the target cell. The method approximates the Euclidian distance between any cells in the gridmap which presents an advantage over the A* grid-search, for which the true Euclidian distance between non-neighbouring grid cells is overestimated as movement is only allowed between the centres of the cells [8]. Even though FMM computes the shortest path, this path may not be feasible due to (for example) the generated path's proximity to obstacles [26].

Researchers have provided extensions to FMM in attempts to overcome safety challenges, increase efficiency and to improve the quality of generated paths. In the FM over Voronoi Diagram method, a generalized Voronoi diagram is used to obtain a roadmap of a binary obstacle gridmap with enlarged obstacles (i.e., to reduce collision risk), which decreases the required search time when FMM is applied [26]. FM^2 uses a fast marching gridmap which is generated by expanding waves at all obstacles in the binary gridmap [26]. Scaling the fast marching gridmap yields maximum safe speeds at all points in the environment and a shortest-time path can be obtained by applying FMM. The saturated variation of FM^2 generates paths at closer, but safe distances to obstacles [26]. There is also a variation of FM^2 that uses a heuristic estimate of the cost-to-go to limit wave expansion [26]. The use of a heuristic estimate of the cost-to-go makes this algorithm similar to the A*. Garrido et al. extended FMM for outdoor planning purposes, using a triangular mesh to represent 3D surfaces and incorporating the calculation of a weight matrix to limit wave propagation speed depending on task requirements [27].

Although grid-based methods have been improved, they present limitations when planning with UGVs in real-world scenarios. The A* family of algorithms are known to suffer from an increase in computational complexity for configuration spaces of higher dimensions [28]. Additionally, paths generated by the A* algorithms are not well suited for

vehicles with kinematic constraints [8]. There are extensions of FMM which support non-holonomic constraints for binary obstacle maps (e.g., [29], [30]). Sampling-based planning algorithms are suitable for planning in high-dimensional configuration spaces [28]. They can also sample with motion primitives (i.e., precomputed feasible paths corresponding to single driving maneuvers), which ensures that generated paths do not violate the kinematic constraints of a given UGV [8].

1.1.2 Sampling-Based Methods

The multi-query Probabilistic Roadmap (PRM) algorithm, presented by Kavraki et al. in 1996, relies on a learning phase and a query phase [15]. The output of the learning phase is a probabilistic roadmap that PRM generates from random configurations in the free configuration space of a robot. Random configurations are connected using a local planner, and the resulting roadmap is stored as an undirected graph, with edges corresponding to simple, feasible paths. In the query phase, multiple queries can be made for paths connecting free initial configurations with free goal configurations. PRM tries, in the query phase, to find paths from the initial and goal configurations to two vertices within the roadmap. A complete path is constructed by concatenating edges found in a graph search, which connect the vertices. In [31], Svestka et al. extended PRM for motion planning with non-holonomic vehicles on planar surfaces by employing a simple local method which constructed paths in absence of obstacles, and reported success if the shortest path (consisting of two extreme rotations and one translation), connecting two input configurations, did not intersect any obstacles. Through simulations and an experiment with a Segway Robotic Mobile Platform, Kobilarov et al. demonstrated near time-optimal motion planning on outdoor terrain surfaces (approximated by planar patches) using a control-driven PRM planner that accounted for kinematic and dynamic constraints [32].

In their presentation of the single-query Rapidly-exploring Random Trees (RRTs) algorithm, Lavalle et al. propose that PRM is impractical when planning under differential constraints, due to the difficulty of designing a local planner [33]. RRT quickly searches nonconvex, high-dimensional spaces, by randomly sampling the search space to incrementally build a space-filling tree that is biased to grow toward large unsearched areas [34]. This algorithm can be adapted to account for non-holonomic and kinodynamic constraints,

using a state transition equation that is integrated in a simulator to check that constraints are satisfied [6]. Compared to RRT, PRM constructs a roadmap a priori. Constructing this roadmap may be computationally challenging, and in many planning problems the environment is unknown [35]. A shortcoming of both RRT and PRM is that they do not consider path optimality. In [35], Karaman et al. presented RRT* and PRM*, which provide asymptotic optimality (AO; convergence toward optimal solutions as the number of samples goes to infinity) by minimizing distance-based cost [12]. As part of the tree construction, RRT* incrementally reroutes edges between vertices and selects parent vertices with the lowest cost between the initial vertex and randomly chosen vertices, to achieve AO [36]. RRT* and its extensions have been used for various motion planning problems involving UGVs on terrain.

Furgale et al. presented a novel framework for autonomous driving on terrain that relies on point cloud data obtained locally by a UGV [37]. In their publication, they state that motion planning methods which rely on explicit surface reconstruction can be computationally expensive for UGVs to perform locally. The method presented by Furgale et al. performs only local terrain assessment by approximating surfaces on-demand using robot-sized planar patches and analyzing the local distribution of points in the point cloud map. In their extension of PRM for autonomous terrain navigation, Kobilarov et al. approximated surfaces locally using small planar patches [37] [32]. The novelty of the method proposed by Furgale et al. lies in their development of a local trajectory optimization scheme that, based on an objective function, iteratively optimizes trajectories without having to consider gradient information or explicit mapping of obstacles. The objective function can be customized to include any quality measures, for example related to distance, path length, terrain smoothness or curvature. Local trajectory optimization is used in this method to improve collision-free paths generated by RRT*-optimized RRT trajectories.

In the publication by Furgale et al., one can observe that the paths generated by RRT are jerky. Although RRT* produces smoother paths in a second step, this feature remains here as well. The jerkyness of trajectories generated by RRT and its extensions have been commented on by several researchers (e.g., [38] [39]), and there are other approaches which can be taken for path smoothing such as B-spline based post-processing [39]. The benefit of the framework proposed by Furgale et al. is that it is developed for motion planning on terrain,

and the work done so far on optimization of nonplanar trajectories is limited [37]. As shown in [40], sampling with motion primitives is another method which can help ensure obtained paths are smoother and respect constraints. Both RRT and RRT* have support for planning under both kinematic [33] [40] and kinodynamic [33] [41] constraints.

Takemura et al. developed a traversability-based version of RRT* for planetary rover motion planning on rough terrain [36]. This method samples vertices directly from dense point cloud data, and includes a traversability assessment, based on a user-defined region and travel cost between two vertices. The travel cost is based on a weighted sum of the rover's roll, pitch and yaw angles and the distance between the vertices. Through simulation, Takemura et al. demonstrated that this assessment can ensure safe and feasible paths (i.e., not AO) on rough terrain. Similar to [37], this approach does not rely on explicit terrain surface reconstruction.

The Particle RRT algorithm, developed by Simmons et al. for rover path planning on terrain, extends RRT by explicitly considering environment uncertainty [42]. In the tree expansion of RRT, the original algorithm applies random actions to currently known and reachable tree states to reach new states. Simmons et al. argue that since the determination of the success of applying an action is binary, there is no room for ranking of multiple actions in tree expansion. The determination is binary because applying an action either satisfies constraints and leads to no collision or it is not applied. Another argument given by Simmons et al. is that actions associated with high cost may be ignored in the original RRT algorithm, even though better solutions may not exist. To obtain paths that are somewhat robust to environment uncertainty, Particle RRT treats each extension to the tree as a stochastic process, and tree extensions are chosen based on the expected probability of executing a path successfully.

Jiang et al. developed R2-RRT*, an extension of RRT* that analyzes mobility reliability to generate reliable and robust paths in face of environment uncertainty in off-road scenarios, by considering soil properties and slope information [43]. Prior to planning, using this approach, the target map and terrain information must be well-characterized, which means the approach may not be applicable when, for example, the only terrain information available comes from sensor data obtained locally. In this case, Particle RRT may be

more applicable as it is developed for UGVs operating on rough terrain with unknown coefficients of friction [42]. Particle RRT performs repeated simulation of the execution of a chosen action using likely friction coefficients. As mentioned by Simmons et al. in their application of Particle RRT on terrain, even when considering interactions between vehicle and soil, vehicle slip and friction coefficients, the simulated results cannot necessarily be replicated in a real-world scenario [44]. However, Simmons et al. also conclude that the generated paths were more reliable and optimal than paths obtained with the basic RRT algorithm. Lee et al. developed a chance-constrained version of RRT* (CC-RRT*) for autonomous tracked vehicles in high slip terrain by modelling the effect of slip on the vehicle (on different soil types) as a process noise with high uncertainty [45].

Another sampling-based method is the Fast Marching Tree (FMT*) algorithm, which builds on RRT* and PRM* (among other methods) [46]. This algorithm performs a forward dynamic programming recursion on a predetermined number of samples, and concurrently searches and builds a tree similar to RRT*. Hence, FMT* differs from PRM-like algorithms as it does not rely on a preconstructed roadmap. FMT* is similar to the grid-based FMM because it moves outward into cost-to-arrive space to construct the tree, and the samples that have been visited are not visited again and therefore not considered for new connections. However, the algorithm does not explicitly solve the discrete Eikonal equation. To increase efficiency, FMT* relies on a lazy collision checking scheme which first assumes no obstacles. According to Janson et al., this scheme may introduce sub-optimal solutions, but if more samples are used, this reduces the number of occurences of such solutions. The method is AO and converges faster than PRM* and RRT*, although uses a definition of AO that differs from the definition used by PRM* and RRT*, in that the algorithm is shown to converge in probability toward optimal solutions as the number of samples increases (a less strict definition). FMT* has been extended for non-holonomic constraints [47], but little work has been done on extending the algorithm for motion planning with UGVs in outdoor terrain scenarios, for example by including a traversability assessment.

Sampling-based methods can solve motion planning problems in high-dimensional configuration spaces that grid-based planners cannot [48]. Many of the algorithms have frameworks in place for motion planning under both kinematic and kinodynamic constraints. Sampling-based motion planning algorithms avoid explicit construction of the

configuration space, and the different sampling schemes present several advantages. Sampling with motion primitives can help ensure solutions respect kinematic constraints, and by sampling directly from point clouds one can for example avoid required computations over high-resolution grids or explicit terrain surface reconstruction. Much work has been done to extend sampling-based planners for UGV motion planning on terrain. However, these approaches only provide sub-optimal solutions. Researchers have combined grid-based and sampling-based methods to obtain cost-optimal and feasible paths [8].

1.1.3 Hybrid Methods

The Hybrid A* algorithm was created and used by the Stanford Racing Team in their entry to the DARPA Urban Challenge [49]. The method consists of two phases. An A* grid-search is used to guide a continuous-state tree search, and local optimization according to an objective function and use of a gradient descent method improves the sub-optimal, feasible paths generated by the hybrid search. The search is guided by two heuristics. One heuristic assumes no obstacles and is based on computing the shortest paths from every point to the final pose in a discretized neighborhood of the goal (i.e., cost-to-go), while accounting for non-holonomic constraints. The heuristic itself is comprised of a maximum of the computed cost and 2D Euclidian distance. The second heuristic does not account for constraints and uses dynamic programming in 2D to compute the shortest distance to the goal based on an obstacle map. By expanding vertices with motion primitives, the generated paths are feasible compared to paths generated by the original A* algorithm. The benefit of the first heuristic is a pruning of tree branches so that the vehicle does not approach the goal with wrong heading, while the second heuristic guides the tree search away from areas that cannot be traversed by the vehicle (in an urban setting) [49]. A gradient descent method is needed to improve on generated paths due to a merging of states occupying same grid-cells. The resulting paths generated by Hybrid A* are kinematically feasible and near-optimal, but as the method was designed for urban environments, it does not consider terrain characteristics such as elevation, inclination or roughness.

Tompkins et al. created a PRM-based A* motion planner for an experimental Mars rover using an a priori point cloud and occupancy grid to generate paths through user-defined waypoints by minimizing a terrain-based cost function [50]. Similar to PRM, the algorithm

relies on a roadmap that is generated a priori from randomly generated vertices in unoccupied space. The cost function, composed of height and slope features of the terrain, is used to verify that edges correspond to traversable paths. Another function is used to query the occupancy grid to ensure points (or vertices) are not situated on large static obstacles. The algorithm maintains a set of valid vertices (corresponding to unoccupied points) and an adjacency list containing traversable edges connecting a maximum number of neighbours to the valid vertices. Terrain characteristics of vertices are assessed by fitting planes covering local areas of points surrounding them. Most calculations are done offline, but there is no guarantee that the roadmap contains the user-defined waypoints as vertices are randomly generated over unoccupied space. An online weighted-graph A* solver, temporarily extends the directed graph with the waypoints and finds a minimum-cost path between them. Minimizing cost ensures that the rover avoids terrain with features corresponding to user-defined thresholds (e.g., heights corresponding to portions of the wheel diameter of the rover). Compared to Hybrid A*, this approach relies on a priori environment information due to the incorporation of PRM, and therefore, may be less applicable for planning in unknown environments. With regards to motion planning on terrain, an advantage of this method compared to Hybrid A* is that it considers terrain traversability.

Jordan et al. introduced another hybrid method for differential-drive wheeled vehicle motion planning on terrain [51]. This method uses a detailed vehicle model, which enables predictions on wheel-terrain interaction, the vehicle's 3D pose and chassis collision. Jordan et al. provide a scoring function that is used for a search similar to A*, by weighing these predictions according to the vehicle's required safety. In their publication, they argue that while the use of a simplified vehicle model enables more efficient planning than what may be acheivable with a high-fidelity model, there may exist fully executable paths that are not considered as a result. According to Jordan et al., the incorporation of a traversability analysis based on an advanced vehicle model enables traversal of complex obstacles, such as ramps only consisting of a pair of narrow lanes. In the presented method, the approach in [52] is used to estimate the orientation of the vehicle on elevation maps that are generated from environment data obtained by short-ranged on-board sensors. The pose of the vehicle is used in a pose evaluation scheme which for example checks for chassis collision and evaluates all wheels. Results from the evaluation scheme is used to calculate safety

criteria such as the angle between the previous and current orientation and tip angle. The traversability of a path is determined by the results of the safety criteria calculation. In the local planning method, the search space corresponds to vehicle poses (i.e., the vertices). The vehicle model is used to expand vertices by applying velocities to the poses and verifying that resulting paths do not violate the safety constraints. A safety violation will result in no further expansion of a given vertex. All leaf vertices can be chosen as end poses of a complete path, and complete paths are obtained by going backwards from the leaf with highest score, to the initial pose. The method by Jordan et al. was demonstrated in a real-world scenario, and is shown to produce short and feasible paths on rough terrain.

Thoresen et al. developed Traversability Hybrid A* to ensure sufficiently long planning horizons and to account for optimality in the sampling search of Hybrid A* for planning on terrain [8]. In this approach, the A* grid heuristic of Hybrid A* is substituted for an FMM grid heuristic that explicitly considers dynamic terrain cost. FMM is chosen as a substitute due to its improvements over A*, such as its near-elimination of the directional bias caused by varied estimated distance accuracy due to limited possible movements between centers of grid-cells. FMM is applied over a nonuniform traversability cost map with the goal as a source point to find the cost of moving from the source to any other map point. Estimated traversability values are linearly mapped to cost through a function, which contains a parameter that can be adjusted to affect how different degrees of traversability are penalized in planning. By applying FMM over the cost map, a fast marching field is obtained and a path is found by following the negative gradient of this field from the starting point. The resulting path ensures that the accumulated cost is minimized. Thoresen et al. explain that by providing an estimate of the weighted traversability cost-to-go, the FMM grid heuristic ensures a long planning horizon. The continuous-state tree search of this approach also samples with motion primitives, like the original Hybrid A* method, which ensures kinematic feasibility. The tree search is similar to the one performed in Hybrid A*, but instead of a distance-based cost, the sampling search uses accumulated traversability cost along the motion primitives. Traversability Hybrid A* also includes a local optimization according to an objective function, but with an added term that maintains traversability when smoothing paths. The traversability map is in this approach obtained by creating inclination maps based on height differences in elevation maps, generated from point cloud data, and linearly

mapping inclination to traversability values. Traversability scores are weighted with cell height uncertainty, resulting in lower scores for cells with high uncertainty. This approach was tested through simulation and with a UGV in a real-world scenario on rough terrain, and is shown to produce near-optimal paths and more traversable paths compared to the original Hybrid A* algorithm.

By employing a grid heuristic to guide a sampling-based search, hybrid approaches can account for optimality when planning on terrain, while also respecting vehicle constraints. When sampling-based methods are used for terrain planning, optimality is rarely a consideration due to (for example) the probabilistic nature of the algorithms, and the complexity of the environment. The development of suitable motion planning approaches for outdoor terrain scenarios relies on a number of factors, which may ultimately determine whether their applications succeed in efficiently providing sufficiently safe, executable paths. These factors include (but are not limited to) how cost functions are formulated, which metrics are chosen, how paths are optimized, how the vehicle and the environment are modeled, and how environment uncertainty is addressed. When planning on terrain, the optimal path is not necessarily the shortest path, but may be the path that optimizes accumulated traversability, as demonstrated in [8].

1.1.4 Traversability Costs

When binary obstacle maps are used for motion planning, distance is typically the only optimization parameter [8]. Optimization on distance alone is normally not sufficient for motion planning on terrain [5]. For a UGV travelling on terrain, the shortest-distance path may not be optimal due to terrain characteristics. Traversability maps can store information on features of the terrain such as inclination, elevation and roughness and enable formulation of cost functions for optimization of other parameters than distance (for instance, to find paths with least inclination) [8]. There are also approaches to traversability assessment which do not rely on explicit maps of the planning environment.

Solveig Bruvoll presented a concept for situation dependent path planning, which enables optimization on multiple simultaneous factors, in addition to distance [5]. The approach was demonstrated through simulation on different terrain types using the A* algorithm,

and takes several aspects into account such as time, accessibility, concealment and cover. This method can be altered and extended to account for other aspects, depending on the requirements of a given situation. To construct a graph that is suitable for planning with A*, the concept includes a path data generation process. This process analyzes terrain data to generate a navigation mesh (consisting of polygons), covering only areas accessible to a given autonomous entity. Accessibility of an area depends on factors such as type of terrain (e.g., forest, grass, swamp ocean or lake), environment elevation data, maximum acceptable slope and specified soil types accessible to an entity (e.g., soft soil, rock, or sand). The navigation mesh is discretized into a graph of vertices and edges corresponding to a subset of the accessible areas. To obtain a suitable cost function, a set of weights is first computed for every aspect considered relevant for the current planning task. For example, edge weights related to time are computed based on the distance between two vertices and the terrain type at the second vertex. The final cost is obtained through a linear combination of weights from all aspects with exponential coefficients. The exponents can be adjusted to generate paths that are optimized based on aspect prioritization.

Tahirovic and Magnani developed a version of RRT that uses a roughness-based metric instead of Euclidian distance when finding nearest neighbors from new states in constructing the space-filling tree [53]. The roughness metric is calculated for every square patch in an elevation map of the environment using an approach by Iagnemma and Dubowsky [54]. This approach estimates roughness by taking the standard deviation of the elevation corresponding to each cell within a patch. The standard deviation is normalized by vehicle wheel diameter to account for varying levels of mobility, depending on vehicle size. Tahirovic and Magnani adjusted the method by adding a friction coefficient, to avoid patches that are not traversable due to small coefficient values. In addition, the method computes a function that provides estimates of the roughness-to-go (i.e., the cost-to-go) for every terrain patch. To expand the space-filling tree, a linear combination between the roughness-so-far (i.e., the cost-so-far) to a new state and the roughness-to-go from the new state to the goal is used as cost. The result is an efficient exploration of the terrain that leads to tree expansion toward less rough areas.

In [9], Pérez-Higueras et al. presented a planning framework for RRT*, including a traversability assessment scheme that works directly on point cloud data. This feature makes the approach similar to [36] and [37]. The approach taken by Pérez-Higueras et al. employs spheres large enough to circumscribe the robot in assessing the terrain surrounding a given candidate point for tree expansion. The cost used for evaluating the terrain within a spherical region is similar to the one used in [5], as it is a weighted linear combination of features deemed relevant for a planning task. The costs chosen by Pérez-Higueras et al. to evaluate the terrain surrounding a candidate vertex are the pitch angle, roll angle and roughness of the surface, number of points in the sphere, distance between the candidate point and the mean of the points inside the sphere, the standard deviation of the point set inside the sphere and the distance from the candidate to the goal. Pérez-Higueras et al. use the number of points in the sphere as an indication of reliability of the surface representation. The standard deviation of points inside a sphere gives information about how points are dispersed within the region. Distance between the candidate and the mean is used as an indication of poorly represented environement regions. When these costs are used for evaluation of the terrain surrounding candidate points, the result is that paths to the leaves of the tree are optimized with respect to the chosen assessment criteria. To choose between the leaves, the cost used is a weighted linear combination of the terrain assessment cost and distance to the leaf, a frontier cost and a return cost. The frontier cost is a weighted linear combination of the number of points in the sphere surrounding a candidate leaf with the standard deviation of points inside the sphere. Distance between the leaf and the closest point in the path followed by the robot so far is used as the return cost, to prevent the robot from revisiting areas near already visited areas.

Several approaches are available for determining terrain traversability in motion planning. A number of approaches generate maps of the environment based on the available sensor data to perform traversability analysis, while others use only the data in locally surrounding regions of a robot, performing this analysis on-demand, with the aim of reducing computational cost during planning. The criteria used for optimization depend on the motion planning task. When motion planning is performed on terrain, a traversability assessment is required [55], to ensure safety with respect to the motion capabilities of a UGV, and to enable optimization on appropriate features for terrain traversal.

1.1.5 Problem Statement

The problem statement in this section is motivated by the research effort that has been given to extend the Rapidly-exploring Random Tree (RRT) and Rapidly-exploring Random Tree* (RRT*) algorithms for Unmanned Ground Vehicle (UGV) motion planning on terrain and the improvements and applicability of the Fast Marching Method (FMM) compared to other grid-based methods in complex environment scenarios. The problem of establishing suitable motion planning methods for UGVs on terrain is yet to be solved [56]. The aim of the thesis will be to investigate this problem by addressing the following research questions:

1. How can RRT and RRT* be adapted to accommodate UGVs with limited maneuvering capabilities, such as the absence of neutral-turn and reverse motion capabilities, and how do these adaptations influence path traversability?
2. Does FMM guarantee kinematic feasibility for vehicles with limited maneuvering capabilities, operating in terrain scenarios?
3. Can FMM be effectively utilized as a benchmark for evaluation of traversability-based RRT and RRT* schemes in UGV motion planning on terrain?

1.1.6 Thesis Outline

There are five main parts to this thesis. Section 2 presents relevant motion planning background, and provides the necessary foundation for subsequent sections. The first main contribution is given in Section 3. In this section, a novel sampling approach for traversability mapping is described. Section 4 contains the second main contribution, including a description of motion planning schemes and algorithms, along with the development of motion primitive schemes for RRT and RRT*, a collision-checking scheme, an objective cost function, and a simple non-holonomic version of FMM. Section 5 presents the motion planning simulation approach, simulation results and a discussion of these results. The thesis is concluded in Section 6, which provides suggestions for future work.

2 Background

To enable UGV motion planning on terrain, a feasibility scheme is essential. Section 2.1 describes a differential-drive kinematic model, for which the autonomous platform developed at FFI (i.e., described in Section 1) is used as an example vehicle. Establishing solutions satisfying constraints of this vehicle is of interest to FFI, and this is the motivation for utilizing the differential-drive model. Section 2.2 defines a central concept to motion planning, namely, the notion of a configuration space. Section 2.3 adds context to the research field, through a high-level overview of autonomous decision-making. The optimal, grid-based, Fast Marching Method (FMM) is introduced in Section 2.4. This section gives an understanding of how FMM produces optimal paths in a given cost field, and this understanding is essential for sections 4, 5 and 6. Section 2.5 introduces the sampling-based Rapidly-exploring Random Tree (RRT) algorithm, and its AO counterpart. Understanding the workings of these algorithms is important, as these algorithms are used together with the vehicle model in Section 2.1 in the development of motion planning schemes, utilized to enable feasible solutions in terrain scenarios.

2.1 Modelling the THeMIS UGV with Differential-Drive Kinematics

The autonomous UGV platform developed by Mathiassen et al. is based on the Milrem Tracked Hybrid Modular Infantry System (THeMIS) 4.5 from Milrem Robotics [10]. The vehicle is shown in Figure 1.

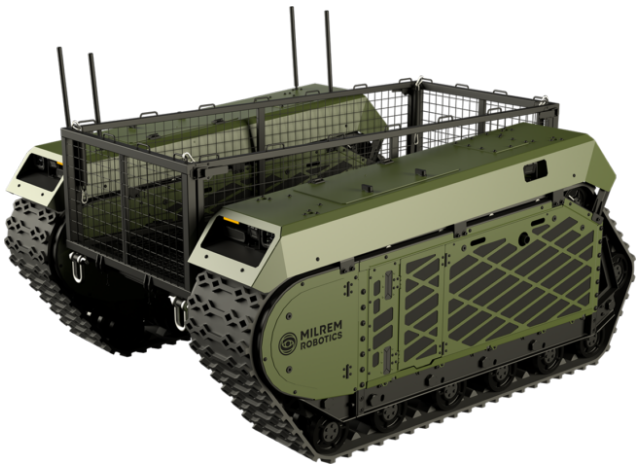


Figure 1: Milrem THeMIS UGV (Image source: [57])

In this thesis, the Milrem THeMIS UGV is represented by the simple differential-drive kinematic model [58]:

$$\dot{x} = \frac{(v_r + v_l)}{2} \cos\theta \tag{1}$$

$$\dot{y} = \frac{(v_r + v_l)}{2} \sin\theta \tag{2}$$

$$\dot{\theta} = \frac{(v_r - v_l)}{L} \tag{3}$$

where $\dot{x}$, $\dot{y}$ and $\dot{\theta}$ are the linear velocities in the x and y directions and the angular velocity of the vehicle respectively. v_r corresponds to the linear velocity of the right track, whereas v_l corresponds to the linear velocity of the left track. L is the length of the trackbase, and θ is the orientation of the vehicle with respect to the x-axis. This formulation allows the vehicle to be represented as a point in planning (please refer to Section 2.2). Figure 2 is an illustration adapted from [45], showing the length of the trackbase and the left and right linear velocities. The red dot represents the center of mass of the vehicle.

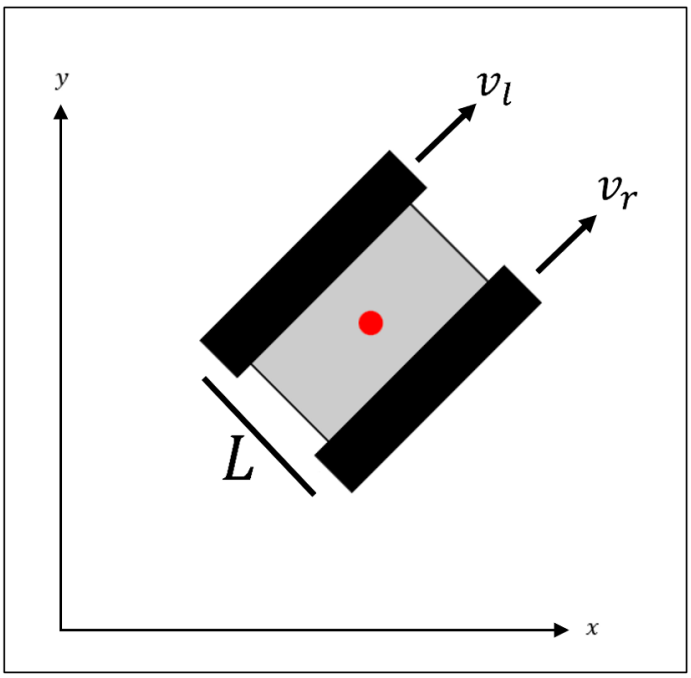


Figure 2: Illustration of the simple differential-drive model. v_l is the linear velocity of the left track, v_r is the linear velocity of the right track, L is the segment length corresponding to the trackbase, and the red circle represents the center of mass.

The non-holonomic constraint of the differential-drive model is given by [14]:

$$\dot{x} \sin \theta - \dot{y} \cos \theta = 0 \tag{4}$$

This equation implies that there is no movement in the direction parallel to the trackbase.

There are several limitations of representing the vehicle using a simple kinematic differential-drive model in motion planning. Through their work, Jordan et al. highlight that difficult paths such as those involving a pair of narrow lanes may be discarded when using a simple model [51]. Pepy, Lambert and Mounier state that for kinematic car-like models, it is assumed that the vehicle can move without skidding, and this implies that the vehicle is following the path at low speeds [59]. This also applies for the differential-drive model, as it assumes that vehicle has perfect traction, meaning that there are no skidding or slippage effects. These effects are apparent for the THeMIS UGV. For this vehicle, there is little movement without slippage.

Pepy et al. note that when the motion planning scheme is based on simple kinematics, and a solution path is found, the control algorithm (i.e., in a path following module), must correct significantly for the position of the vehicle so that it can follow the path. Furthermore, the authors highlight advantages through a side-by-side comparison of kinematics and dynamics in RRT, suggesting that the incorporation of dynamics in motion planning enables solution paths that are more realistic with respect to obstacles in the environment. However, they also state that considering dynamics in motion planning can complicate the planning problem. The authors mention that it is, for example, impossible to generate Dubin's curves [60] that obey non-holonomic and dynamic constraints. Canny et al. conjecture that if contact is allowed, the complexity of a kinodynamic planning problem increases significantly [61]. The reason for using a kinematic differential-drive model to represent the vehicle as a point in this thesis, is that it simplifies the problem of developing a motion planning scheme that ensures feasibility. This is sufficient for simulation purposes, but for real-world experiments, more sophisticated approaches may be required.

2.2 Configuration Space

Motion planning would be difficult and computationally expensive, if planning could only be done in a robot's workspace. To calculate a path for a differential-drive robot in its workspace, one must calculate the path by planning the motion of each point of the vehicle's shape according to its degrees of freedom, to ensure obstacles are avoided (i.e., the vehicle has a footprint) [62]. There is a central concept in motion planning that reduces this problem to a simpler one. It is possible to represent a robot by a point according to its state variables in a different space, that defines all admissible and inadmissible robot configurations. To clarify, inadmissible robot configurations correspond to states the robot cannot assume, due to the obstacles in a planning environment. The space used to represent admissible and inadmissible vehicle configurations corresponding to a given workspace is called the configuration space (or c-space).

For the differential-drive model, a state is given by $[x, y, \theta]^T$, and this is a point in its c-space for a given workspace. Hence, the c-space for a given workspace corresponding to the model presented in Section 2.1 is a subset of the Special Euclidean group SE(2) (i.e., the group of all 2D translations and counter-clockwise 2D rotations).

2.3 Motion Planning and Autonomous Decision-Making

Given the definition of the c-space in Section 2.2, the motion planning problem is reduced to computing a set of admissible configurations that takes a vehicle from a source to a destination. According to Paden et al., at a high level, autonomous decision-making is a hierarchy that enables an autonomous vehicle to accomplish driving objectives, by enabling an automatic selection of controlled variable values, governing a vehicle's motion based on sensor data and prior knowledge of a planning environment [63]. Hence, to perform motion planning, a map of the environment is required. The output of the motion planning module is a reference path that is used as input to a path following scheme (i.e., an algorithm responsible for local feedback control). For example, the output of the path following scheme can be linear velocity commands that guide a UGV along a reference path. In this context, the focus of this thesis is on producing a map of an environment based on sensor data and on generating reference paths that can be used as input to a control scheme.

2.4 Fast Marching Method

The Fast Marching Method (FMM), presented by Osher and Sethian, is an efficient numerical technique for propagating a wavefront through one or more mediums. To update the values of cells in a grid map as the wavefront propagates from a source point, FMM iteratively solves a nonlinear partial differential equation called the Eikonal equation [64]:

$$1 = F(\mathbf{x})|\nabla T(\mathbf{x})| \tag{5}$$

This equation consists of a speed function F (always positive), and the gradient magnitude of a time-of-arrival field T, both with dependency on the position $\mathbf{x}$. By considering a single point $\mathbf{x}$, the intuition behind this equation becomes clearer. In this case $F(\mathbf{x})$ corresponds to the speed of the wavefront at $\mathbf{x}$, and $T(\mathbf{x})$ corresponds to the time it takes for the wavefront to reach $\mathbf{x}$ from the source point of the expansion. Dividing by F on both sides, gives:

$$\frac{1}{F(\mathbf{x})} = |\nabla T(\mathbf{x})|$$

According to the Eikonal equation, the magnitude of the gradient of the time-of-arrival field at a point is inversely proportional to the speed at the same point. For example, the speed of a wavefront in water will be the same everywhere in the medium, provided there are no obstacles (i.e., assuming no disturbance). Hence, if the propagation is frozen, then at all points along the circular wavefront, the T-values are equal. As the wavefront propagates, the T-values will increase the further it moves from the source point. The effect of propagating through oil and water will be that the wavefront has lower speed values in the oil compared to the water, which means the time-of-arrival field will reflect that it takes longer for the wavefront to propagate through regions characterized by lower speeds. In this case, regions consisting of oil correspond to domains of higher cost, and regions consisting of water correspond to domains characterized by lower cost (i.e., with respect to T). A shortest-time path can be found by first expanding the wave from a source point to a goal point and then performing gradient descent in T from the goal to the source point. If the wave expands in a single medium, the fastest path will also be the path with the shortest distance. A benefit of the increasing T-values from the source point is that this method,

in contrast to (e.g.) Artificial Potential Field (APF) methods, is not susceptible to getting stuck in local minima [8]. The time-of-arrival field will only present a global minima at the source point.

Using finite difference approximations for the partial derivatives of T, Osher and Sethian propose the following upwind scheme as a discretization of the Eikonal equation on a 2D grid (with $\Delta_{x \vee y}$ denoting the grid map resolution, assuming quadratic cells) [64]:

$$\max(D_{ij}^{-x}T, 0)^2 + \min(D_{ij}^{+x}T, 0)^2 + \max(D_{ij}^{-y}T, 0)^2 + \min(D_{ij}^{+y}T, 0)^2 = \frac{1}{F_{ij}^2} \tag{6}$$

The incorporation of min and max functions ensures that only information from the direction the wavefront has traveled from, is incorporated into the gradient magnitude of T (i.e., the desired time-of-arrival field reflects the fastest path through one or more mediums). A simpler, but less accurate discretization is [64]:

$$\max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 + \max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2 = \frac{1}{F_{ij}^2}, \tag{7}$$

where

$$D_{ij}^{-x} = \frac{T_{i-1,j} - T_{i,j}}{-\Delta_{x \vee y}},$$

$$D_{ij}^{+x} = \frac{T_{i+1,j} - T_{i,j}}{\Delta_{x \vee y}},$$

$$D_{ij}^{-y} = \frac{T_{i,j-1} - T_{i,j}}{-\Delta_{x \vee y}},$$

$$D_{ij}^{+y} = \frac{T_{i,j+1} - T_{i,j}}{\Delta_{x \vee y}}$$

Substituting this into Equation 7, yields:

$$\max\left(\frac{T_{i,j} - T_{i-1,j}}{\Delta_{x \vee y}}, \frac{T_{i,j} - T_{i+1,j}}{\Delta_{x \vee y}}, 0\right)^2 + \max\left(\frac{T_{i,j} - T_{i,j-1}}{\Delta_{x \vee y}}, \frac{T_{i,j} - T_{i,j+1}}{\Delta_{x \vee y}}, 0\right)^2 = \frac{1}{F_{ij}^2},$$

and by setting

$$T = T_{i,j},$$

$$F = F_{i,j},$$

$$T_1 = \min(T_{i-1,j}, T_{i+1,j}),$$

$$T_2 = \min(T_{i,j-1}, T_{i,j+1}),$$

the equation can be simplified to

$$\max\left(\frac{T - T_1}{\Delta_{x \vee y}}, 0\right)^2 + \max\left(\frac{T - T_2}{\Delta_{x \vee y}}, 0\right)^2 = \frac{1}{F^2} \tag{8}$$

Equation 8 is solved in three steps [26]:

1. If the greater T-value obtained by solving the following is larger than T_1 and T_2:

$$\left(\frac{T - T_1}{\Delta_{x \vee y}}\right)^2 + \left(\frac{T - T_2}{\Delta_{x \vee y}}\right)^2 = \frac{1}{F^2},$$

   then this T-value is the correct solution.

2. If $T < T_2$ in Step 1, then the correct solution is obtained by solving

$$\left(\frac{T - T_1}{\Delta_{x \vee y}}\right)^2 = \frac{1}{F^2}$$

3. If $T < T_1$ in Step 1, then the correct solution is obtained by solving

$$\left(\frac{T - T_2}{\Delta_{x \vee y}}\right)^2 = \frac{1}{F^2}$$

Hence, the Eikonal equation can be solved for T when a speed function has been specified. For example, setting F to a constant value implies a constant speed everywhere on the grid, which corresponds to wavefront propagation through a single medium. Therefore, this speed can be adjusted to ensure prioritization of paths (i.e., produced by gradient descent through the resulting field T) adapted to a given planning scenario. In Section 4.3, an objective inclination-based cost function is presented. This cost function works a substitution for the inverse of the speed function. The T-value obtained by following the steps above, corresponds to the output of *solveEikonal(g_{kl})* in the pseudocode for the FM algorithm on the following page.

Algorithm 1 Fast Marching Algorithm [26]

Require: Gridmap G of size $m \times n$
Require: Set of cells Ori where the wave is originated
Ensure: Gridmap G with the T value set for all cells

```
Initialization;
for all g_ij ∈ Ori do
    g_ij.T ← 0
    g_ij.state ← FROZEN
    for all g_kl ∈ g_ij.neighbours do
        if g_kl = FROZEN then
            skip
        else
            g_kl.T ← solveEikonal(g_kl)
            if g_kl.state = NARROW_BAND then
                narrow_band.update_position(g_kl)
            end if
            if g_kl.state = UNKNOWN then
                g_kl.state ← NARROW_BAND
                narrow_band.insert_in_position(g_kl)
            end if
        end if
    end for
end for
Iterations;
while narrow_band NOT EMPTY do
    g_ij ← narrow_band.popfirst()
    for all g_kl ∈ g_ij.neighbours do
        if g_kl = FROZEN then
            skip
        else
            g_kl.T ← solveEikonal(g_kl)
            if g_kl.state = NARROW_BAND then
                narrow_band.update_position(g_kl)
            end if
            if g_kl.state = UNKNOWN then
                g_kl.state ← NARROW_BAND
                narrow_band.insert_in_position(g_kl)
            end if
        end if
    end for
end while
```

The FM algorithm starts by accepting a grid map with all cells set to infinity. An initialization procedure ensures the source of the wave is set to the state *FROZEN* (i.e., the state indicating that wave has already passed over a cell), and the Eikonal equation is solved for every neighbor of the source cell(s). Upon receiving a T-value, the state of the neighbor is checked. If the state is *NARROW_BAND*, then this cell is part of the wavefront, and the position of the cell in a min heap used to maintain this front is updated to preserve the correct structure (i.e., with the minimum T-value on top, and decreasing values from top to bottom). The state *UNKNOWN* implies the cell has not yet been considered, and should be added to the set of cells corresponding to the wavefront. After the initialization procedure, the main loop is executed. This loop is similar to the loop in the initialization procedure. First, the cell in the wavefront with minimum T-value is considered. Then, for every neighbor, the Eikonal equation is solved, a T-value is received, states are checked and corresponding updates are applied as before. This procedure is repeated until there are no cells left in the wavefront (i.e., all cells have been updated).

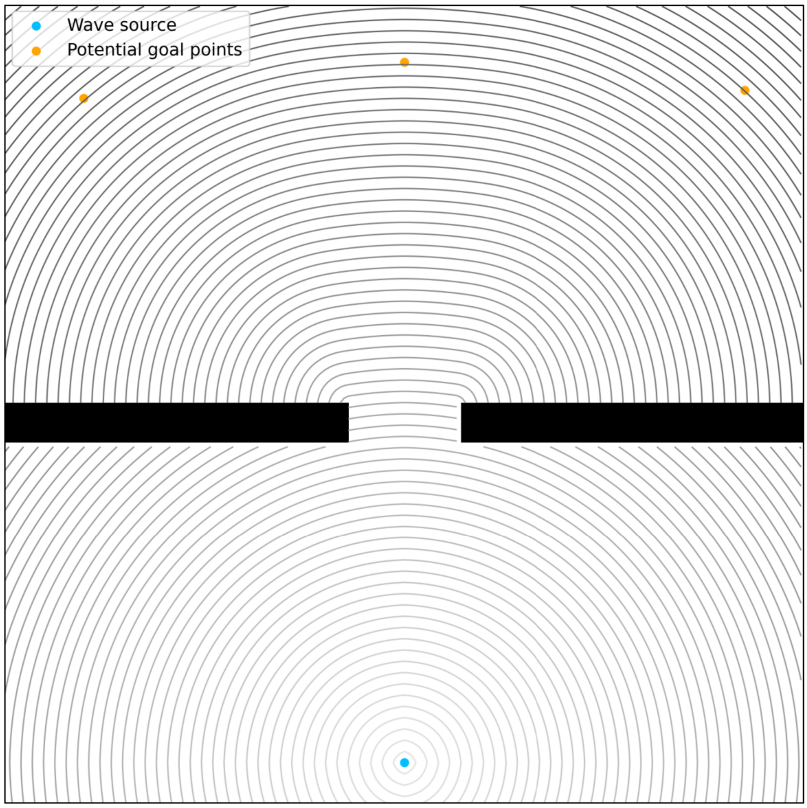


Figure 3: FM algorithm applied to a binary obstacle map. The contour lines represent the level sets of the time-of-arrival field. In this case, the wave is expanded from a cyan source point.

FMM is similar to Dijkstra's algorithm, but nearly eliminates its grid bias due to the incorporation of a local solution of the Eikonal equation in place of Dijkstra's graph update [65]. The update in FMM is more consistent with the underlying continuous space, and is therefore expected to produce more optimal paths than that of Dijkstra's algorithm [66]. Figure 3 shows a simple wave diffraction example in a binary obstacle map, produced by FMM. From any point in the white region of the space, an optimal path to the source can be found by following the maximum gradient of the field. In this case, because FMM has been expanded over a binary obstacle map, this corresponds to wavefront propagation through a single medium (i.e., the speed function is constant everywhere in the white region, and zero in the black regions).

Figure 3 is an example used to illustrate the workings of the FM algorithm, but for the proposed algorithms presented in this thesis, the wave is expanded from the goal. The result is that gradient descent is computed in the direction of travel. Thoresen et al. state that by expanding from the goal, the FM algorithm can be used to provide a weighted estimate of the traversability cost-to-go [8]. This is the motivation for using FMM as a benchmark algorithm in Section 5 (please refer to Section 4.3).

2.5 Rapidly Exploring Random Trees

Steven M. Lavalle originally developed Rapidly-exploring Random Trees (RRTs) to facilitate efficient searches in high-dimensional spaces with algebraic and differential constraints [67]. RRT samples points in a search-space to bias exploration toward large unsearched regions by incrementally growing the tree toward the sampled points [33]. The algorithm is suitable for handling differential constraints, as it enables a continuous-state search, in contrast to a grid-based search. Pepy et al. note that a differentially constrained model can be directly integrated in the algorithm to obtain feasible connections between two configurations, as the construction of the tree is incremental [59]. An example expansion with motion primitives is shown in Section 4.1.

2.5.1 Rapidly Exploring Random Tree

The RRT algorithm starts by first initializing the root of the tree at a start position. A node is randomly sampled from the search space, and an attempt is made to extend the tree toward the random sample by a length ϵ. In the event that the randomly sampled node is within ϵ from the root, this node is added to the tree. If the algorithm is not subject to differential constraints, an extension must be checked to ensure it is collision-free. Otherwise, the extension must satisfy the constraints of a given vehicle in addition to obstacle avoidance. The collision-detection module used in both RRT and RRT* is described in Section 4.2. The algorithm proceeds to the next iteration if no extension is possible. When the new node has been added, a new configuration is sampled, and the nearest node in the tree to the randomly sampled node is chosen as a parent for extension towards the new random sample. The algorithm continues until the maximum number of iterations is reached, or the goal is found. If the goal is reached by the tree after a number of iterations, a path can be found by backtracking over branches from the goal, through the parent nodes in the tree, until the root is found. **Algorithm 2** is the pseudocode for the RRT algorithm.

Algorithm 2 Rapidly-Exploring Random Tree (RRT) Algorithm [33]

```
BUILD_RRT(x_init):
    T.init(x_init);
    for k = 1 to K do
        x_rand ← RANDOM_STATE();
        EXTEND(T, x_rand);
    Return T
```

```
EXTEND(T, x):
    x_near ← NEAREST_NEIGHBOR(x, T);
    if NEW_STATE(x, x_near, x_new, u_new) then
        T.add_node(x_new);
        T.add_edge(x_near, x_new, u_new);
        if x_new = x then
            Return Reached;
        else
            Return Advanced;
    Return Trapped;
```

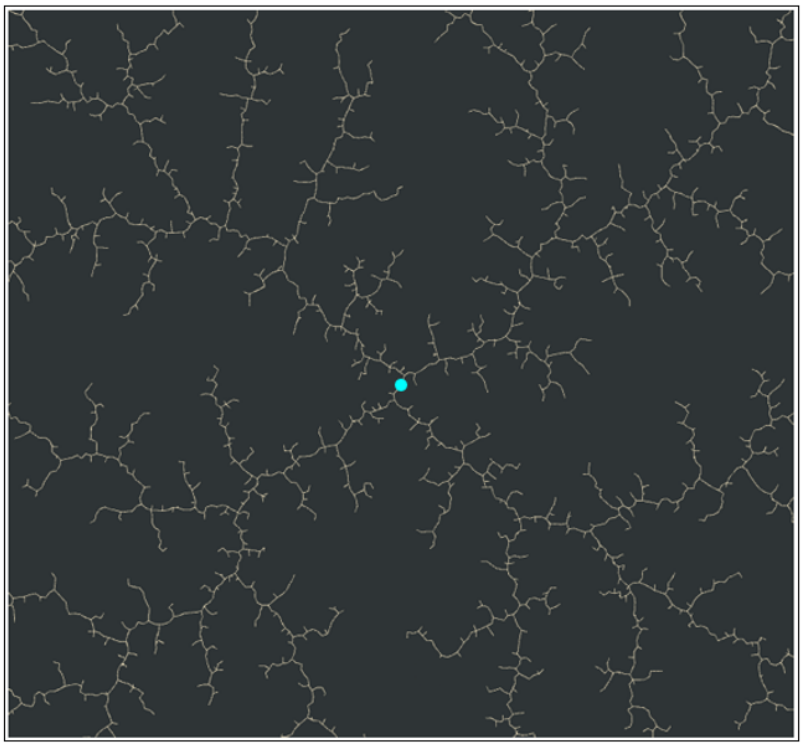


Figure 4: Application of the Rapidly Exploring Random Tree (RRT) algorithm, with the root node represented in cyan. The tree is constructed in an obstacle-free environment.

2.5.2 Rapidly Exploring Random Tree*

A limitation of the RRT algorithm is that it does not improve on intermediate solutions, and as a result, solution paths are generally sub-optimal. To address this problem, Sertac Karaman and Emilio Frazzoli proposed an algorithm called RRT* [35]. This algorithm is an asymptotically optimal variant of RRT, meaning that it converges to an optimal solution as the number of samples increases. The difference between the algorithms is that RRT* includes optimization steps based on neighbors in a radius of a newly added node to improve on intermediate solutions as the tree is expanded. There are two main steps in the optimization procedure. Firstly, when a node is added to the tree, all nodes in a radius from the new node are evaluated to determine if a better connection exists, compared to the connection that is obtained between the nearest neighbor to a random sample and the new node. In the default algorithm, a "better" connection corresponds to one that results in a shorter distance between the new node and the root. Secondly, when the new node is connected to the tree, all neighbors of the new node are evaluated to determine if the neighbors can be connected to the new node to obtain shorter distances between the

neighbors and the root, compared to the current distances. RRT* incrementally rewires the tree to obtain higher quality solutions, compared to RRT.

A result of improving on intermediate solutions during tree building is that paths computed with RRT* are straighter than those that can be computed with RRT. This can be seen by comparing the trees in Figure 4 (Section 2.5.1) and Figure 5. The cyan circles represent the root nodes of the trees. The straighter branches highlight that by picking a configuration in the search-space, a shorter path to the root is almost surely obtained by RRT*.

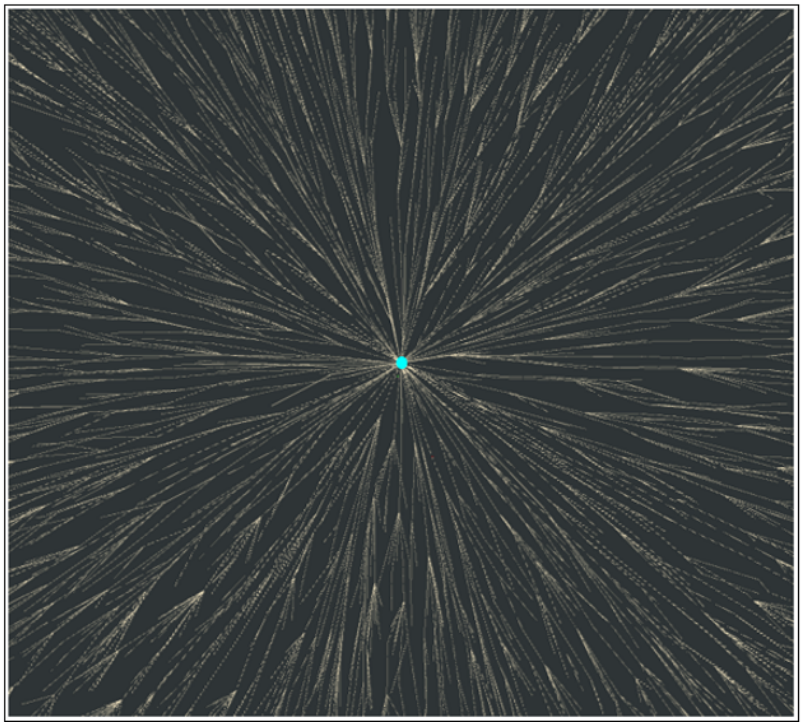


Figure 5: Application of the Rapidly Exploring Random Tree* (RRT*) algorithm, with the root node represented in cyan, without obstacles.

3 Traversability Mapping

The autonomous UGV platform presented in Section 2.1 is equipped with a Light Detection and Ranging (LIDAR) sensor. This sensor can be used to produce a point cloud of the terrain surrounding the UGV. The point cloud is, in this case, a set of 3D points providing data that gives information about the shape of the terrain surface at different elevations. This data can be used to assess the traversability of the terrain surrounding the UGV, and to enable motion planning, a traversability assessment is required [55].

Several approaches to conducting traversability assessment based on point clouds exist. A common approach is to construct Digital Elevation Models (DEMs), for example, 2.5D elevation maps. According to Stölzle et al., occlusions may be handled poorly by such methods, as occluded areas may be classified as obstacles, or the missing values may be filled in by traditional interpolation methods [68]. Other approaches to traversability assessment involve local on-demand analysis of a point cloud based on robot-sized areas during motion planning, such as the approaches taken by Pérez-Higueras et al. [9] and Krüsi et al. [37]. Han et al. note that compared to grid-based methods, the approach by Krüsi et al. is more flexible and therefore more suitable for unstructured environments as complex geometries in outdoor scenarios, such as rocks or irregular bumps cannot be modelled adequately by traditional grid map methods [69]. An advantage of performing local on-demand analysis of a point cloud is that it may reduce the computational burden associated with generating and maintaining larger data structures [9]. However, some motion planning applications may rely on a complete map of the environment to ensure certain safety requirements are met, such as those related to travel time or risk [43].

3.1 Minimum Enclosing Cylinder

The concept of a minimum-enclosing cylinder is central to this thesis. This cylinder is computed based on the dimensions (i.e., length, width and height) of the THeMIS UGV, made available by Milrem Robotics [70]. The radius of a minimum-enclosing circle is calculated as the half-diagonal of a rectangle of dimensions corresponding to the length and width of the THeMIS UGV. This radius is relevant for the minimum-enclosing circles used in the motion planning schemes presented in Section 4. The minimum-enclosing cylinder that is relevant for this section, corresponds to a cylinder with a radius equal to the minimum-enclosing radius and a height equal to the platform height. This is a simplification as the platform may be fitted with equipment beyond the dimensions given in the specification. The presented scheme in this section can easily be extended to accommodate a cylinder of different dimensions, for example if additional safety precautions are required due to the characteristics of a given planning environment. This also applies for the minimum-enclosing circle in Section 4.

3.2 Point Cloud Dataset

The point cloud data set used to generate the inclination maps presented in Section 3.7, is downloaded from `hoydedata.no`, a web portal by the Norwegian Mapping Authority (Kartverket) [71]. This portal provides open-access to high-resolution terrain data for Norway. The point cloud dataset utilized in this thesis, is captured from an airborne LIDAR system. This specific data set is chosen, as it enables motion planning in an environment that consists of a range of different variations in obstacles and elevations. A wide range of different planning scenarios can be extracted based on this data set. Section 4 provides a description of chosen and randomly sampled scenarios used in motion planning simulation with respect to one of the inclination maps presented in section 3.7. Figure 6 gives a rough high-level overview of the point cloud from four different perspectives.

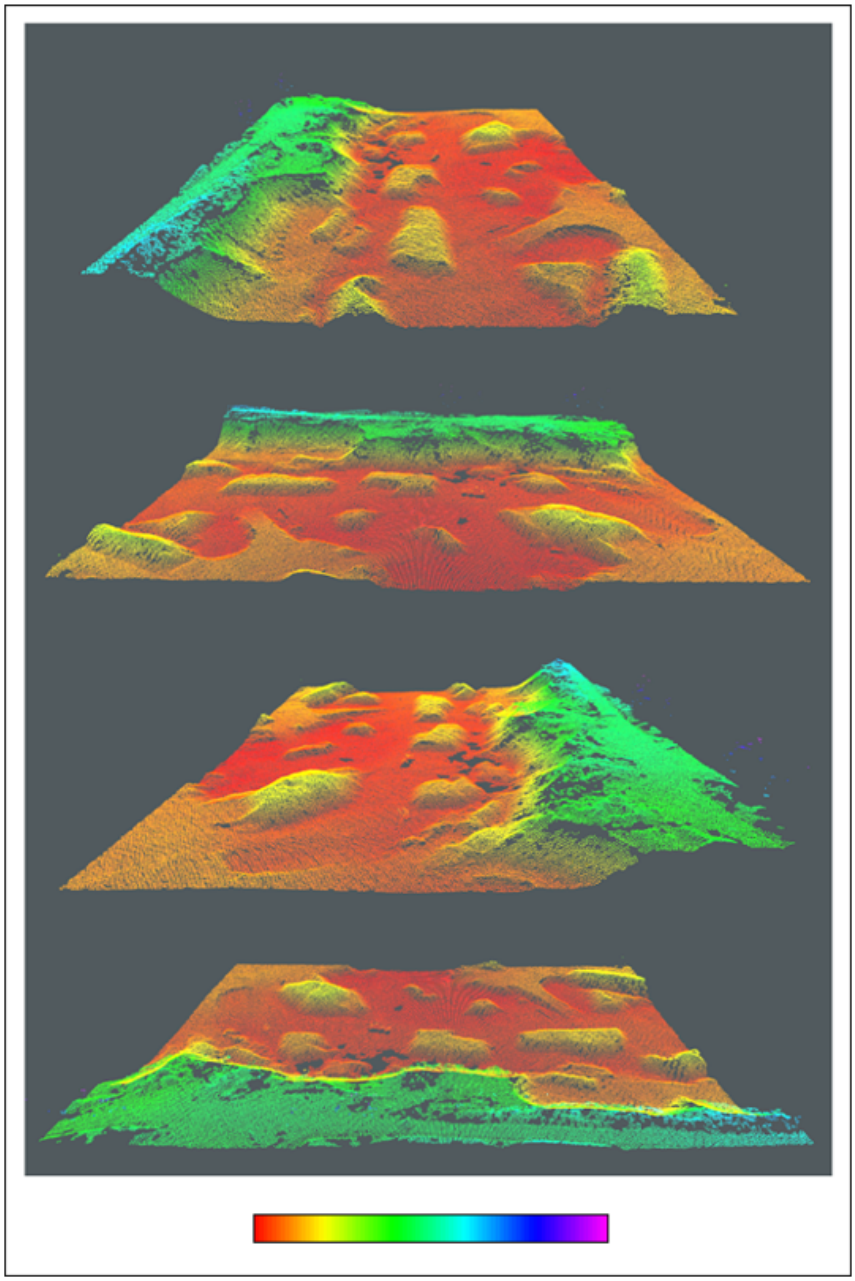


Figure 6: High-level overview of the point cloud dataset from four different angles. The color bar is a cropped HSV color palette, ranging from red to purple. The color red indicates points corresponding to a range of the lowest height values, while purple indicates points corresponding to the range of the greatest height values within the dataset.

3.3 Sampling Scheme

In this thesis, a grid-based traversability map is computed based on overlapping minimum-enclosing cylinders by a fixed radius over a point cloud and by performing a simple traversability assessment in each cylinder to save values in a grid map structure with quadratic cells. The overlapping radius corresponds to the minimum-enclosing cylinder radius. Based on the traversability assessment, the lowest recorded value is kept in cells corresponding to a single cylinder. In the context of the simple assessments that are done in sections 3.4 and 3.5, the lowest value corresponds to the lowest estimated inclination value. Hence, the cylinder value corresponding to the highest recorded traversability, with respect to inclination estimates, is kept. The base of a cylinder is set to be the same height as the point with lowest height value within the cylinder, before the cylinder height is set, to cover the entire span of the point cloud in the z-direction. This is done to enable traversal underneath obstacles. An illustration of the sampling pattern is shown in Figure 7.

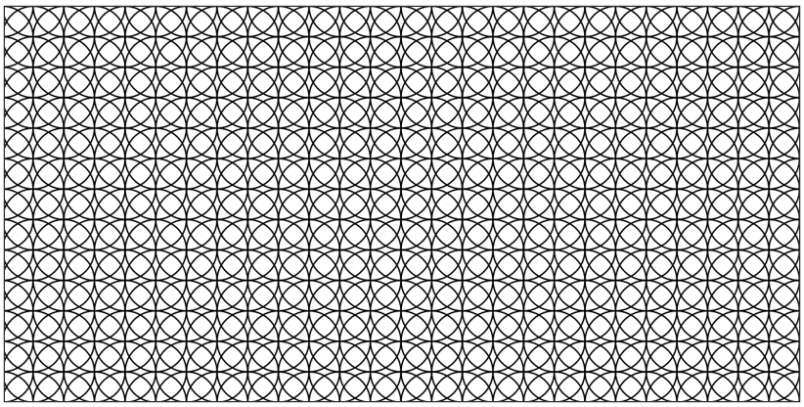


Figure 7: Illustration of the sampling pattern. Cylinders are overlapped by the radius of the minimum-enclosing circle described in Section 3.1, to cover the entire grid map.

An advantage of incorporating a minimum-enclosing geometric shape in conducting traversability assessment may be increased flexibility compared to traditional grid-based methods [69]. Incorporating additional traversability-based metrics to perform terrain assessment inside a cylinder, such as those relating to roughness [53], and point distribution [9], in addition to inclination-based metrics, may increase accuracy of traversability assessment for the current scheme. A motivation for overlapping cylinders is that it enables a reevaluation of traversability based on same and different points by considering different

neighborhoods of a set of points, before a final decision is made for a given cylinder. Another is that since the final map is a standard grid map, the overlap is necessary for a complete coverage of cells. In environments with complex features, an analyis based on a single local neighborhood of points may not provide enough information for accurate assessment of traversability, as indicated by Stölzle et al. [68]. On the other hand, a limitation of reevaluating traversability based on same points is that it may result in high computational cost, as in traditional grid map approaches [9].

3.4 Triangle-Based Approach

For the triangle-based approach, within a single cylinder, the point corresponding to the lowest elevation value, and the point of the highest elevation are found. As is shown in Figure 8, the points can be used to construct a right-angled triangle. Hence, an inclination, denoted by α, can be estimated roughly based on computing the arc-tangent of the absolute difference in elevation between the lowest and highest points (denoted by e) in the cylinder, divided by the 2D Euclidean distance between the two points (denoted by d). This method is repeated for all cylinders, and the lowest recorded inclination estimate for an individual cylinder is kept. The intuition behind this choice, is that if higher estimated inclinations are kept, then regions characterized by traversable slopes may be discarded. For less than two points within a cylinder, the corresponding cells are classified as obstacles. Consider two overlapping cylinders where the cells in one is classified as obstacles and the second is not, this will result in parts of the cells, initially classified as obstacles, being overwritten. This is related to the local variability of the terrain, and that reassessing traversability based on different neighborhoods can result in a different assessment of the same region.

A limit is enforced to ensure that α-values over a certain threshold are characterized as obstacles. The limit can be adjusted to allow for a wider range of admissible inclinations, or conversely, to limit the range of admissible inclinations. The height of the cylinder and the longest length of the the vehicle is used to compute the limit, in an attempt to avoid excessive conservatism. In this thesis, the threshold is chosen to ensure that the maximum attainable admissible e-value is approximately 50 cm, given that the THeMIS UGV has a ground clearance of 40 to 60 cm [70].

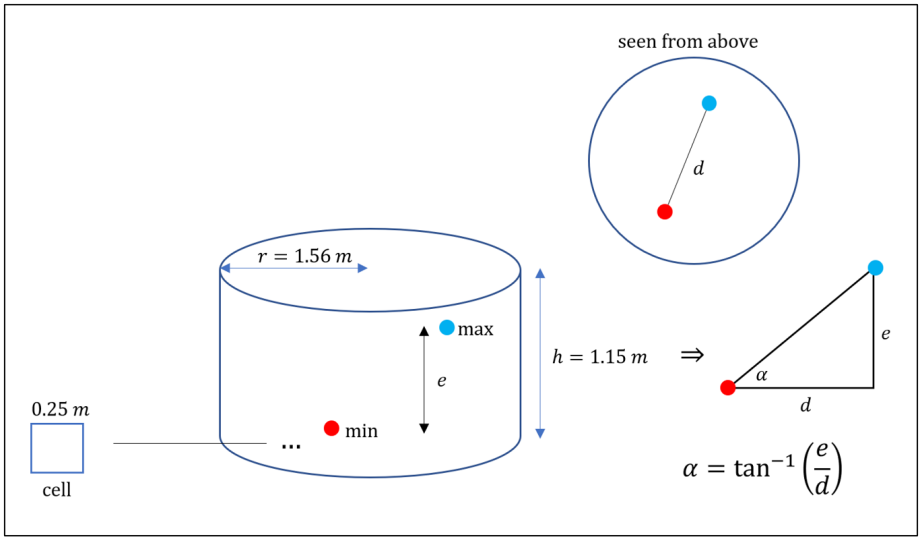


Figure 8: Illustration of the triangle approach used to estimate inclination values from point cloud data within a single cylinder. An angle (α) is computed based on the arc-tangent of the absolute difference in elevation between the lowest and highest point (e) within a cylinder divided by the 2D Euclidean distance between the two points (d). h denotes the cylinder height, and r denotes the radius of the cylinder.

Simply using a ground clearance to determine a threshold for obstacles, may induce collision, as obstacles in terrain scenarios can be highly non-linear. This can be seen by inspecting the point cloud in Figure 6, Section 3.2. Although the overlap between cylinders may capture some local variations, other variations may not be considered due to the limited number of points used to estimate inclination.

For real-world scenarios, more sophisticated approaches may be required for traversability assessment. The benefit of the sampling-based scheme is that the flexibility that is enabled by utilizing minimum-enclosing cylinders, allows for incorporation of a combination of metrics (e.g., a weighted linear combination of roughness, point distribution, variances of angles or other relevant quantities, similar to [9]) to estimate traversability within cylinders. The triangle-based estimation approach is chosen for simplicity, to shift the focus on motion planning, which is the main topic of this thesis. Another reason for this choice is that the approach in Section 3.5 does not appear to yield more accurate estimates of inclinations compared to the triangle-based approach (please refer to Section 3.8). However, a comprehensive comparison to existing methods is necessary. The triangle and surface normal methods are only compared to each other based on a rough visual inspection of the terrain data in Figure 6.

By saving inclination values in quadratic cells within and intersected by the borders of cylinder bases, a simple traversability map is obtained that gives a more nuanced representation of the planning environment compared to a traditional binary obstacle map, which is insufficient for motion planning on terrain.

3.5 Surface Normal Approach

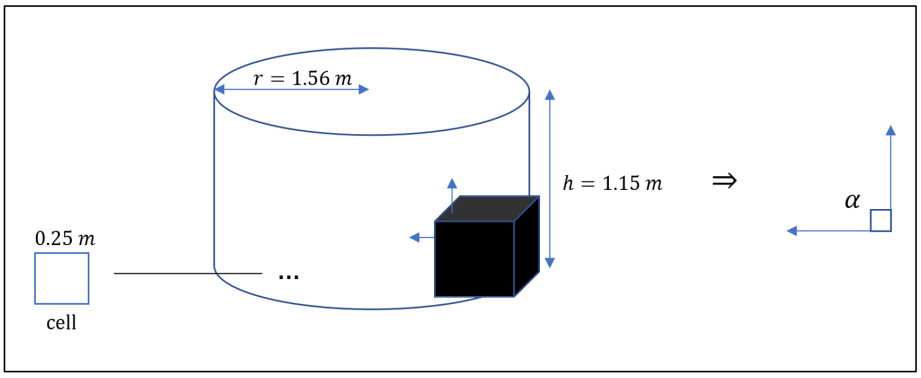


Figure 9: Example illustration of the intuition behind the surface normal approach. The black cube represents an obstacle, and the blue arrows pointing out of the box represent the pair of estimated surface normals with the largest 3D angle between them. In this example, the ground is flat within the cylinder. The resulting angle is used as an inclination estimate (denoted by α). h denotes the cylinder height, r denotes the cylinder radius. It can be seen that the obstacle is situated on the border of the base of the cylinder. Hence, some points on the surface of the obstacle are not considered for the assessment within this cylinder.

The approach in this section depends on built-in functions from the Point Cloud Library (PCL) (described in Section 3.6. The surface normal approach, compares maximum 3D angles between estimated surface normals for all points inside a cylinder. More details on the surface normal estimation can be found in [72].

The surface normal estimation approach, given by PCL, is based on computing the covariance matrix for all points in a local neighborhood given by a radius from a point of interest (i.e., an approximate sphere surrounding the point). Since the covariance matrix is symmetric, its eigenvectors form a basis in 3D (i.e., they are orthogonal). Hence, an estimation for a given point, based on the local neighborhood can be achieved by computing the eigenvalues and corresponding eigenvectors for the covariance matrix and picking the eigenvector corresponding to the lowest eigenvalue as an estimate of the surface normal for the point of interest (i.e., the direction of least spread in directionality between the points in the neighborhood). In other words, the method uses Principal Component Analysis (PCA)

to solve a plane-fitting problem based on the neighborhood of points. The number of points within a cylinder vary significantly between cylinders for the given point cloud. In order to produce results that appear sensible, the surface normal approach uses a search radius $r_{search} = 1.0$ m and a threshold corresponding to the value $\alpha_{lim} = 30$ (i.e., the α-value used to determine the separation between admissible inclination estimates and obstacles).

The surface normal estimation assumes that the points in the local neighborhood form a 2D plane, otherwise the estimation may be less accurate. Maximum 3D angles are compared for each pair of the estimated surface normals, and the maximum angle of these angles is chosen as an inclination estimate (a method from PCL enables computation of the maximum 3D angle between a pair of surface normals). Similarly to the triangle approach, the lowest estimated inclination value is kept for an individual cylinder. Intuitively, this approach may produce estimations of terrain inclinations that are less accurate compared to the triangle approach. Consider a scenario where the cube-shaped obstacle in Figure 9 is situated in the middle of the cylinder as opposed to on the edge of the border of the cylinder base, and the estimation approach produces an estimate corresponding to π radians. In practice, an angle that is more relevant for a vehicle navigating through this cylinder is $\pi/2$ radians, considering that the ground surrounding the cube-shaped obstacle is flat in this example. A potential over-generalization based on the triangle approach may in this case produce an estimation that is more relevant for navigation through the region with flat ground and a small cube-shaped obstacle, provided that the elevation of the obstacle does not induce collision (i.e., an appropriate threshold value is used).

Intuitively, the surface normal approach may be highly sensitive to local variations, and this can potentially lead to an overly-conservative estimation of inclinations (please refer to Section 3.8). However, due to the search radius that is used, local variations can also potentially be missed by this approach. Additionally, if a cylinder region corresponds to a perfect incline, leading to ground characterized by higher elevation (i.e., an unrealistic scenario), the maximum angle that is computed is zero, although there is a distinct slope within the cylinder. This is another example of a case where the triangle approach may produce a more accurate assessment of traversability in terms of inclinations, compared to the surface normal approach.

3.6 Implementation

Péter Fankhauser et al., developed the universal ANYbotics grid map library [73]. This library is designed for mobile robotic mapping, and includes a ROS interface, which makes it highly applicable for traversability mapping and motion planning simulation. In particular, there is a module in the grid map library that provides functionality for conversion of point clouds to grid maps [74]. The grid map library also includes methods for iterating over cells within a circle of a specified radius. This functionality was instrumental in developing the proposed traversability mapping scheme. Extensive modification to the grid map library was required, to obtain the functionality that is described in sections 3.3, 3.4 and 3.5. Some of the functionality in the ANYbotics grid map library depends on the Point Cloud Library (PCL), which is a standalone, open project for 2D/3D image and point cloud processing (for more information, please refer to [75]).

3.7 Results

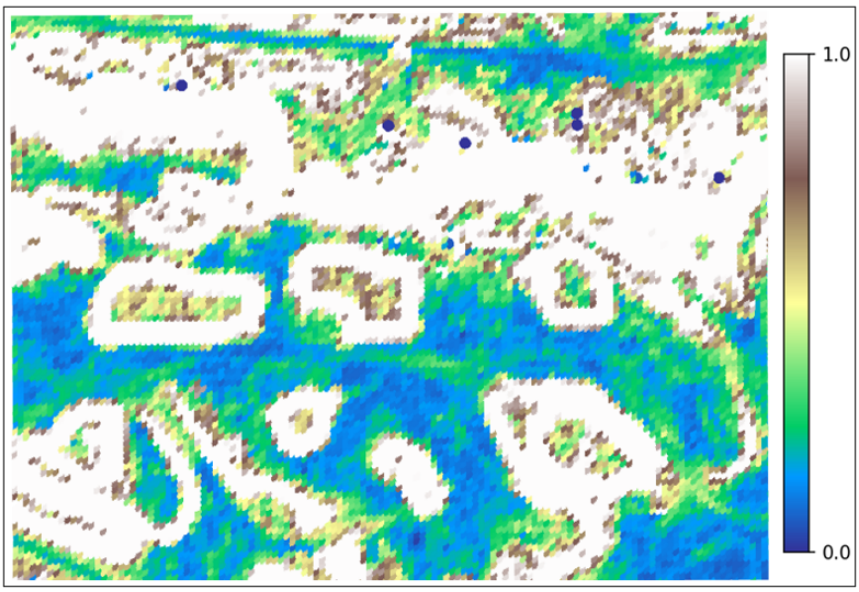


Figure 10: Triangle approach with α_{lim} = 11.8.

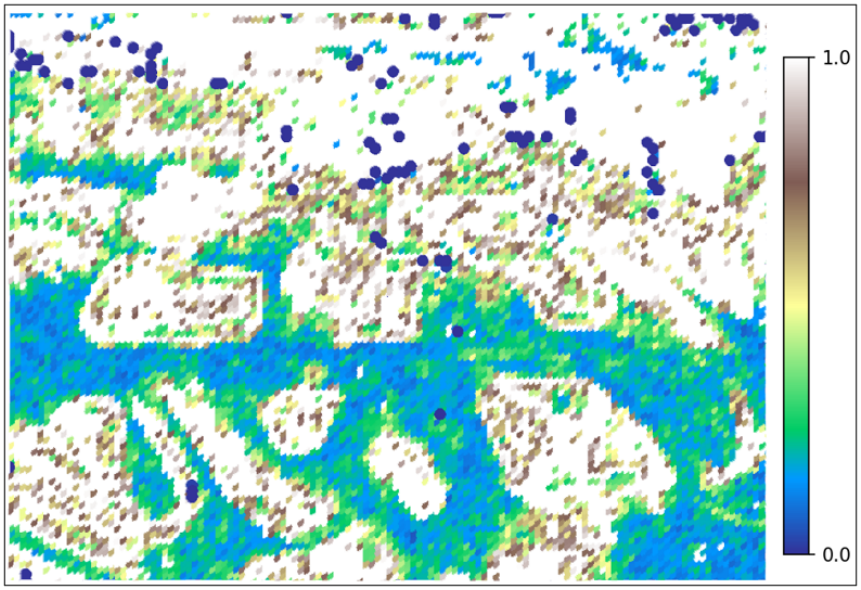


Figure 11: Surface normal approach with r_{search} = 1.0 and α_{lim} = 30.0.

Figure 10 shows the inclination map that is used for motion planning. The map is normalized between 0.0 and 1.0, and a standard color palette is shown. The order of colors between 0.0 and 1.0 correspond to increasing values. This also applies for Figure 11. It can be seen that regions corresponding to admissible inclinations for Figure 10, in the upper part of the map, correspond to large white regions in Figure 11.

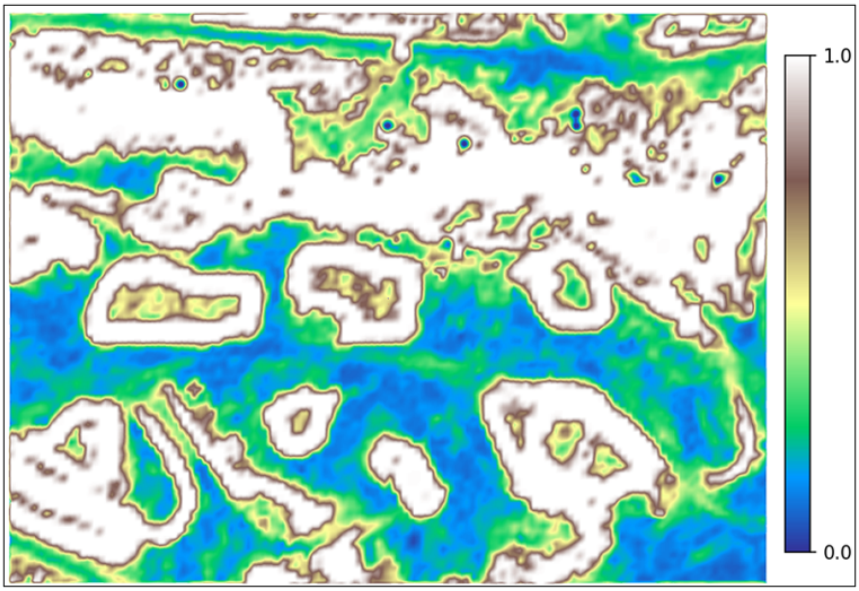


Figure 12: Mean inclination map (triangle approach with α_{lim} = 11.8).

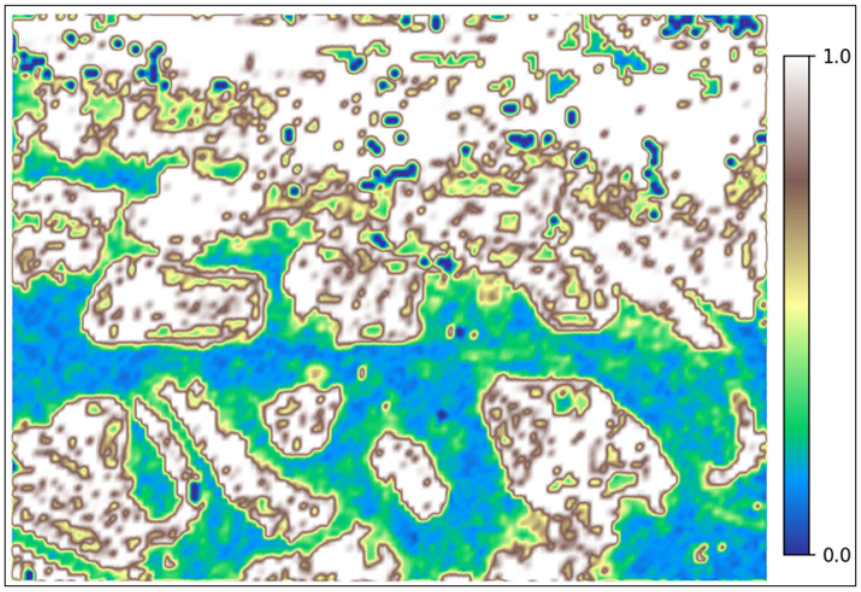


Figure 13: Mean inclination map (surface normal approach with search radius r_{search} = 1.0 and α_{lim} = 30.0).

Figures 12 and 13 corresponds to mean inclination maps for the triangle-based and surface normal approaches, respectively. These maps are generated by applying a simple 3x3 filter over all cells in the grid maps, and for each cell, computing the average of normalized inclination estimates over the values in the filter, and saving the corresponding average in the center cell (i.e., the cell currently being iterated over). This results in more prominent separations between obstacles and admissible inclination estimates.

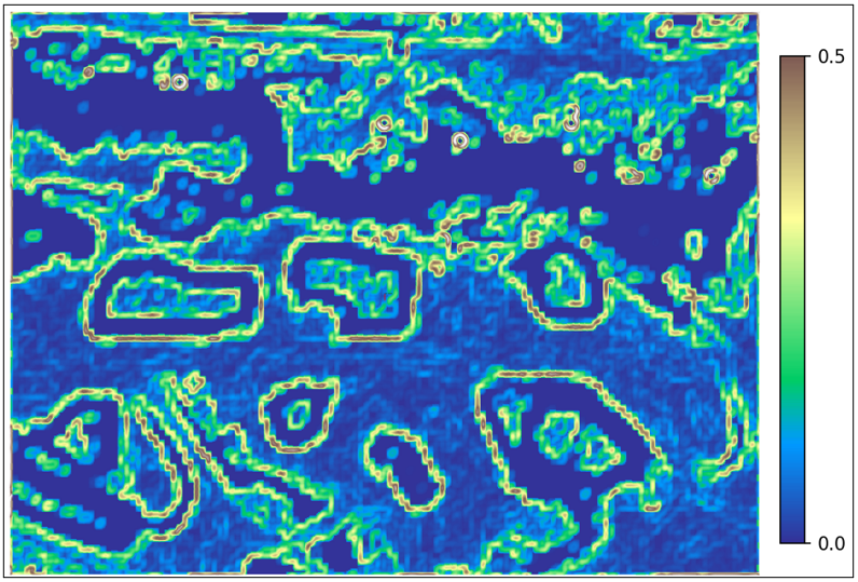


Figure 14: Standard deviations (triangle approach with $\alpha_{lim} = 11.8$).

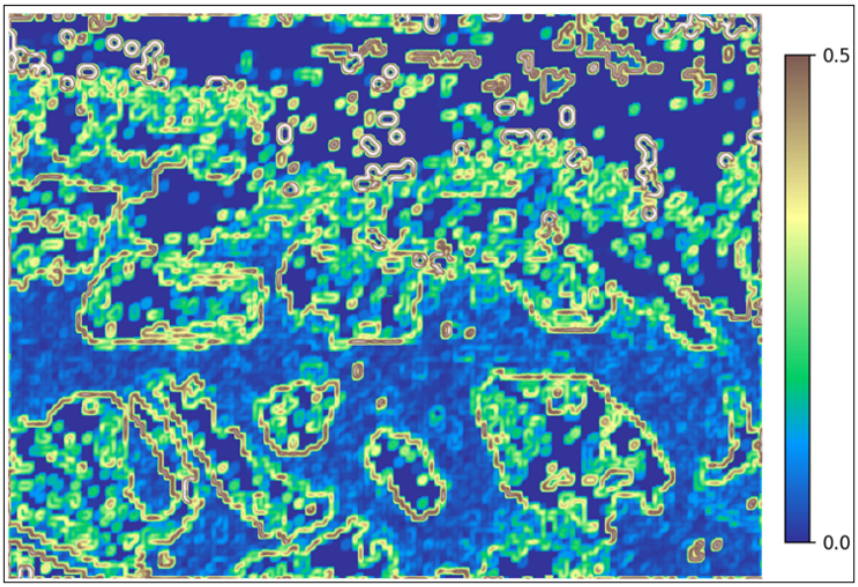


Figure 15: Standard deviations (surface normal approach with search radius $r_{search} = 1.0$ and $\alpha_{lim} = 30.0$).

Figures 14 and 15 correspond to standard deviation maps for the triangle-based and surface normal approaches, respectively. The maps are generated by applying a 3x3 filter over all cells in the map of normalized inclination values, and for each cell, computing the standard deviation for normalized inclination estimates over the values in the filter, and saving the corresponding standard deviation in the center cell. It can be seen for both methods that the variation in inclination values is higher close to obstacles. The highest standard deviation represented in the maps is 0.5 (brown regions), while the lowest is 0.0 (dark blue regions).

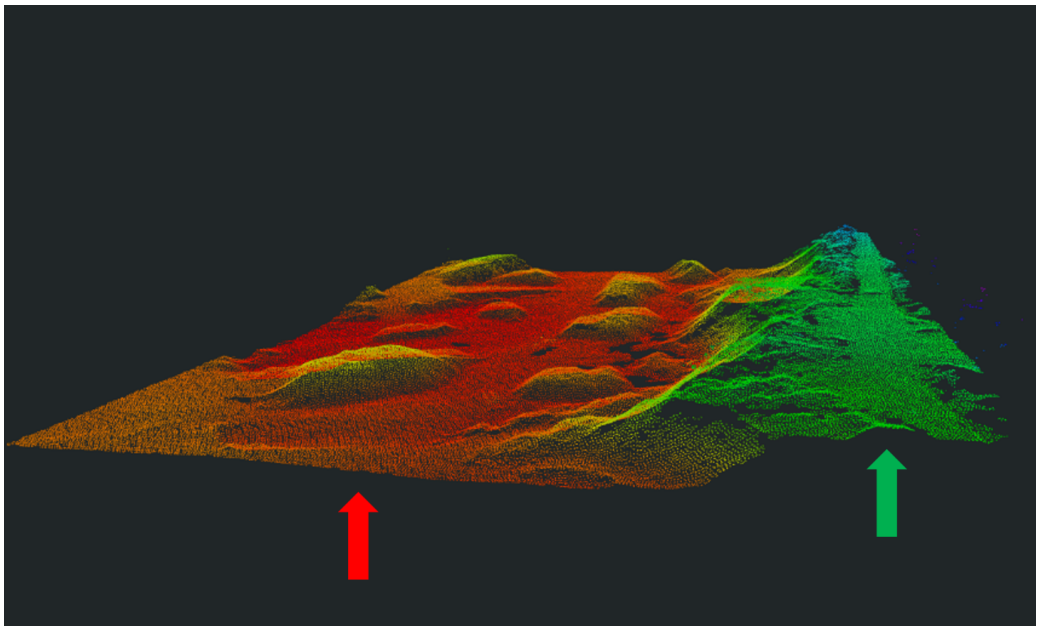


Figure 16: View of point cloud highlighting red and green regions. The red regions correspond to points within the range of the lowest elevations in the map, while green points are situated at a range of higher elevations. The green region that is closer to the green arrow, corresponds to the white region in the upper part of Figure 11.

3.8 Discussion

By inspecting figures 16, 10, and 11, it appears that there is no indication that the green region in Figure 16 consists almost purely of obstacles. It can be seen that the terrain elevation appears to increase more dramatically, closer to the white regions for the green scenario in Figure 10. It can also be seen from Figure 16, that there are red regions that appear to consist of points closer in elevation values compared to the other parts of the map. The same applies for the green region. Hence, this is an indication that the surface normal method is highly sensitive to local variations, so much that entire potential planning regions are evaluated as obstacles. The observation that the surface normal approach is sensitive, can also be seen by looking at the standard deviation map in Figure 15. Compared to Figure 14, there are higher standard deviations for the surface normal approach in small regions in between obstacles, although it is difficult to observe that these variations are present when inspecting the point cloud in Figure 6. This observation is not surprising, as a slight disturbance may cause large angles for the surface normal approach, as described in Section 3.5. It appears that the triangle approach produces more accurate estimates of

inclinations compared to the surface normal approach. By inspecting Figure 6, and 10, it can also be seen that there are some regions (e.g., narrow regions) in the inclination map for the triangle-based approach, that appear to overestimate the inclination, leading to inappropriately narrow passages compared to the passages in the point cloud. This may be an indication that over-generalization in the triangle-based approach can lead to an overestimation of inclination values. However, the narrow passages may also result from a α_{lim} that is set too low for the given point cloud.

The choice of a search-radius in the surface normal approach equal to 1.0 m, may also result in over-generalization with respect to inclination estimates, since a larger search radius implies that more points are used to approximate a single surface normal. This may not accurately reflect the surface normal for a given point, due to the presence of non-linearities in terrain slope. Simply using the largest angle between a pair of surface normals, implies an overestimation of inclinations. Therefore, performing traversability assessment based on a variance measure between angles may be more appropriate in this case. Since the triangle-approach appears to provide a more accurate representation of the inclinations in the given point cloud, this approach is used for motion planning. Despite the limitations of the triangle-based approach, it seen from the figures in Section 3.7 that variations in inclinations are captured to some extent. This is sufficient to enable motion planning simulation. This discussion is purely based on visual interpretations, and a comprehensive analysis based on comparisons with standard approaches for inclination assessment is required to determine whether the approaches can produce accurate models of terrain data (e.g., when different metrics are incorporated into the assessments).

4 Motion Planning

This section aims to address the first research question in Section 1.1.5, by adapting RRT and RRT* to accommodate UGVs without neutral-turn and reverse motion capabilities. In section 4.1, the development of motion primitive schemes for RRT and RRT* is achieved by utilizing the differential-drive model in Section 2.1 to construct a distinct motion primitive set that is used throughout tree construction. The incorporation of kinematics in these sampling-based algoirithms is motivated by the demonstration by Pepy et al. [59], which clearly shows that RRT is suitable for a direct incorporation of a feasibility scheme to expand branches between configurations, while satisfying kinematic constraints. Sections 4.3 and 4.4 aim to address the second and third research questions in Section 1.1.5. Section 4.4, describes potential limitations of using FMM directly in motion planning on terrain for UGVs with limited maneuvering capabilities. This section effectively strengthens the motivation for conducting an investigation into the application of novel feasibility schemes for RRT and RRT*, for UGV motion planning in terrain scenarios. Section 4.3 presents an inclination-based objective cost function. This cost function is used to define a cost field that is used to evaluate performance of all algorithms presented in 4.5. This section is focused on the third research question in Section 1.1.5. Section 4 also gives a description of planning scenarios used for simulation in Section 5.

4.1 Motion Primitive BVP Expansion and Rewiring

In this thesis, the proposed algorithms employ a distinct set of motion primitives (i.e., simple, short and feasible precomputed paths) in an iterative fashion to construct solutions. The use of motion primitives in motion planning algorithms is a popular and simple method for ensuring that the solutions adhere to the kinematic constraints of a vehicle [76]. This primitive set is designed to only allow forward motion, which ensures that the motion planning scheme is suitable for vehicles without neutral-turn or reverse-manuevering capabilities (please refer to Section 4.4). Motion primitives have been successfully applied to RRT and RRT*. Vonásek et al. successfully switch between the primitives in a fixed set of motion primitives to obtain global solutions for RRT [77]. This approach is similar to the one presented in this section, due to the use of a fixed set of primitives in constructing

a path. The work in [78], extends RRT* with a scheme for dynamic and partially known environments that uses a database of precomputed primitives.

The motion primitive set is constructed by integrating the differential-drive model (presented in Section 2.1), to compute successive vehicle states using positive belt-speeds only. By applying the distinct set in an iterative fashion, the problem of developing a collision-checking scheme is simplified (please refer to Section 4.2). A symmetrical motion primitive set, characterized by positive forward speeds for all primitives to ensure feasibility, is used in Traversability Hybrid A* to produce near-optimal paths in terrain based on minimizing on the accumulated cost along the primitives [8].

Figure 17 shows an example RRT expansion using the distinct primitive set. All primitives are included to illustrate how the set is iteratively expanded to construct a tree. A random node in the search space is initially sampled from a uniform distribution. From the start configuration, the initial primitive set is expanded, and the closest node (i.e., corresponding to an end point in the set) to the randomly sampled node is chosen for tree expansion. This process is repeated to ensure the tree contains a given number of nodes. To construct a solution, the nodes within a 5 m radius from the goal are evaluated based on the accumulated cost from the root node, to select a solution that corresponds to the lowest accumulated cost. The red path in Figure 17 is a potential solution path between the root node of the tree (i.e., the initial configuration) and a potential goal node. In this case, all end points of the motion primitive sets initially correspond to potential nodes for expansion.

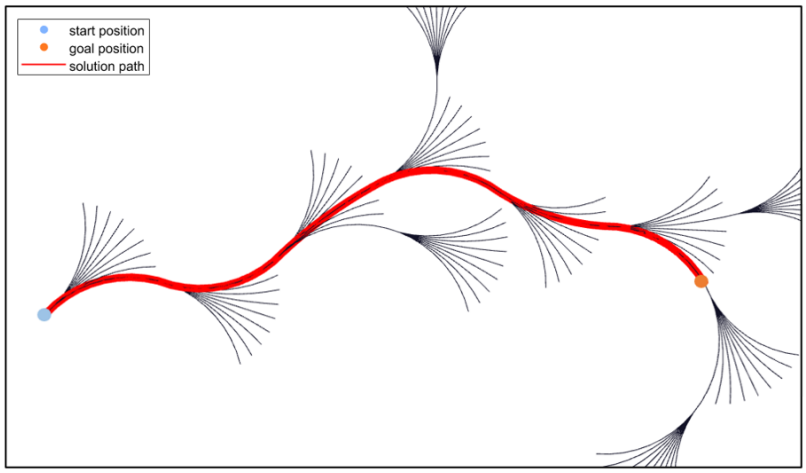


Figure 17: Motion primitive RRT expansion and solution path

Given that RRT does not include optimization steps, the exploration is random, which means that solutions can potentially be improved by increasing the number of nodes used to construct a tree. Using a larger number of nodes leads to an increased exploration of the search-space. The distinct motion primitive set used to perform RRT expansion and tree-building ensures kinematic feasibility, but a limitation of the scheme is that it constrains exploration. In the default RRT algorithm, the straight-line expansions are not restricted to fixed lengths, meaning that if a random configuration is sampled below a given threshold from the nearest node in the tree, the tree is expanded to the random configuration. On the other hand, the expansion used in this thesis is different from the straight-line expansions in RRT as the primitive sets are based on integrating a vehicle model. Only one of the primitives in a set correspond to a fixed length expansion along a straight line. The motivation for extending the motion primitive scheme to RRT* is to investigate whether solution quality is improved compared to the RRT-variant, by applying the same primitive sets in a BVP-optimization procedure.

The optimization scheme used for RRT* iteratively solves two-point boundary value problems (TBVPs), to connect primitives, but the choice to improve a solution based on neighboring nodes follows the same underlying principles as in the default RRT* algorithm. In the first step of the procedure, when a new primitive set is expanded, the node that is closest to the randomly sampled node is first chosen for expansion. This corresponds to the step that is repeated throughout tree-building in the motion primitive RRT-variant. For the RRT*-variant, all neighbors in a radius from the newly expanded node are used to expand new primitive sets. If any of the end points of these primitive sets satisfy the TBVP (i.e., connections to the newly expanded node exist), the accumulated costs from the newly expanded node to the root is evaluated based on comparing the different connections possible (i.e., based on TBVP solutions). The connection corresponding to the lowest accumulated cost is chosen. Hence, the newly expanded node is given a new parent and connection to the tree if that connection results in a lower accumulated cost to the root node.

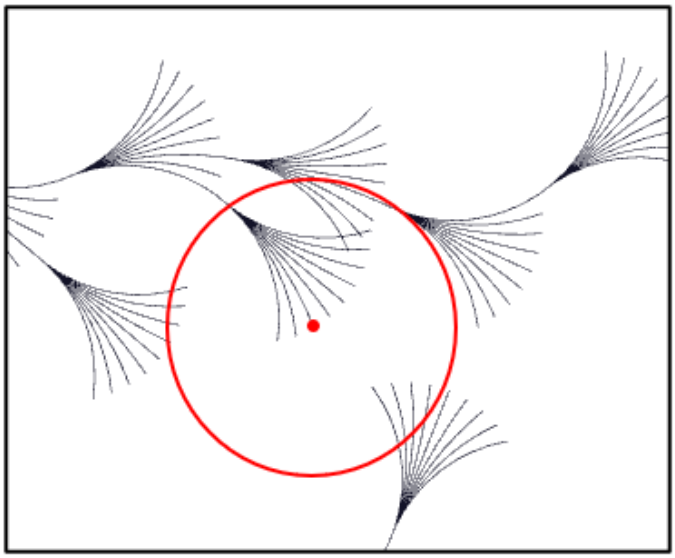


Figure 18: Potential nodes for solution improvement

In the second step of the optimization procedure, the motion primitive set is expanded from the newly added node to find connections to its neighboring nodes. If these connections exist, the accumulated cost from the neighboring nodes to the root are evaluated based on connections from the tree to those nodes, through the newly expanded node. A connection that yields a lower accumulated cost from the root to a neighboring node is chosen. When this happens, the connection between the neighboring node and its parent is severed, and the newly added node is assigned as the neighbor's new parent. Figure 18 is an example illustration of potential neighboring nodes surrounding a newly added node. The potential neighbor nodes correspond to the end points of the primitive sets within the red circle.

A clear limitation of this scheme compared to the optimization procedure in the default RRT* algorithm, is that connections between nodes are likely to be improved to a lesser extent. Similar to the motion primitive RRT-variant, tree-building is constrained. Due to the constrained construction of the tree, there may be a lesser number of nodes available for potential solution improvement in a neighborhood surrounding a node. In addition, the choice to only allow forward positive speeds may result in failure to optimize in a given circle as nodes may not be reachable [79]. Webb et al. state that the optimization procedure in RRT* is suitable when any pair of states in the search-space can be connected in an optimal trajectory [40]. Therefore, the TBVP solution presented in this section is not expected to maintain AO and convergence properties of RRT*. Figure 18 is an unrealistic

example based on a small number of nodes. All expanded primitives are shown. This example is used to illustrate the principle of the scheme, but in practice a large number of nodes must be enforced to ensure that connections are replaced in the optimization procedure.

Results indicate that the motion primitive expansion results in a greedy exploration that does not lead to higher-quality solutions for the proposed schemes. The schemes are used to construct solutions based on different costs, and the proposed algorithms are evaluated based on the same cost field (please refer to Section 5). Solutions can potentially be improved by increasing the number of nodes in the trees, or by using a richer set of primitives that enables connections between more states in the search-space. However, solving a larger number of two-point boundary value problems (TBVP) can be computationally expensive [79], and the state reachability-issue would still be present. Palmieri et al. propose an extend function for RRT and RRT* that is shown to outperform certain motion primitive sets characterized by primitives of varied lengths [79]. The approach by Sakcak et al. may serve as a better alternative for use of motion primitives in RRT*, as this approach considers a discrete set of state pairs in a grid, and the optimal cost between states in pre-computing primitives that are stored in a database for planning [78].

4.2 Collision-Detection Scheme

In motion planning on terrain, safety should be a priority [8]. Zhang et al. state that a common approach to ensure obstacle avoidance is to expand obstacles to ensure a hindrance-distance is maintained [80]. On the other hand, the authors argue that safety is not guaranteed by purely ensuring obstacles are expanded by a given distance, because safety also depends on the speed of a robot (i.e., higher speeds dictate a larger distance requirement). In addition, they note that in complex environments, obstacle expansion might block passable regions, resulting in failure to obtain solutions.

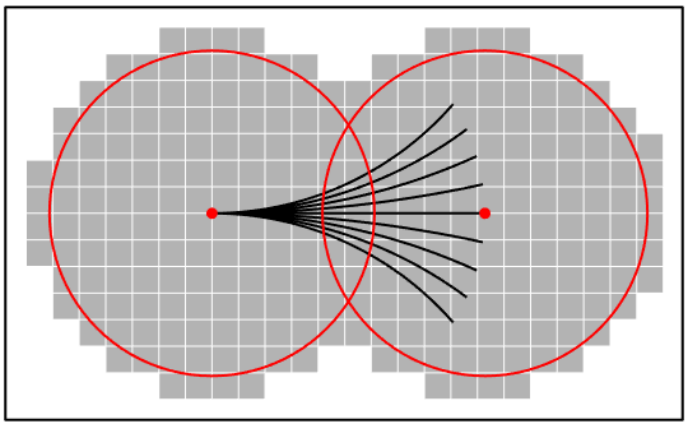


Figure 19: Proposed collision-detection scheme

The collision-checking scheme, illustrated in Figure 19, checks for obstacles in all cells within and intersected by the border of a minimum-enclosing circle surrounding each node in the tree. A fixed radius is used to enforce a clearance from obstacles, but the approach implicitly accounts for robot speeds and motion capabilities, because feasible paths are effectively encoded into the tree, due to the distinct motion primitive set that is used throughout construction. The combination of clearance checks with ensured feasibility can potentially help strike a balance between effective navigation and safety in a given environment.

Figure 19 shows how collision-checking is done for two nodes in the tree when the expanded node corresponds to the end point of the straight primitive in the set. Based on the current primitive set, a minimum-enclosing radius is used, but the following principle can be used

to choose an appropriate collision checking radius:

$$r_{col} = k\frac{L_{max}}{2.0} \tag{9}$$

where L_{max} corresponds to the length of the longest primitive in a distinct motion primitive set. In the scheme used in this thesis, the set is characterized by equal lengths for all primitives, meaning that L_{max} corresponds to the length of any primitive in the set. For example, the choice $k \geq 1.0$ ensures that the circle borders will at least intersect, for two connected nodes. Choosing a higher value of k results in a more conservative approach, which increases overall safety. However, if k is increased too much, this may result in a high computational load and increase in unnecessary processing of cells due to overlapping regions between circles. In some specific scenarios, depending on the shape and placement of obstacles, the choice $k = 1.0$ may not be sufficient to detect all obstacles. Hence, depending on the scenario and safety requirements, choosing a larger value of k may be necessary. In the event that the required value of k results in an r_{col} greater than the minimum-enclosing radius of the vehicle, solutions through narrow passages may not be considered. The scheme may be less applicable for distinct primitive sets featuring longer primitives, where a larger value of r_{col} is required.

Figure 19 shows that, for the expansion corresponding to the straight primitive in the set, the minimum-enclosing radius ensures that the circles always overlap to some extent (i.e., $k \approx 1.46$). However, as seen in the figure, some regions are not covered by the clearance check. A comprehensive analysis, preferably based on real-world experiements, is required to determine whether a minimum-enclosing radius is sufficient for the given scheme.

Bialkowski, Karaman, Otte and Frazzoli state that collision checking is the main computational bottleneck in sampling-based algorithms [81]. In their work, they propose a highly efficient collision-detection scheme that stores a lower bound on distance to the nearest obstacle for each collision checked point. The similarity to the scheme presented in this section is that it is based on distances between samples in the tree and obstacles. The approach by Bialkowski et al. allows samples to be immediately determined collision-free without calls to the collision-checking module based on comparing the distances between unchecked samples to samples that have been checked, to lower bound distances between

checked samples and obstacles. Incorporating this approach may improve the efficiency of the current motion planning scheme. Since simulation results indicate that the collision-detection scheme is the significant contributor to higher path traversability, obtaining higher quality solutions with the given motion planning scheme may be difficult as the clearance checks may also limit exploration of the search-space.

4.3 Inclination-Based Objective Cost and Optimal Traversability

In this thesis, the inclination-based cost function that is used by the sampling-based RRT and RRT* variants (i.e., algorithms prefixed by ObjCost in Section 4.5) is given by the following equation:

$$\frac{1}{F_{i,j}} = \lambda e^{-\frac{(\alpha^{\wedge}_{i,j} - \alpha^{\wedge}_{\max})^2}{\eta}} + (1-\lambda) e^{-\frac{(\alpha^{\wedge}_{i,j})^2}{\eta}} \tag{10}$$

The left-hand side of the equation is denoted by the inverse of the speed function used in the Eikonal update in FMM. This section will go into further detail about the reasoning behind this choice. In the proposed cost function, $\alpha^{\wedge}_{max}$ represents the maximum admissible inclination value in a normalized inclination map (i.e., $\alpha^{\wedge}_{max} = 1.0$). $\alpha^{\wedge}_{i,j}$ represents the inclination value of the current neighbor cell i, j, for which the Eikonal equation is solved. The first term ensures a shift of the Gaussian peak to $\alpha^{\wedge}_{max}$. This term is responsible for penalization of terrain with higher admissible inclination values. The purpose of the second term is to penalize terrain with lower admissible inclination values. Penalization can be controlled by varying λ alone. Therefore, η is fixed in simulation ($\eta = 0.18$).

Figure 20 shows the effect of decreasing λ from 1.0. When $\lambda = 1.0$, only the first term is prioritized, and the red curve (i.e., the contribution from the first term) overlaps with the blue curve which represents the resulting cost. In this case the green curve corresponding to the contribution from the second term is zero. Figure 20 shows that decreasing λ leads to a higher degree of prioritization of the second term, which counteracts the effect of the first term on the resulting cost. Hence, an interval of admissible inclinations can be assigned a lower cost, given the current formulation.

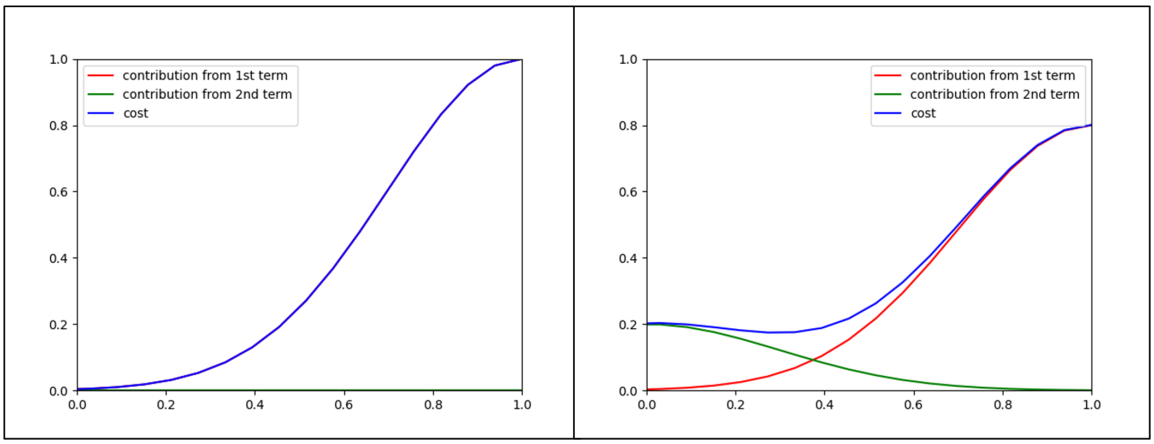


Figure 20: The effect of decreasing λ from 1.0 to 0.8 on the resulting cost.

It is important to acknowledge that assigning a lower cost to higher admissible inclinations may require additional a priori information relating to the underlying characteristics of the terrain to produce costs that more accurately reflect the traversability for a given environment. However, there may be scenarios where lower inclination values correspond to less traversable regions. For example, lower inclinations are associated with water flowing at slower rates [82]. Intuitively, enclosed regions characterized by low inclinations, surrounded by elevated terrain, may harbor still water-bodies resulting from an accumulation of water over time. Furthermore, when the available point cloud data is sparse, the presence of dense vegetation, potentially at lower inclinations may be impossible to uncover based on analyzing the given data. In this case, when traversability is solely based on inclination data and other terrain characteristics are unknown, prioritizing slightly higher admissible inclination values than the lowest admissible value may increase the likelihood of avoiding obstacles that cannot be discerned by excluding inclination values over a threshold.

In this thesis, given the characteristics of the traversability map that is used for simulation, results indicate prioritizing higher admissible inclinations leads to prioritization of regions closer to obstacles (please refer to Section 5), which implies that the cost assigned based on admissible inclinations when the second term is prioritized, may not accurately reflect the true traversability of the terrain with respect to inclination estimates. Therefore, the cost formulation may be improved by accounting for additional features of the terrain to determine a suitable interval of inclination values based on a given scenario.

For the sake of comparison and evaluation, the optimal path in terms of traversability between a start and goal point is in this thesis defined as the one that minimizes the accumulated cost in the cost field given by the cost formulation presented in this section. Therefore, the optimal path is implicitly determined by the value of λ, and given by gradient descent through the FM field (i.e., gradient descent is expected to optimize according to the objective, given by the cost formulation in this section). The accumulated cost along a path is given by [8]:

$$A_c = \int_{\mathcal{S}} c \, ds \tag{11}$$

where c is given by the cost formulation and $\mathcal{S}$ is the path along which the cost c is integrated to find A_c. In Section 5, this is used as a metric for evaluation. Another metric is average cost, which is defined by the following expression [8]:

$$\mu_{A_c} = \frac{A_c}{L} \tag{12}$$

According to Jaillet et al., average costs can be misleading as they can potentially reward paths with unnecessary detours, as the accumulated cost is distributed evenly across a path [83]. The authors explain that through a low-cost region, a path with detours obtains a lower average cost than a straight path through the same region. Yet, the straight path may be more optimal in terms of traversability. For this reason, average costs should be interpreted with caution. Jaillet et al. argue that A_c is potentially more reliable as a criterion for evaluating performance in this context. Intuitively, the path integral will penalize detours, because it simply accumulates costs over a path.

The optimal path is the one that minimizes the accumulated cost, and gradient descent through the FM cost field is expected to always produce the optimal path with respect to the other algorithms presented in this section (i.e., when accumulated costs are evaluated over the field defined by the cost formulation that is given in this section). For this reason, results from FMM are used as ideal benchmarks for accumulated costs in Section 5. These benchmarks are "ideal" as, unlike the motion planning algorithms presented in this section, FMM does not account for vehicle orientation. Furthermore, the NH-FMM algorithm that

is presented in this section gives rise to results (please refer to Section 5) which show that paths generated by FMM do not guarantee feasibility for vehicles that are limited by the maneuverability constraints considered in this thesis.

4.4 Non-Holonomic Fast Marching Method

Due to the presence of obstacles and potentially high curvatures of an FMM path, this path does not guarantee feasibility for vehicles without neutral-turn and reverse-motion capabilities (as shown in the results discussed in Section 5). Figure 21 serves as an illustration of an example scenario, where the FMM path follows the obstacles closely. The purple arrow going from the start position represents the initial vehicle orientation. In this scenario, if the FMM path is used as input to a path following scheme, a vehicle without neutral-turn or reverse-motion capabilities (or both) may be unable to reach the goal. The example scenario in Figure 21 is used to highlight that potential limitations of FMM paths may also concern other types of vehicles that are not limited by the motion constraints considered in this thesis.

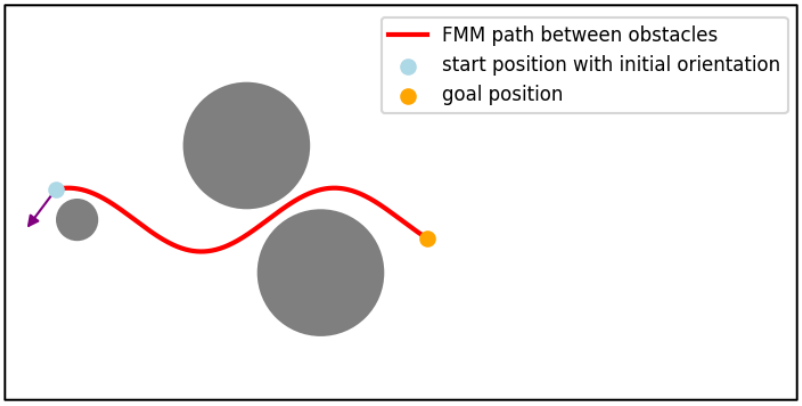


Figure 21: Illustration of FMM path scenario with simple, circular obstacles.

In Figure 21, due to the initial orientation, a vehicle without neutral-turn and reverse-motion capabilities would have to initially drive in a direction corresponding to the direction of the arrow to align with the red path. Due to the presence of the smaller obstacle, initial alignment to the path may be difficult, because of the vehicle's required clearance from obstacles. The part of the red path going through the narrow passage between the two larger obstacles may also fail to satisfy the vehicle's clearance requirements. This means that, if

a vehicle has neutral-turn and reverse-motion capabilities, it may still be unable to follow an FMM path. Furthermore, since FMM does not consider vehicle orientation, parts of an FMM path may have curvatures exceeding the curvature of the maximum curve a vehicle can navigate due to its constraints.

For a vehicle with neutral-turn or reverse-motion capabilities, or both, the problem of initial alignment to the FMM path can potentially be addressed, given that the FMM path is known. Assuming that a vehicle has neutral-turn capability, but does not have reverse-motion capabilities, the FMM path can be used to estimate an orientation that may enable initial alignment to the path in the presence of obstacles close to the starting position.

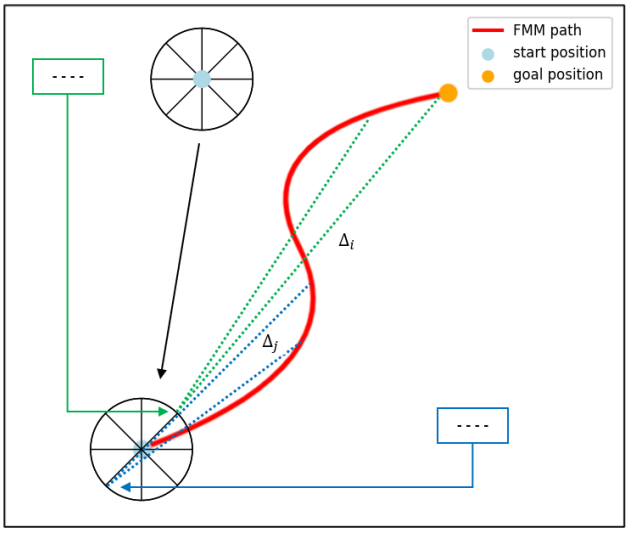


Figure 22: Illustration of a scheme that can be used to estimate an orientation for initial alignment to FMM path.

Figure 22 is a simple illustration of a scheme that may be used to estimate the orientation needed for a vehicle with neutral-turn capability to initially align to the FMM path. The black circle surrounding the start position has strictly positive and fixed line segments going from the vehicle's start position to the circle's boundary (i.e., the length is given by the radius of the black circle). The fixed segments are chosen to intersect the boundary of the black circle at fixed intervals. The set of segments can be used to extract potential orientations that may improve vehicle alignment with respect to the FMM path. By calculating the sum of the distances between the end point of a single segment and equidistant points along the path, the following expression is obtained for the green dashed lines in Figure 22 (note that

only a few of the dashed lines are illustrated):

$$D_{green} = \sum_{i=1}^{n} \Delta_i \tag{13}$$

where D_{green} is the accumulated distance corresponding to the sum of the lengths of the green dashed lines, and n is the number of green dashed lines, corresponding to the number of equidistant points along the path. For the blue dashed lines (n is unchanged):

$$D_{blue} = \sum_{j=1}^{n} \Delta_j \tag{14}$$

Due to the location of the end point of the line segment corresponding to the blue dashed lines compared to the location of the end point of the segment corresponding to the green dashed lines (as seen in Figure 22), the following inequality holds:

$$D_{blue} > D_{green} \tag{15}$$

Hence, the orientation that can be extracted from the line segment corresponding to the green dashed lines is a better candidate compared to the orientation that can be extracted from the line segment corresponding to the blue dashed lines. This is to show how accumulated distances corresponding to end points of a set of line segments distributed inside a circle, surrounding the start position, can be compared to estimate an orientation that is more aligned to the FMM path. For example, based on the distinct motion primitive set described in Section 4.1, the length of an arbitrary primitive in the set can be used as a radius for the black circle. The estimation may be improved by increasing the number of line segments inside the black circle, and by performing a fine-grained re-sampling of the FMM path to obtain a larger number of equidistant points.

By allowing for a single in-place rotation at the start position to the estimated orientation, a vehicle with neutral-turn capability may be able to initially follow the path in the scenario that is illustrated in Figure 21. In this scenario, for a vehicle with reverse-motion capabilities, initial alignment to the FMM path may be possible, for example, by allowing for reverse maneuvers in a region surrounding the start position. This assumes that due to

the vehicle's initial orientation and the presence of the obstacle close to the start position, initial alignment is impossible if the vehicle does not have neutral-turn or reverse-motion capabilities.

Given the estimated initial orientation, the distinct motion primitive set can be used to connect along the FMM path, assuming that the narrow passage in Figure 21 satisifies the vehicle's required obstacle clearance, and that the maximum curvature of the FMM path allows for such connections. Since the FMM path is dependent on the underlying cost field, it may follow narrow corridors and exhibit high curvatures. Therefore, such connections may result in failure to produce solutions for a given planning problem. This highlights some of the limitations of using FMM directly for planning in complex environments, when solutions must adhere to the constraints of a vehicle.

The NH-FMM algorithm used to generate the results presented in Section 5, does not allow for neutral-turns or reverse maneuvers, and employs the distinct primitive set to connect along the FMM path. If this algorithms fails to produce a solution, either the curvature of the FMM path is too large, or the clearance requirement is not satisfied. NH-FMM uses the same collision-checking scheme presented in Section 4.2, by checking for clearance at motion primitive endpoints, for the sake of maintaining consistency between the algorithms that are used in evaluation (please refer to Section 5).

Results show that the NH-FMM algorithm significantly outperforms other algorithms employing the motion primitive scheme with respect to accumulated cost when a solution is found. Therefore, when motion primitives are successfully connected along the FMM path, this solution is preferred. Hence, a potential improvement to the current scheme may be to combine NH-FMM with a sampling-based approach. For example, by resorting to a sampling-based exploration when further connections to the FMM path are no longer possible, it may be possible to reestablish connection to the optimal cost path. Plaku, Kavraki and Vardi propose a multi-layered synergistic approach to planning under dynamic constraints that combines properties of deterministic and sampling-based methods to obtain solutions [84]. In the approach, a tree search establishes feasible connections between a sequence of decomposition regions, and the progress from the tree search is fed to a high-level deterministic planner.

4.5 Overview of Algorithms and Associated Costs

The motivation for including a larger set of algorithms based on the motion primitive scheme presented in Section 4.1, is that when a new scheme is attempted, a deeper insight into its effectiveness can potentially be gained by applying the scheme to different costs and evaluating the algorithms based on the same cost field. An algorithm that successfully optimizes on a given cost field will reflect higher performances compared to an algorithm that produces sub-optimal paths in the same field. Therefore, by evaluating all algorithms based on the same cost field, and associating variations of the scheme with different costs, this approach can potentially make it easier to determine if an algorithm successfully optimizes on traversability according to the definition given in Section 4.3. However, simply comparing performance based on a variation of a scheme employing different costs may not be sufficient. Although a comparison based on cost variations in a new scheme may result in observable trends in performance, the existence of such trends is unknown in advance. This is the main reason for incorporating standard algorithms to produce idealistic benchmarks. Section 4.6 provides a description of the implementation of the algorithms outlined in this section.

The following is an overview of all algorithms used to generate results.

- FMM - *Fast Marching Method*. Gradient descent through the FM cost field. FMM is expected to produce the optimal path according to the definition of optimal traversability, given in Section 4.3. Hence, the accumulated cost of an FMM path is expected to be lower compared to other algorithms for any given planning scenario. This algorithm produces idealistic benchmarks for accumulated costs.
- Default RRT* - *Rapidly-exploring Random Tree**. Distance-optimization and collision-checking along branches only. This algorithm is expected to generally produce the optimal-distance path (i.e., RRT* is AO, please refer to Section 2.5.2). The algorithm is expected to produce idealistic benchmarks for path lengths.
- NH-FMM - *Non-holonomic Fast Marching Method*. Paths are generated by attempting to connect motion primitives to the resulting FMM path with a bias toward the goal. When this algorithm fails, either there is not sufficient clearance along the FMM path, or the curvature of the FMM path is too aggressive. The collision-checking scheme presented

in Section 4.2 is used to evaluate clearance (please refer to Section 4.4).

- MP-RRT - *Motion Primitive Rapidly-exploring Random Tree*. Motion primitive scheme with clearance checks (given by the collision-checking scheme presented in Section 4.2) for every node, applied to the basic RRT algorithm. This algorithm saves the accumulated distance from the root to every node in a tree, and chooses a solution node with lowest accumulated distance from the root node.
- ObjCost-RRT - *Objective Cost Rapidly-exploring Random Tree*. Motion primitive scheme with clearance checks for every node applied to RRT. This algorithm saves the accumulated cost (please refer to the formulation of the objective inclination-based cost in Section 4.3) from the root node to every node in the tree, and picks the path with the lowest accumulated cost from the root.
- Inc-RRT - *Inclination-Based Rapidly-exploring Random Tree*. Similar to ObjCost-RRT, but picks the path with the lowest accumulated inclination cost (uses the inclination map directly).
- Dist-RRT* - *Distance-based Rapidly-exploring Random Tree-star*. Motion primitive RRT* with clearance checks for every node and BVP-rewiring, attempts to optimize on distance.
- ObjCost-RRT* - *Objective Cost Rapidly-exploring Random Tree-star*. Similar to Dist-RRT*, but attempts to optimize on accumulated cost (please refer to the formulation of the objective inclination-based cost in Section 4.3) along the motion primitives.
- Inc-RRT* - *Inclination-based Rapidly-exploring Random Tree-star*. Similar to ObjCost-RRT*, but attempts to optimize on accumulated inclination cost along the motion primitives (uses the inclination map directly).

4.6 Implementation

The implementation of FMM, presented in Section 2.4 is based on code from an open-source framework made available by Javier V. Gomez [85]. Although the code used to generate motion planning simulation results in Section 5 is a substantially modified version of the C++ FMM implementation by Gomez, the open-source code was instrumental to understanding the fine-grained details of the motion planning method. Gomez is known for his collaboration with Alberto Valero-Gomez, Santiago Garrido and Luis Moreno on the saturated FM square (FM^2) and the heuristic FM^2 star (FM^{2^*}) methods. The presentation of the extended methods is found in [26]. The FMM implementation was adapted to ROS 1 Noetic, and to the Anybotics grid map library that was extended to enable generation of inclination maps in Section 3 (please refer to Section 3.6 for details on the grid map library and the implementation related to traversability mapping). For RRT and RRT* the free-use Python code [86] made available by Md Mahdbubur Rahman was translated to C++ and extensively modified for use with ROS 1 and the Anybotics library to incorporate the motion primitive schemes presented in Section 4.1, and the collision-scheme presented in Section 4.2. In [87], Rahman et al. propose a sampling-based extension to RRT* to support multiple objectives, non-additive costs and cooperative conditions.

4.7 Chosen Planning Scenarios

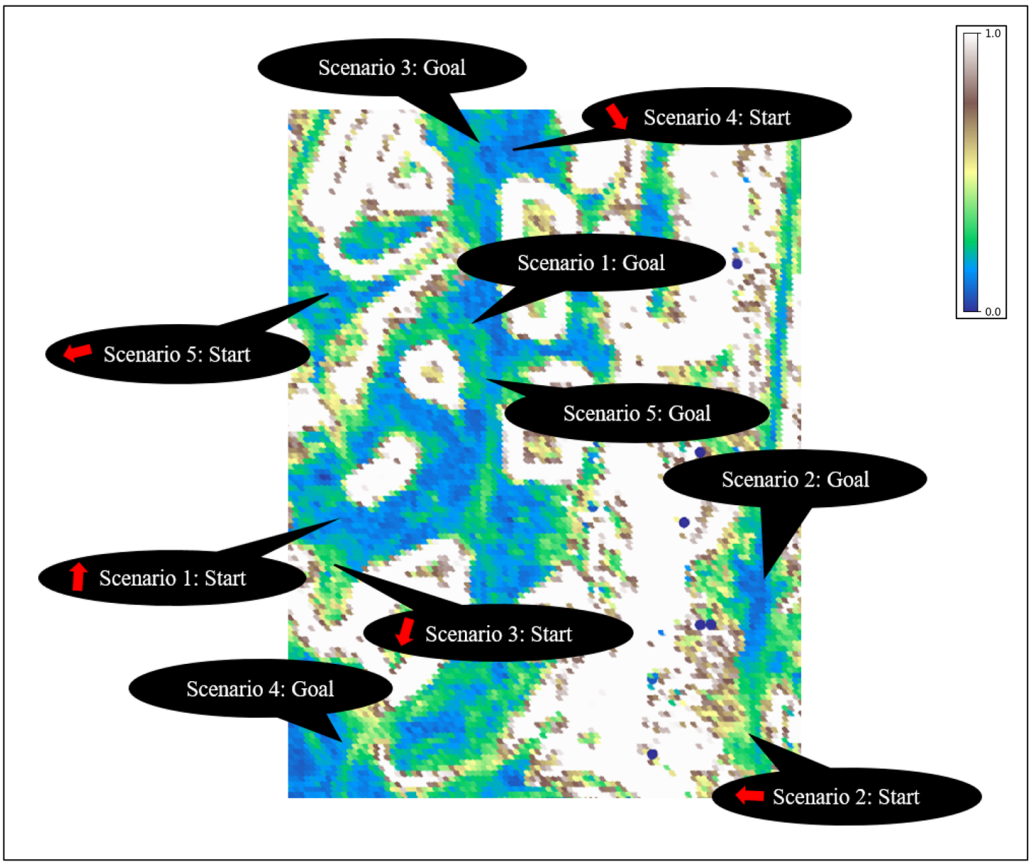


Figure 23: Rough high-level overview of chosen planning scenarios. The black arrows point to start and goal positions for the 5 chosen scenarios, and red arrows represent sampled orientations at the start positions.

To enable a comparison between algorithms (outlined in Section 4.5), that is less biased by the specific variations of the terrain in a planning region, governed by a distinct pair of initial and goal configurations, a set of initial and goal configurations is chosen based on the terrain data to capture different planning scenarios. From 5 chosen scenarios, the first is used as an example in this section. A high-level overview of approximate start and goal positions, including sampled orientations is given in Figure 23.

The first planning scenario consists of an open planning environment with alternative paths to the goal. The alternative paths for this planning scenario consist of different types of variations in inclination values, and for a given value of λ, one path is preferred by FMM over others. As the default RRT* algorithm uses distance-optimization, this path is not expected to vary significantly from one λ-value to another (i.e., Default RRT* is generally expected to produce the shortest path, and is not influenced by λ). For example, if the

accumulated cost for Default RRT* is only slightly higher compared to FMM for a pair of given paths in the same planning scenario, this means the Default RRT* path is traversable according to the definition given in Section 4.3. However, since FMM and default RRT* do not enforce a clearance like the other algorithms, such a result may indicate that the path chosen by FMM is one that prioritizes high inclination values, or one that is close to obstacles. Hence, an advantage of using more than one benchmark algorithm is that such scenarios can be uncovered more easily. This can increase transparency of the data, which is especially useful when the cost function used to evaluate the algorithms has not been tested before (e.g., the cost function may not necessarily reflect terrain traversability accurately for all λ-values - please refer to Section 5). On the other hand, the main reason for using more than one algorithm to benchmark against is that they are only ideal. For example, since FMM is expected to always produce the optimal path in terms of traversability, other algorithms cannot be expected to produce accumulated costs that can compete with the results by FMM. Figure 25 shows paths generated by all algorithms for the first scenario (with $\lambda = 1.0$). This picture is included to provide rationale for incorporating a set of chosen planning scenarios. The figure also shows the color scheme that is used to represent the different algorithms presented in Section 4.5.

When $\lambda = 1.0$, the yellow path in the picture is chosen by FMM, whereas a shorter, blue path is chosen by the default RRT* algorithm. In this specific scenario, when λ is decreased, FMM will eventually find a path that is close to the default RRT* path. The terrain followed by the blue path is characterized by higher inclinations, while the terrain followed by the yellow path is characterized by lower inclinations. As seen in the figure, the paths chosen by the other algorithms are varied. A relatively open region, characterized by alternative paths with varied terrain may provide insight into the performance of algorithms when the generated paths are prone to vary from one λ-value to another, due to the characteristics of the terrain in the planning region. However, variations in paths may also occur due to the intrinsic properties of the algorithms. For this reason, other planning scenarios are included. Scenario 2 (Figure 23) features a partially constrained environment with an obvious path to the goal. In Scenario 3, there are longer, alternative paths to the goal in an unconstrained environment. Scenario 4 (Figure 23) consists of an open starting environment and a partially constrained environment surrounding the goal, with a distinct, prominent

path to the goal. For Scenario 5, the starting environment is constrained, and there are two constrained alternative paths leading to the goal. The variation in length between the start and goal positions is done to enable evaluation of performance of the algorithms when they are forced to choose longer paths. To reflect that the initial orientation of an autonomous vehicle may be unknown, for each planning scenario, an initial orientation is sampled from a uniform distribution. The red arrows in Figure 23 represent the sampled orientations used to generate the simulation results in Appendix B.

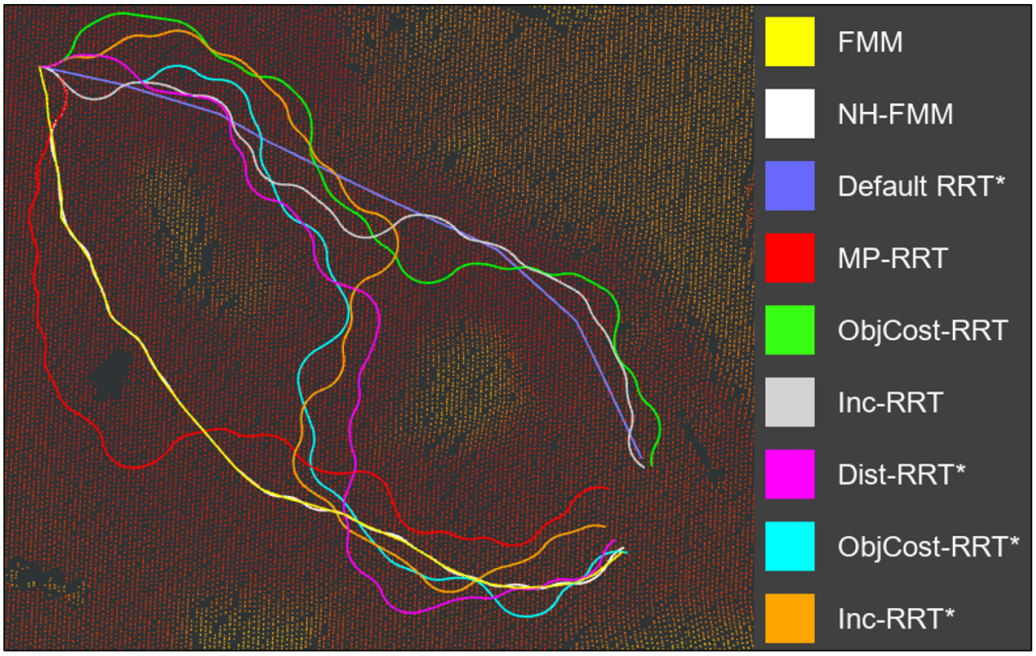


Figure 24: A set of paths generated by all algorithms for the first planning scenario. The background is part of the point cloud used to generate inclination maps for planning. Paths are made more visible by this choice of background. Please refer to Figure 23 for a rough overview of inclination values relevant for these paths.

4.8 Randomly Sampled Scenarios

Although the 5 chosen scenarios in Section 4.7 are different, simply using these scenarios to generate results has the potential to produce unnecessary intentional or unintentional biases (e.g., favoring paths generated by one algorithm over others) in the resulting data. To reduce the effect of such biases, a randomized approach is included. This approach samples 10 different initial and goal configurations from uniform distributions, ensuring that there is sufficient clearance for initial expansion, a path to the goal and that the minimum distance between the initial and goal coordinate is 10 meters. The reason for incorporating these requirements in sampling is that the algorithms sensitive to initial orientation return failure, depending on the initial orientation, if the sampled coordinates are too close to obstacles, or too close to the edges of the map. They also fail to generate paths if the distance between the initial and goal configurations is too short. A single expansion from the root of a tree, results in a path that corresponds to a single motion primitive, with a minimum length corresponding to the minimum possible length of a primitive in the set. Allowing implicit failures to be represented in the resulting data may result in a less meaningful comparison of performance in terms of path lengths, accumulated and average costs. To clarify the meaning of "implicit failures" in this case, consider a scenario where the initial coordinate is at the edge of the traversability map. If the sampled initial orientation results in an initial expansion in a direction toward the outside of the map, then the algorithms will fail because a distinct motion primitive set is used throughout tree-building and this set does not allow for reverse motion or neutral turns. This highlights another limitation of using a restricted motion primitive set to ensure feasibility for vehicles without reverse motion and neutral-turn capabilities, in addition to the limitations discussed in Section 4.4.

The primary takeaway from sections 4.5, 4.7 and 4.8 is that evaluation of new schemes may be improved in terms of transparency with respect to the effectiveness of a proposed scheme, by incorporating a set of algorithms associated with different costs based on the same scheme and scenarios with different terrain variations, to compare performances. The results presented in Section 5, highlight advantages of enabling a more transparent comparison when evaluating a new scheme.

5 Simulation

In this section, a set of planning scenarios and sampled configurations used to generate results are outlined, and the simulation approach is described. The resulting data is used to compare performance of the presented algorithms based on path length, accumulated costs and average costs of the generated paths for chosen and randomly sampled scenarios. The aim of this section is to evaluate performances of NH-FMM and the proposed sampling-based schemes for RRT and RRT* (described in Section 4.1), by comparing to the idealistic benchmark algorithms (i.e., FMM and Default RRT*) based on the chosen and randomly sampled planning scenarios described in sections 4.7 and 4.8. This evaluation is necessary to determine whether the algorithms employing the schemes can effectively produce traversable paths according to the cost formulation given in Section 4.3 when this cost field is used by the algorithms in planning.

5.1 Simulation Setup

The five chosen scenarios, described in Section 4.7, are used to generate the resulting data by running the algorithms 10 times for three values of λ (i.e., $\lambda \in \{1.0, 0.8, 0.6\}$). For each λ-value, an average is computed based on the ten runs, making up a total of 150 runs for the chosen scenarios. For the random scnenarios described in Section 4.8, one sampled scenario is run once for each λ-value, and an average is computed over the 10 sampled scenarios for each value. The randomized approach makes up a total of 30 runs, and standard deviations for the accumulated and average costs are included. Hence, results from a total of 190 runs are represented in the data.

5.2 Results

This section presents the results from the motion planning simulation procedure described in Section 5.1. The provided figures and tables are used as examples for the discussion in Section 5.3. Descriptions of the chosen and randomly sampled scenarios are found in sections 4.7 and 4.8. The data corresponding to λ-values equal to 1.0 and 0.6 is shown for the chosen scenarios 2-5. For clarity, the table indexing in this section, corresponds to that of Appendix B (please refer to this appendix for a complete set of results).

5.2.1 Planning Scenario 1

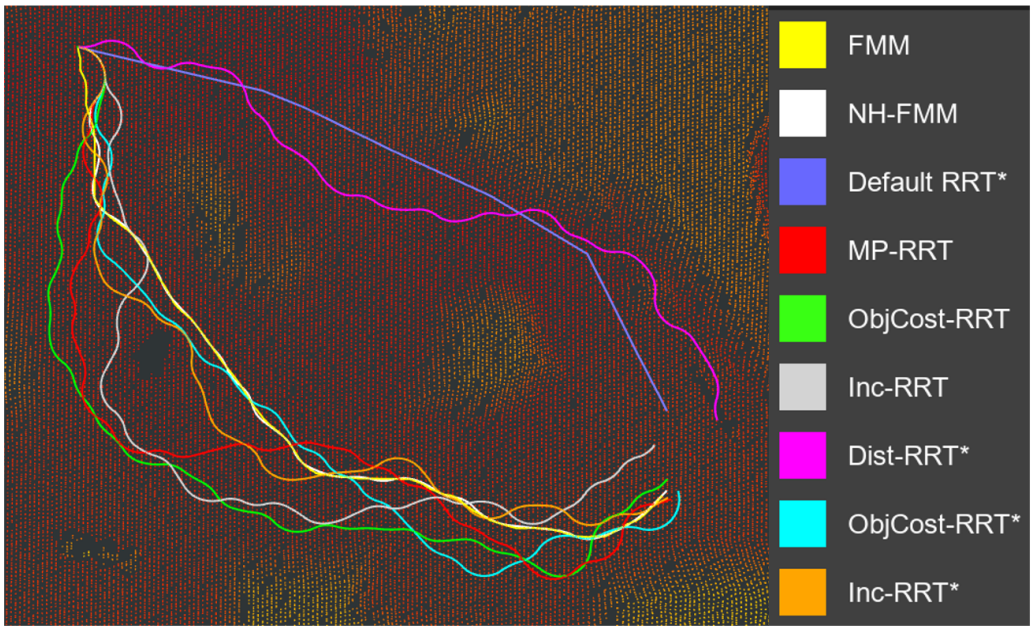


Figure 25: Example paths generated by all algorithms for the first planning scenario with $\lambda = 1.0$. The path generated by NH-FMM (white) is close to the FMM path (yellow) and is therefore barely visible in this picture. Please refer to Section 4.7 for a description of planning scenarios.

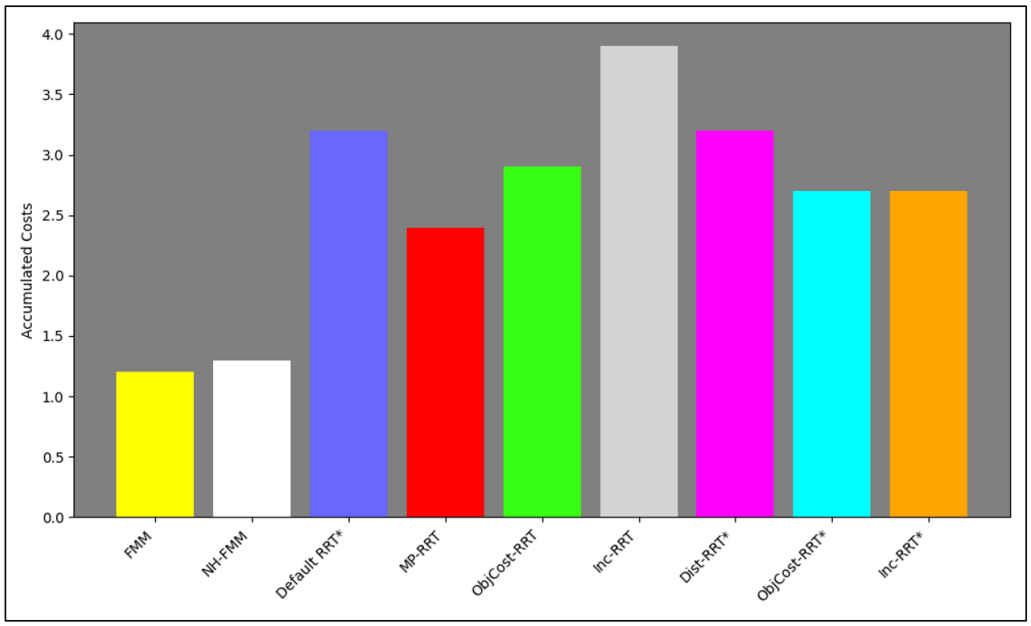


Figure 26: Averages of accumulated costs over 10 runs per algorithm for Scenario 1 with $\lambda = 1.0$. Please refer to Section 4.5 for an overview of algorithms and associated costs.

Table 1 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 1 with $\lambda = 1.0$. The accumulated costs from this table are represented in Figure 26. It can be seen from this table that, on average, Default RRT* produces the shortest path, and NH-FMM and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM.

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	87.6	1.2	0.014
NH-FMM	90.0	1.3	0.014
Default RRT*	72.9	3.2	0.043
MP-RRT	100.7	2.4	0.024
ObjCost-RRT	101.5	2.9	0.029
Inc-RRT	111.8	3.9	0.036
Dist-RRT*	104.1	3.2	0.030
ObjCost-RRT*	106.7	2.7	0.025
Inc-RRT*	101.5	2.7	0.027

Table 1

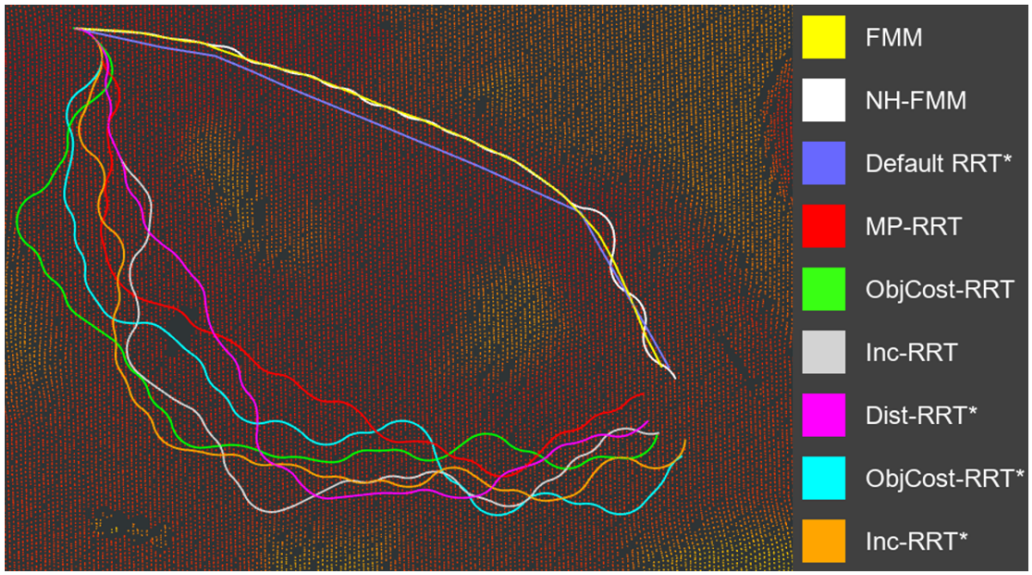


Figure 27: Example paths generated by all algorithms for the first planning scenario with $\lambda = 0.8$. As seen in the picture, when $\lambda = 0.8$, FMM picks a path (yellow) closer to the Default RRT* path (blue), as described in Section 4.7.

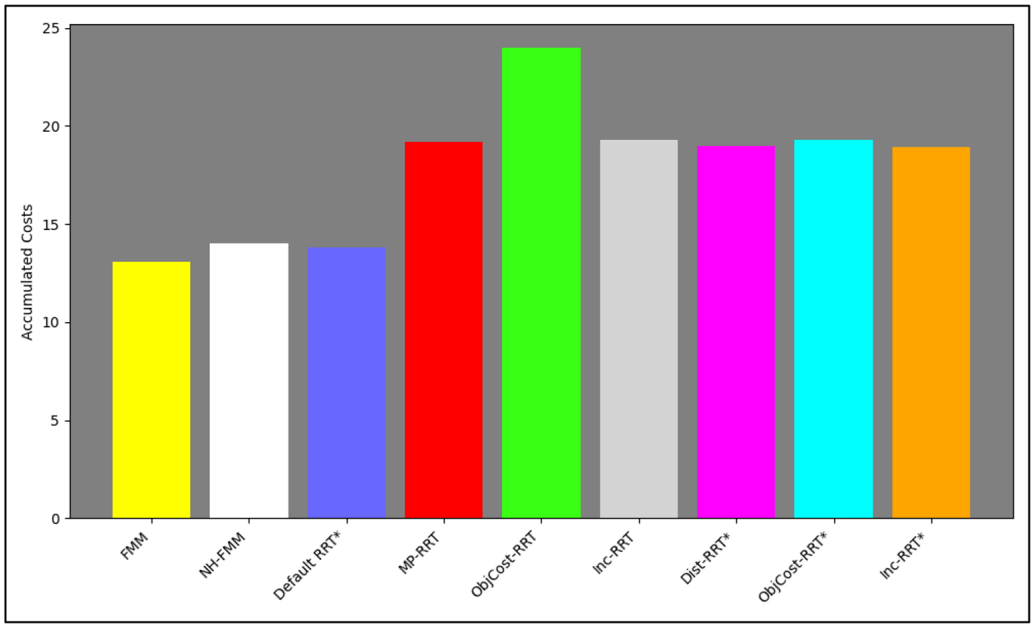


Figure 28: Averages of accumulated costs over 10 runs per algorithm for Scenario 1 with $\lambda = 0.8$.

Table 2 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 1 with $\lambda = 0.8$. The accumulated costs from this table are represented in Figure 28. On average, Default RRT* produces the shortest path, and NH-FMM and FMM produces paths with lowest average costs. FMM obtains the lowest accumulated cost. The accumulated cost for Default RRT* with $\lambda = 0.8$ is closer to the accumulated cost for FMM, compared to that of the other algorithms.

$\lambda = 0.8$	Length (m)	Acc. cost	Avg. cost
FMM	72.6	13.1	0.181
NH-FMM	77.1	14.0	0.181
Default RRT*	72.4	13.8	0.190
MP-RRT	102.0	19.2	0.188
ObjCost-RRT	123.2	24.0	0.194
Inc-RRT	102.0	19.3	0.189
Dist-RRT*	100.5	19.0	0.189
ObjCost-RRT*	102.4	19.3	0.188
Inc-RRT*	99.4	18.9	0.190

Table 2

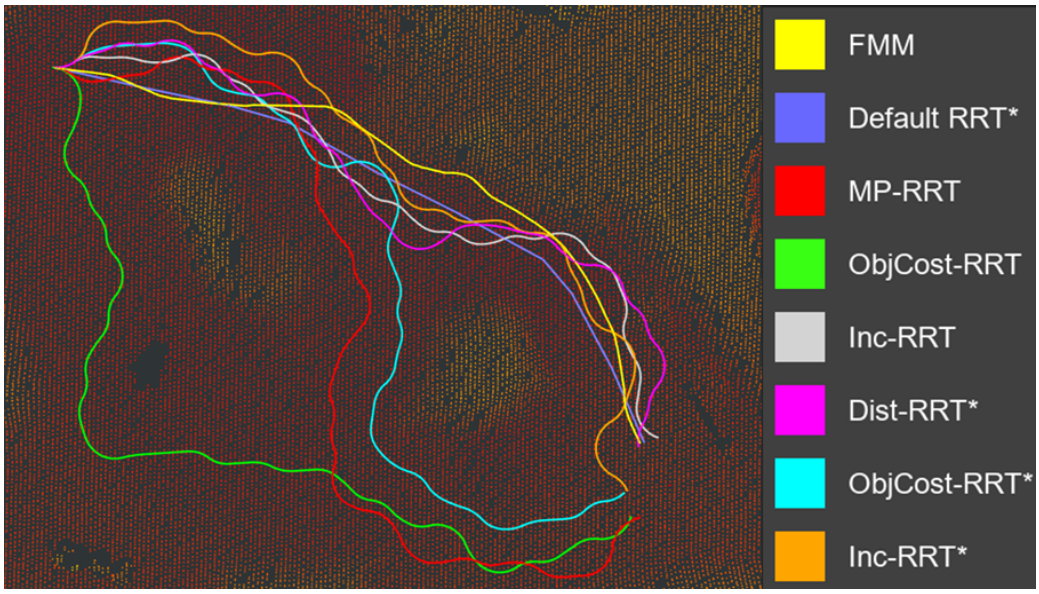


Figure 29: Example paths generated by all algorithms for the first planning scenario with $\lambda = 0.6$. For Scenario 1 with $\lambda = 0.6$, NH-FMM consistently fails to produce a solution. It can be seen from this figure that FMM chooses a path that is closer to the RRT* path compared to Figure 25, similar to the FMM and RRT* example paths represented in Figure 27.

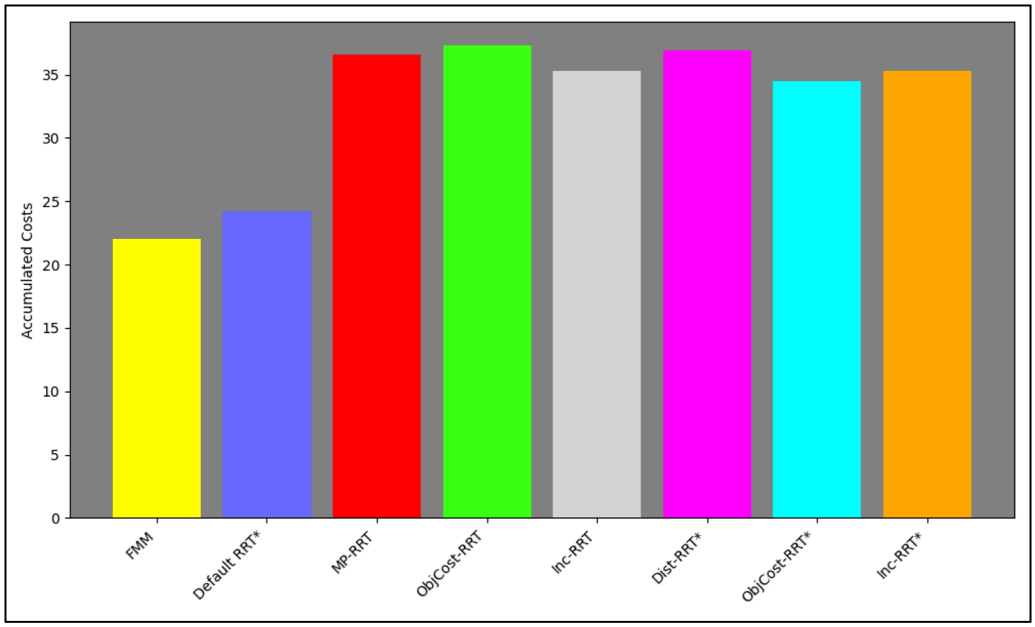


Figure 30: Averages of accumulated costs over 10 runs per algorithm for Scenario 1 with $\lambda = 0.6$.

Table 3 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 1 with $\lambda = 0.6$. The accumulated costs from this table are represented in Figure 30. On average, Default RRT* produces the shortest path, and Default RRT* and FMM produces paths with lowest average costs. It can be seen that FMM also obtains the lowest accumulated cost. The table shows that the accumulated cost for Default RRT* with $\lambda = 0.6$ is closer to the accumulated cost for FMM with respect to accumulated costs of the sampling-based variants.

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	75.6	22.0	0.291
NH-FMM	-	-	-
Default RRT*	72.5	24.2	0.334
MP-RRT	105.0	36.6	0.349
ObjCost-RRT	107.7	37.3	0.345
Inc-RRT	100.7	35.3	0.351
Dist-RRT*	105.4	36.9	0.350
ObjCost-RRT*	99.0	34.5	0.348
Inc-RRT*	100.5	35.3	0.352

Table 3

5.2.2 Planning Scenario 2

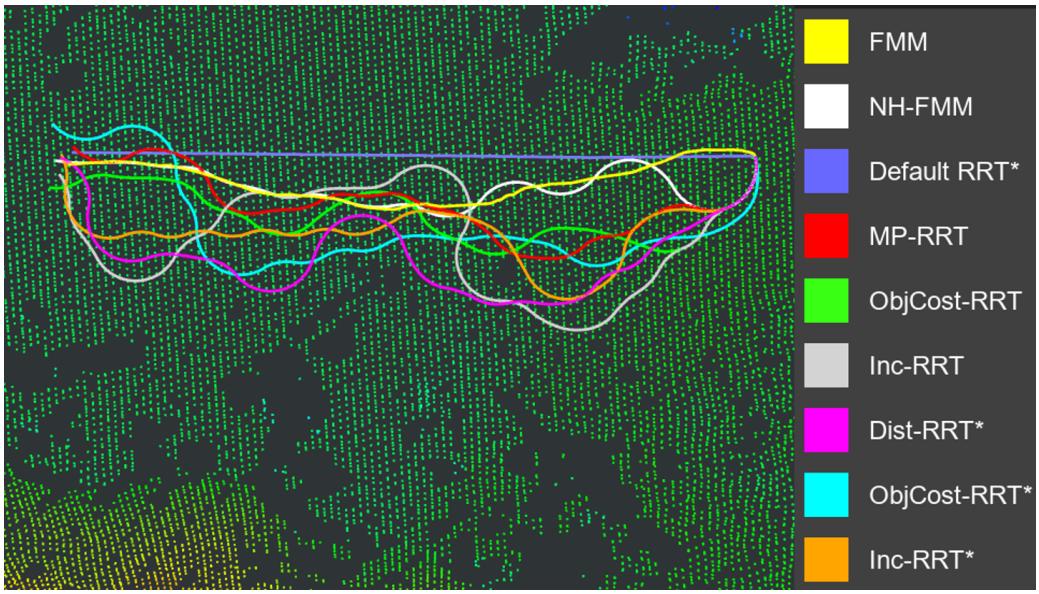


Figure 31: Example paths generated by all algorithms for the second planning scenario with $\lambda = 1.0$. It can be seen that compared to Figure 25, Section 5.2.1, the FMM (yellow) and Default RRT* (blue) paths are closer for $\lambda = 1.0$ in this scenario.

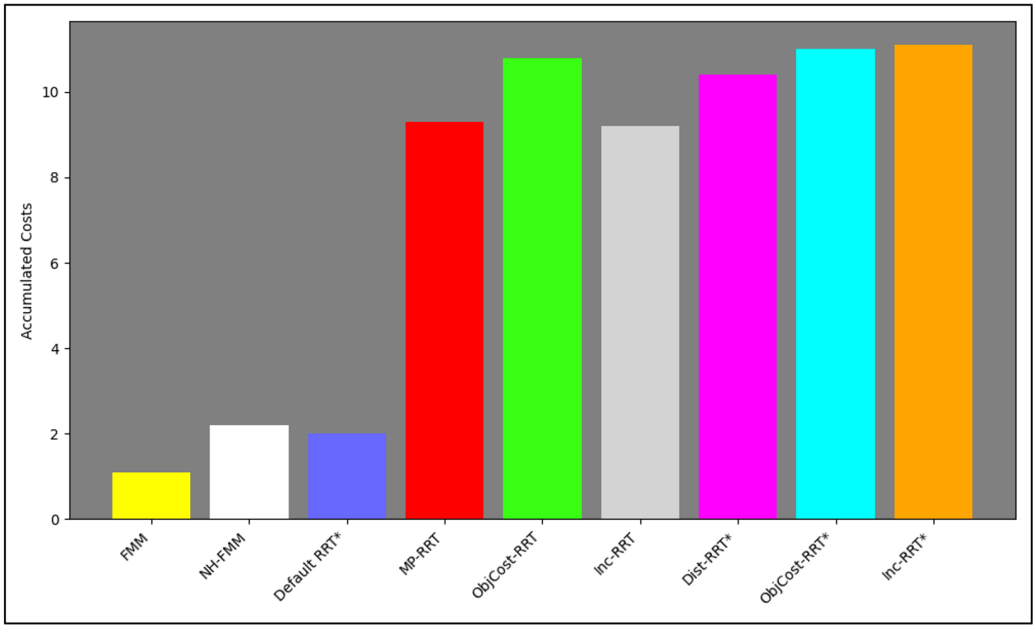


Figure 32: Averages of accumulated costs over 10 runs per algorithm for Scenario 2 with $\lambda = 1.0$.

Table 4 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 2 with $\lambda = 1.0$. The accumulated costs from this table are represented in Figure 32. It can be seen from this table that, on average, Default RRT* produces the shortest path, and NH-FMM and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM. This table shows that similar to Scenario 1 (Section 5.2.1) for $\lambda = 0.8$, and $\lambda = 0.6$, the accumulated cost for Default RRT* is closer to that of FMM compared to accumulated costs for the sampling-based variants.

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	48.2	1.1	0.022
NH-FMM	53.6	2.2	0.040
Default RRT*	46.6	2.0	0.043
MP-RRT	59.8	9.3	0.155
ObjCost-RRT	63.0	10.8	0.168
Inc-RRT	61.0	9.2	0.150
Dist-RRT*	60.0	10.4	0.170
ObjCost-RRT*	61.3	11.0	0.179
Inc-RRT*	60.8	11.1	0.183

Table 4

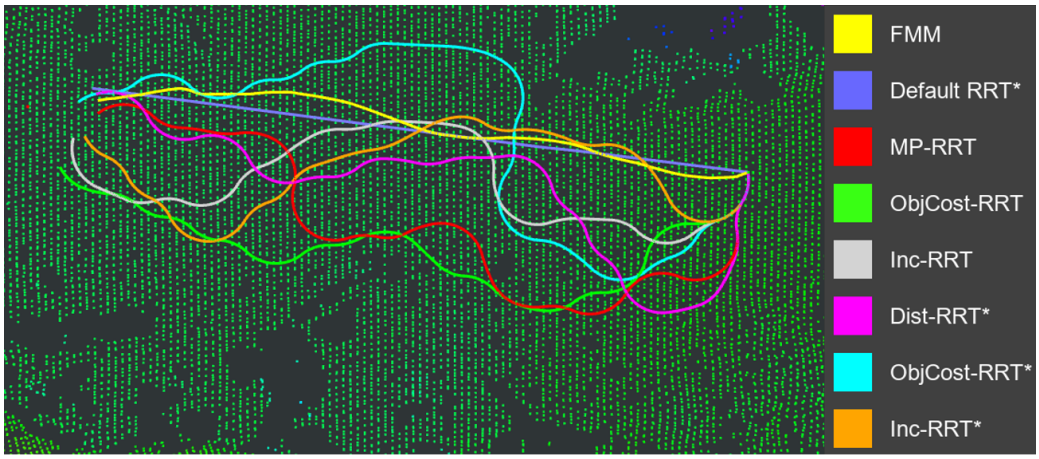


Figure 33: Example paths generated by all algorithms for the second planning scenario with $\lambda = 0.6$. For Scenario 2 with $\lambda = 0.6$, NH-FMM consistently fails to produce a solution. Compared to Figure 31, the FMM (yellow) and Default RRT* (blue) paths are closer for $\lambda = 0.6$.

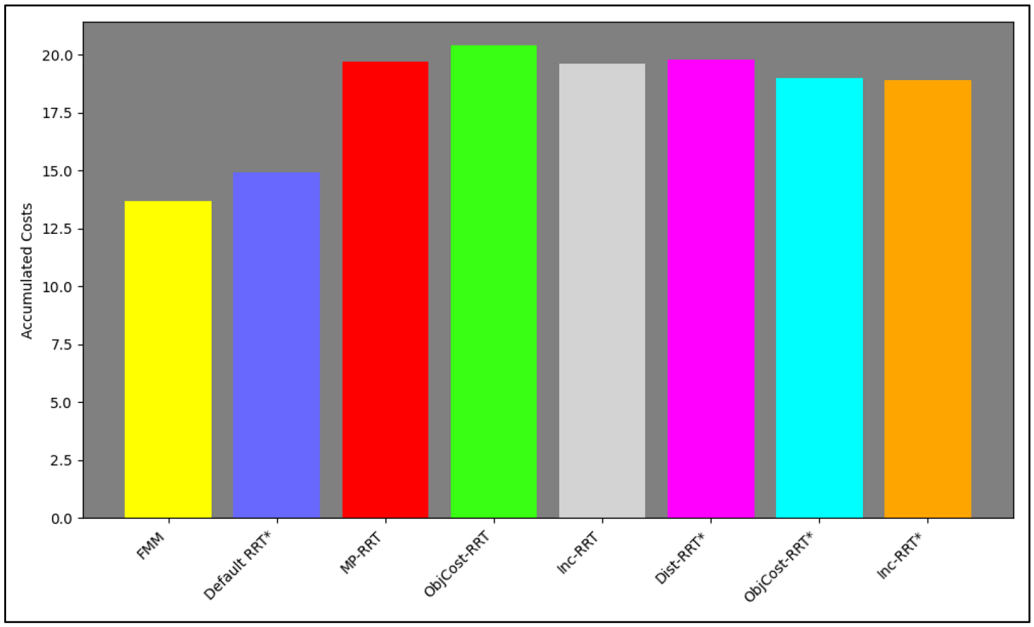


Figure 34: Averages of accumulated costs over 10 runs per algorithm for Scenario 2 with $\lambda = 0.6$.

Table 6 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 2 with $\lambda = 0.6$. The accumulated costs from this table are represented in Figure 34. It can be seen from this table that, on average, Default RRT* produces the shortest path, and ObjCost-RRT and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM. The accumulated cost for Default RRT* is closer to that of FMM, compared to that of the sampling-based variants.

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	47.2	13.7	0.290
NH-FMM	-	-	-
Default RRT*	46.5	14.9	0.321
MP-RRT	59.5	19.7	0.331
ObjCost-RRT	64.3	20.4	0.317
Inc-RRT	60.4	19.6	0.325
Dist-RRT*	61.0	19.8	0.325
ObjCost-RRT*	58.9	19.0	0.323
Inc-RRT*	58.3	18.9	0.325

Table 6

5.2.3 Planning Scenario 3

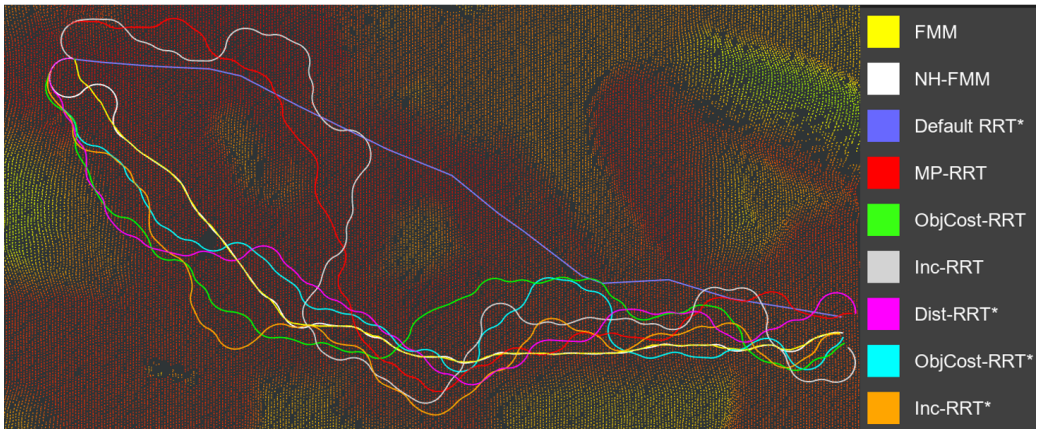


Figure 35: Example paths generated by all algorithms for the third planning scenario with $\lambda = 1.0$. The NH-FMM path (white) is close to the FMM path (yellow), and therefore barely visible.

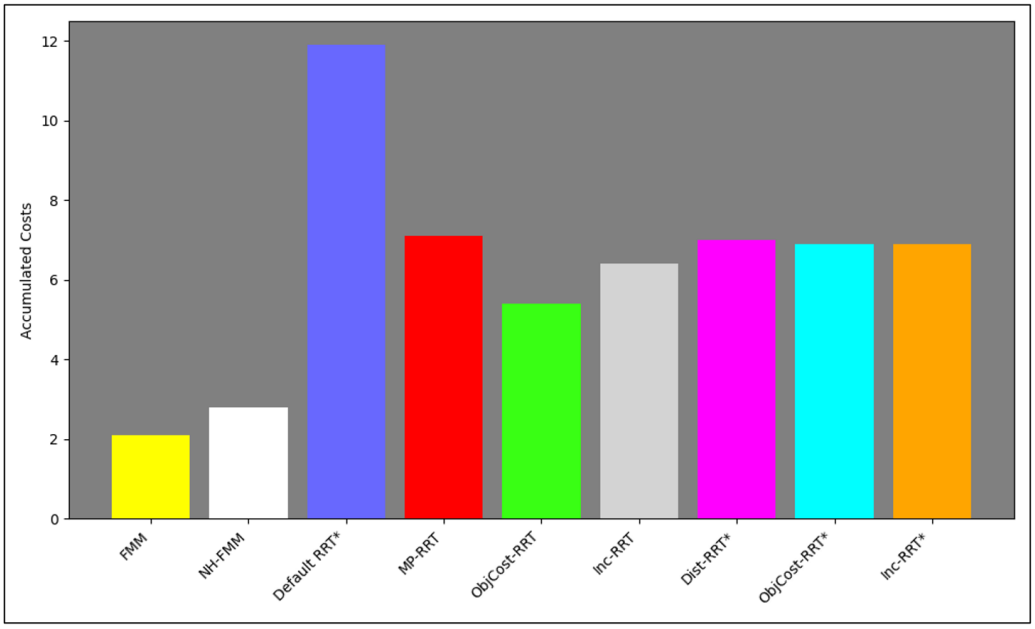


Figure 36: Averages of accumulated costs over 10 runs per algorithm for Scenario 3 with $\lambda = 1.0$.

Table 7 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 3 with $\lambda = 1.0$. The accumulated costs from this table are represented in Figure 36. On average, Default RRT* produces the shortest path, and NH-FMM and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated costs for the sampling-based variants and NH-FMM are closer to the accumulated cost for FMM, compared to that of Default RRT*.

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	141.7	2.1	0.015
NH-FMM	154.2	2.8	0.018
Default RRT*	128.5	11.9	0.093
MP-RRT	175.4	7.1	0.040
ObjCost-RRT	166.4	5.4	0.033
Inc-RRT	172.9	6.4	0.037
Dist-RRT*	174.1	7.0	0.041
ObjCost-RRT*	175.4	6.9	0.040
Inc-RRT*	175.6	6.9	0.039

Table 7

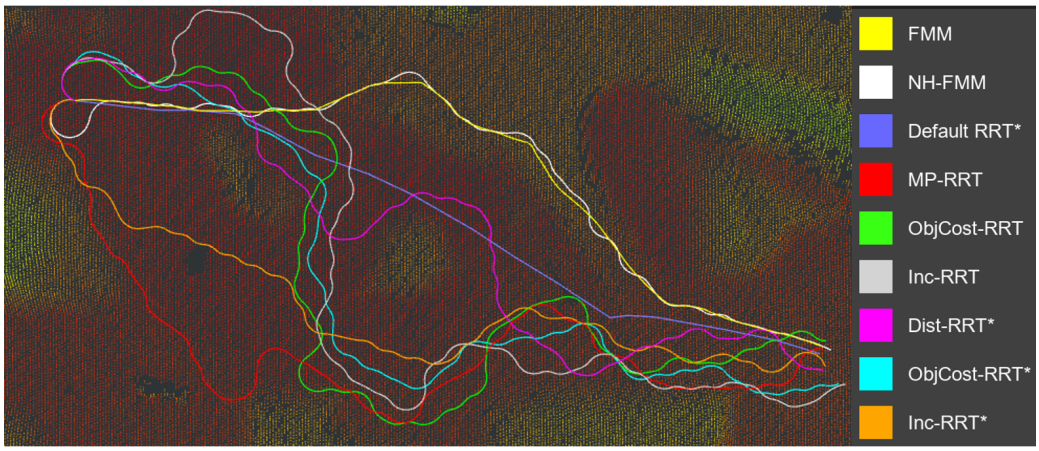


Figure 37: Example paths generated by all algorithms for the third planning scenario with $\lambda = 0.6$. In this case, FMM chooses a path (yellow) through a narrow passage to a constrained region of higher elevation. From the constrained region, the FMM path follows another narrow passage to an open region of lower elevation to reach the goal. For Scenario 3 with $\lambda = 0.6$, NH-FMM consistently finds a path.

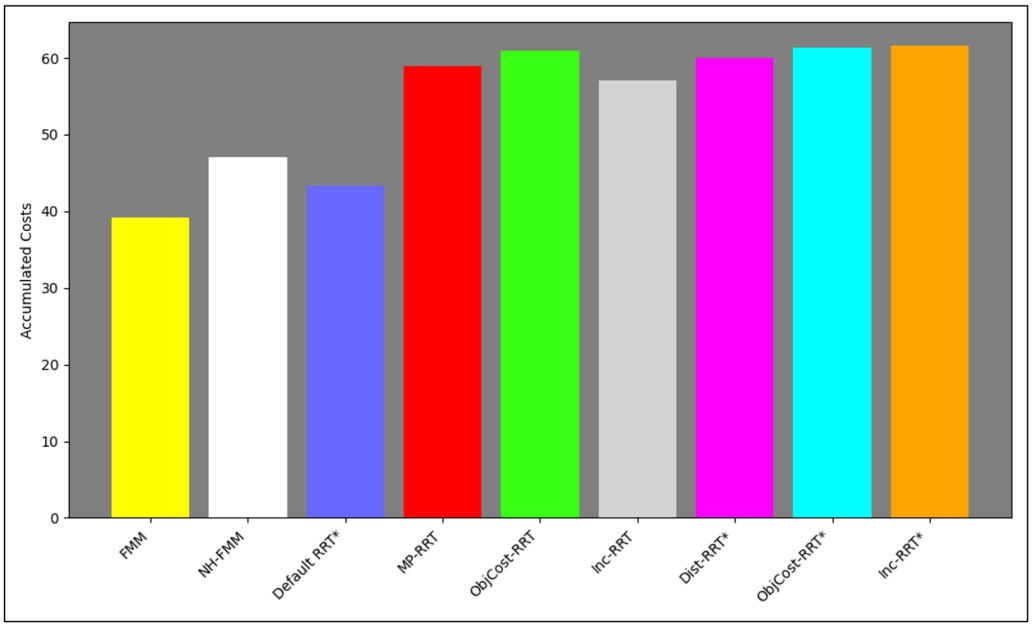


Figure 38: Averages of accumulated costs over 10 runs per algorithm for Scenario 3 with $\lambda = 0.6$.

Table 9 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 3 with $\lambda = 0.6$. The accumulated costs from this table are represented in Figure 38. On average, Default RRT* produces the shortest path, and NH-FMM and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated cost for Default RRT* is closer to the accumulated cost for FMM, compared to that of the other algorithms.

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	135.9	39.2	0.288
NH-FMM	156.4	47.0	0.300
Default RRT*	128.6	43.3	0.336
MP-RRT	172.9	59.0	0.341
ObjCost-RRT	178.9	60.9	0.341
Inc-RRT	167.7	57.1	0.341
Dist-RRT*	174.6	60.0	0.344
ObjCost-RRT*	177.4	61.3	0.345
Inc-RRT*	179.7	61.6	0.343

Table 9

5.2.4 Planning Scenario 4

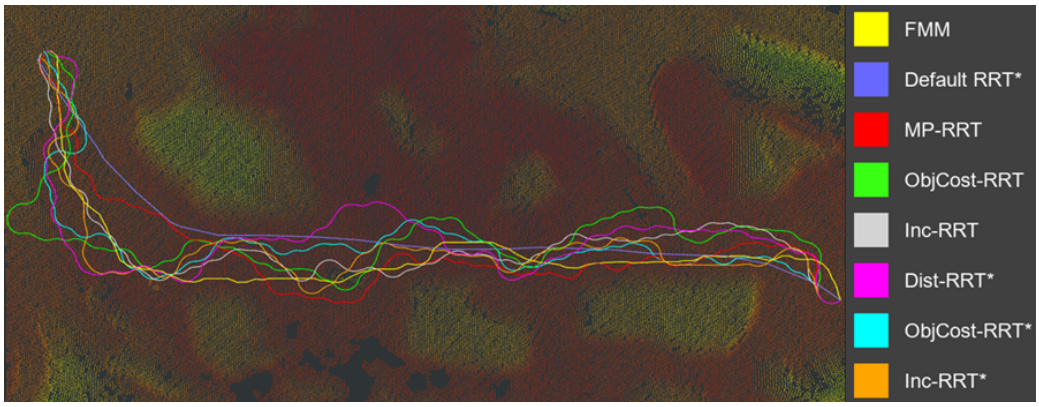


Figure 39: Example paths generated by all algorithms for the fourth planning scenario with $\lambda = 1.0$. For this scenario, NH-FMM consistently fails to produce solution paths.

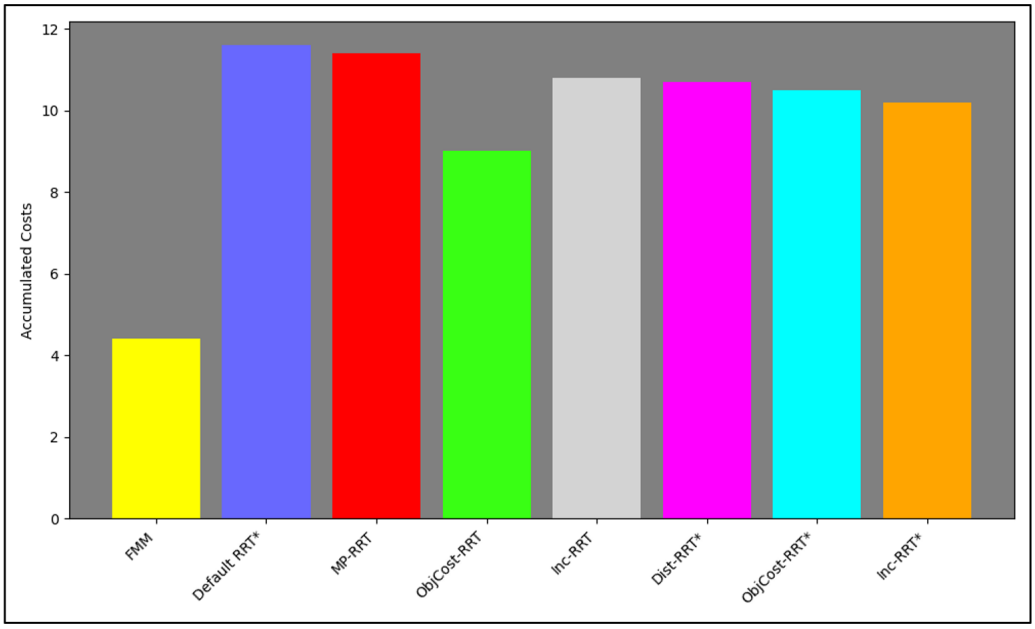


Figure 40: Averages of accumulated costs over 10 runs per algorithm for Scenario 4 with $\lambda = 1.0$.

Table 10 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 4 with $\lambda = 1.0$. The accumulated costs from this table are represented in Figure 40. On average, Default RRT* produces the shortest path, and ObjCost-RRT and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated cost for Default RRT* is higher, compared to that of the sampling-based variants.

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	219.1	4.4	0.020
NH-FMM	-	-	-
Default RRT*	194.2	11.6	0.060
MP-RRT	246.3	11.4	0.046
ObjCost-RRT	231.3	9.0	0.039
Inc-RRT	247.6	10.8	0.043
Dist-RRT*	241.4	10.7	0.044
ObjCost-RRT*	245.9	10.5	0.043
Inc-RRT*	249.8	10.2	0.041

Table 10

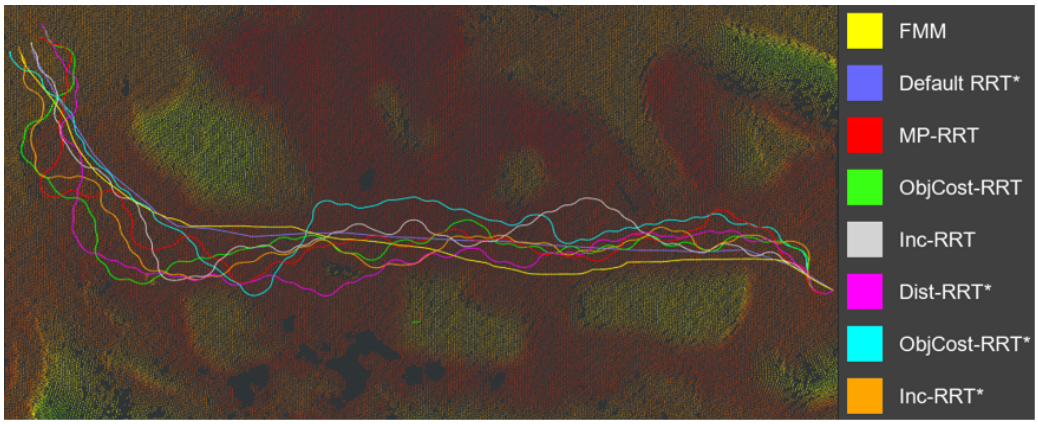


Figure 41: Example paths generated by all algorithms for the fourth planning scenario with $\lambda = 0.6$. For this scenario, NH-FMM consistently fails to produce solution paths. Compared to Figure 39, the FMM path (yellow) and Default RRT* (blue) path are closer.

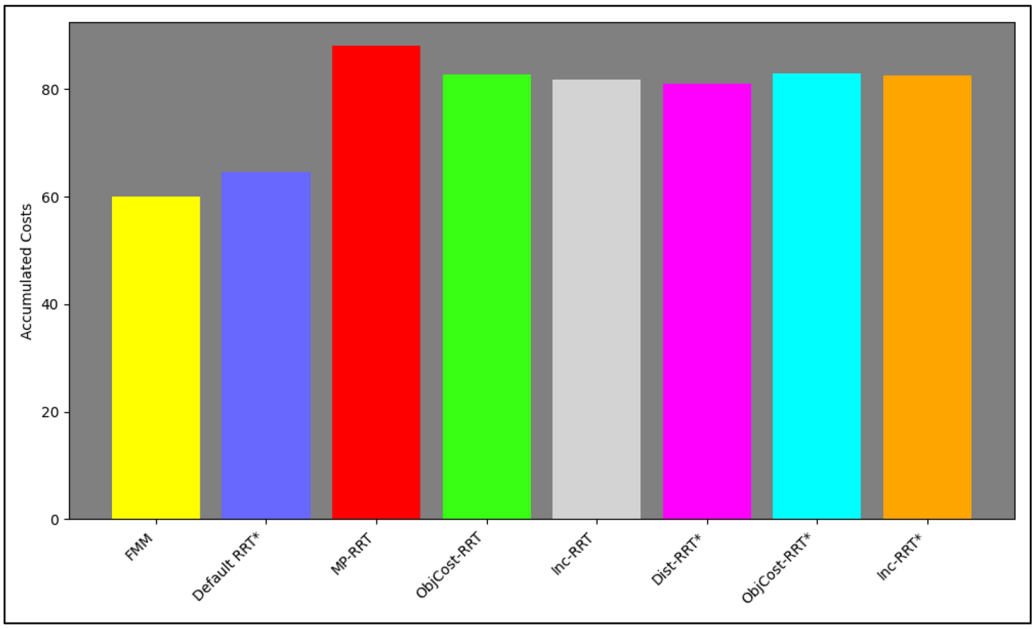


Figure 42: Averages of accumulated costs over 10 runs per algorithm for Scenario 4 with $\lambda = 0.6$.

Table 12 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 4 with $\lambda = 0.6$. The accumulated costs from this table are represented in Figure 42. On average, Default RRT* produces the shortest path, and Default RRT* and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated cost for Default RRT* is lower, compared to the sampling-based variants.

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	196.6	60.1	0.306
NH-FMM	-	-	-
Default RRT*	193.8	64.7	0.334
MP-RRT	259.6	88.1	0.339
ObjCost-RRT	242.5	82.8	0.341
Inc-RRT	239.5	81.9	0.342
Dist-RRT*	237.8	81.0	0.341
ObjCost-RRT*	242.3	83.0	0.343
Inc-RRT*	241.2	82.5	0.342

Table 12

5.2.5 Planning Scenario 5

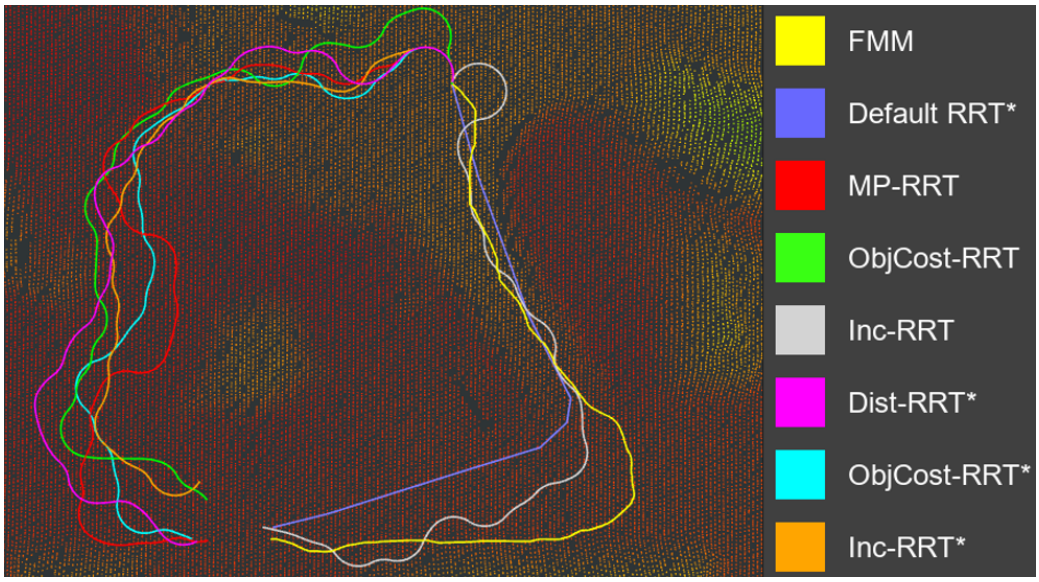


Figure 43: Example paths generated by all algorithms for the fifth planning scenario with $\lambda = 1.0$. For this scenario, NH-FMM consistently fails to produce solution paths.

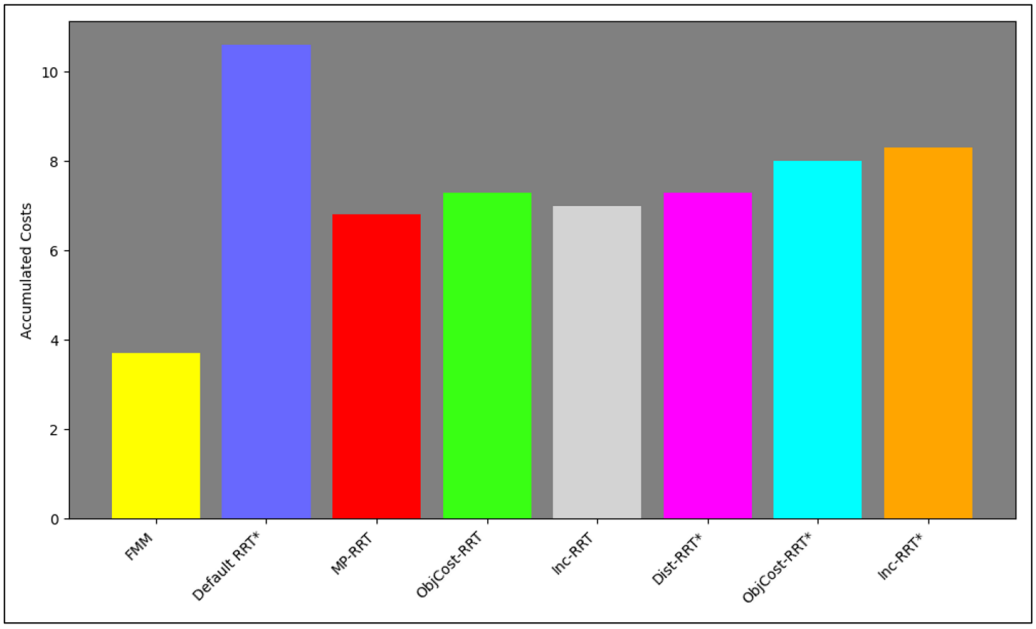


Figure 44: Averages of accumulated costs over 10 runs per algorithm for Scenario 5 with $\lambda = 1.0$.

Table 13 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 5 with $\lambda = 1.0$. The accumulated costs from this table are represented in Figure 44. On average, Default RRT* produces the shortest path, and MP-RRT and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated cost for Default RRT* is higher, compared to that of the sampling-based variants.

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	94.1	3.7	0.039
NH-FMM	-	-	-
Default RRT*	72.8	10.6	0.146
MP-RRT	105.2	6.8	0.066
ObjCost-RRT	103.5	7.3	0.070
Inc-RRT	106.7	7.0	0.068
Dist-RRT*	106.9	7.3	0.069
ObjCost-RRT*	102.8	8.0	0.078
Inc-RRT*	117.4	8.3	0.074

Table 13

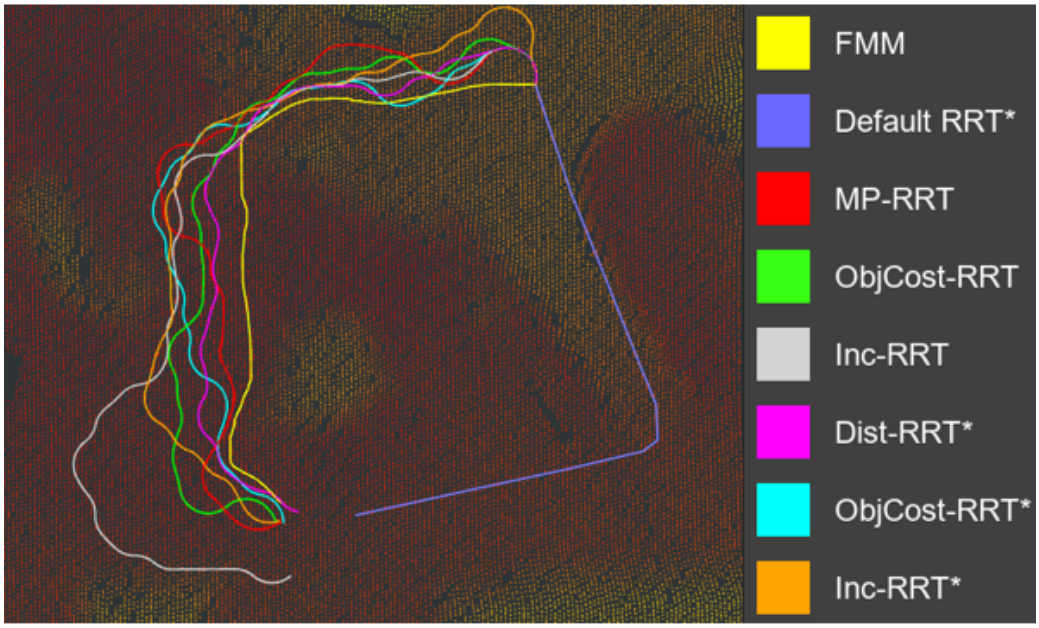


Figure 45: Example paths generated by all algorithms for the fifth planning scenario with $\lambda = 0.6$. For this scenario, NH-FMM consistently fails to produce solution paths. Compared to Figure 43, FMM chooses a path (yellow) through a different passage to the goal.

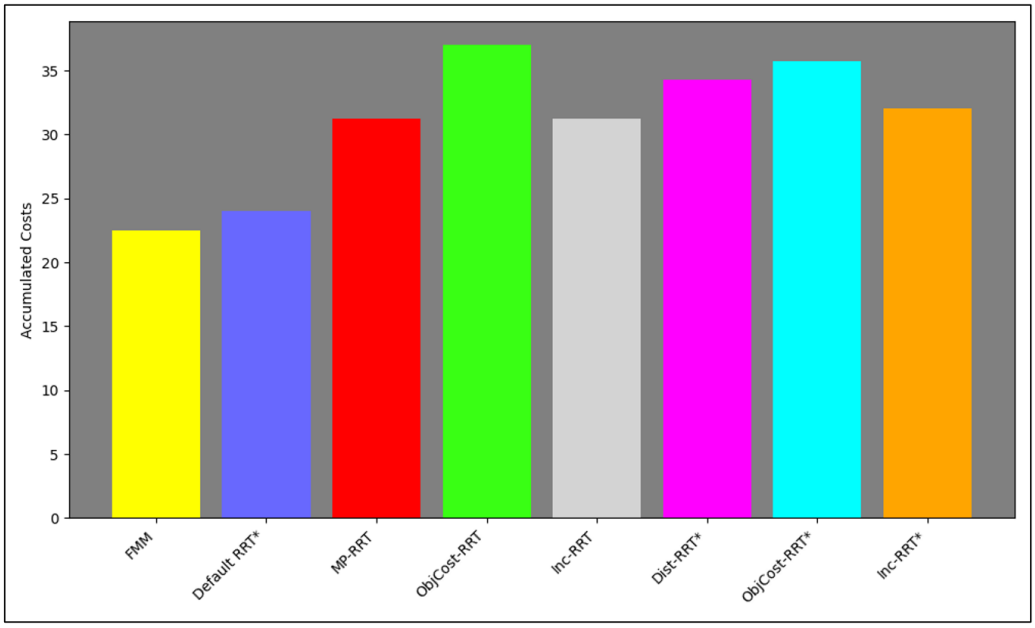


Figure 46: Averages of accumulated costs over 10 runs per algorithm for Scenario 5 with $\lambda = 0.6$.

Table 15 displays averages of path lengths, accumulated costs and average costs computed over 10 runs for Scenario 5 with $\lambda = 0.6$. The accumulated costs from this table are represented in Figure 46. On average, Default RRT* produces the shortest path, and Inc-RRT* and FMM produces paths with lowest average costs. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated cost for Default RRT* is lower, compared to the sampling-based variants.

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	73.6	22.5	0.306
NH-FMM	-	-	-
Default RRT*	72.8	24.0	0.330
MP-RRT	96.6	31.2	0.322
ObjCost-RRT	113.7	37.0	0.323
Inc-RRT	96.8	31.2	0.322
Dist-RRT*	105.6	34.3	0.324
ObjCost-RRT*	110.1	35.7	0.323
Inc-RRT*	99.6	32.0	0.321

Table 15

5.2.6 Random Simulations

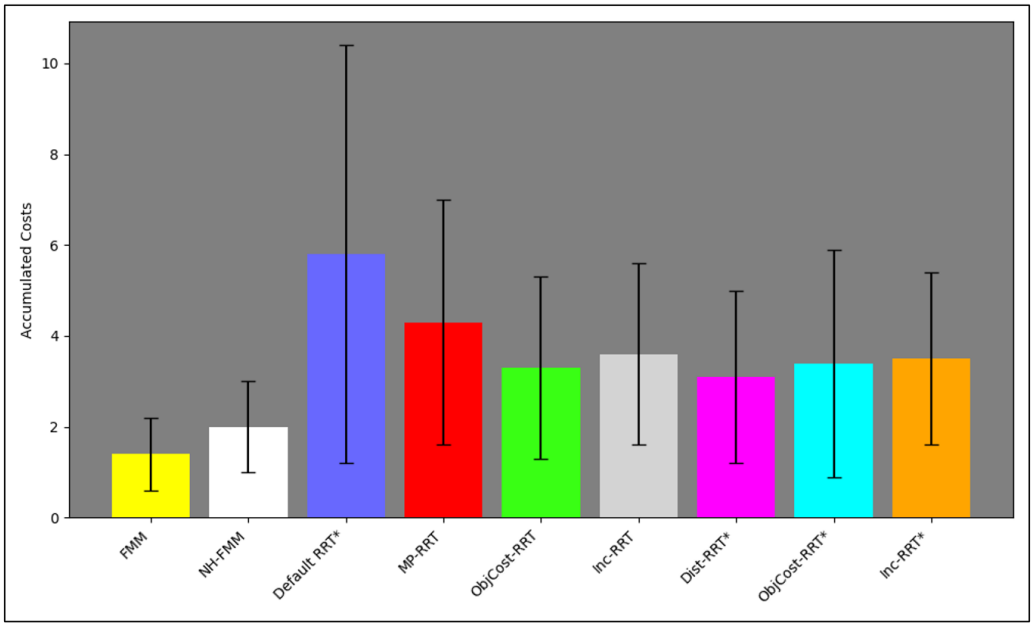


Figure 47: Averages of accumulated costs over 10 sampled scenarios for $\lambda = 1.0$, with error bars showing standard deviations of accumulated costs.

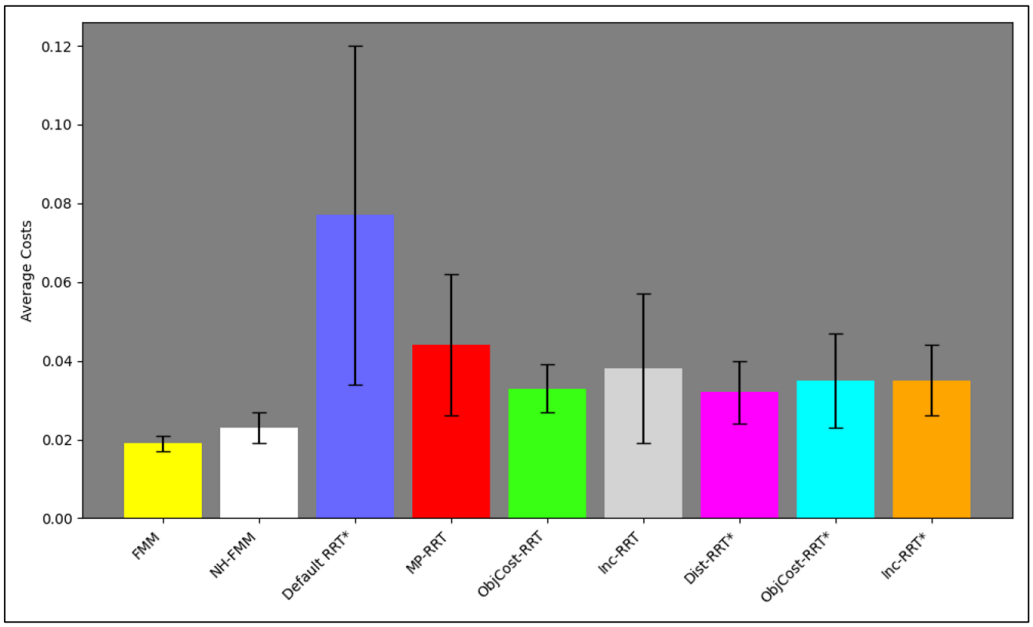


Figure 48: Averages of average costs over 10 sampled scenarios for $\lambda = 1.0$, with error bars showing standard deviations of average costs.

Table 16 displays averages of path lengths, accumulated costs and average costs computed over 10 randomly sampled scenarios with $\lambda = 1.0$. On average, Default RRT* produces the shortest path, and NH-FMM and FMM produces paths with lowest average costs. For the 10 randomly sampled scenarios with $\lambda = 1.0$, NH-FMM consistently finds solution paths. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated cost, average cost and corresponding standard deviations for Default RRT* are higher, compared to the other algorithms. The accumulated costs, average costs and corresponding standard deviations of this table are shown in figures 47 and 48.

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost	Acc. cost (σ_s)	Avg. cost (σ_s)
FMM	74.4	1.4	0.019	0.8	0.002
NH-FMM	87.0	2.0	0.023	1.0	0.004
Default RRT*	65.3	5.8	0.077	4.6	0.043
MP-RRT	97.7	4.3	0.044	2.7	0.018
ObjCost-RRT	97.7	3.3	0.033	2.0	0.006
Inc-RRT	95.7	3.6	0.038	2.0	0.019
Dist-RRT*	95.7	3.1	0.032	1.9	0.008
ObjCost-RRT*	93.0	3.4	0.035	2.5	0.012
Inc-RRT*	100.2	3.5	0.035	1.9	0.009

Table 16

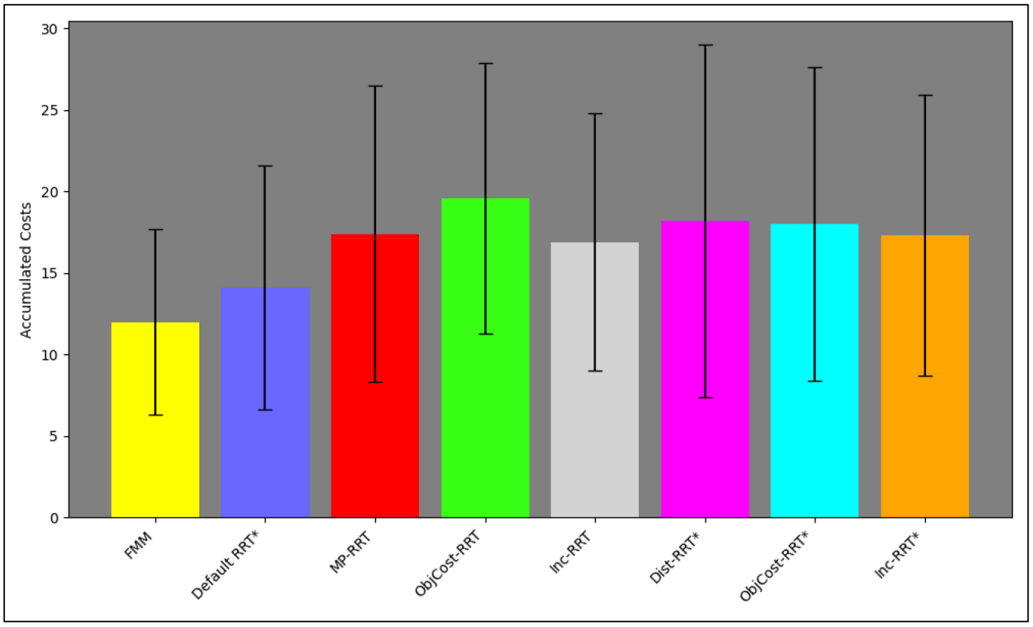


Figure 49: Averages of accumulated costs over 10 sampled scenarios for $\lambda = 0.8$, with error bars showing standard deviations of accumulated costs.

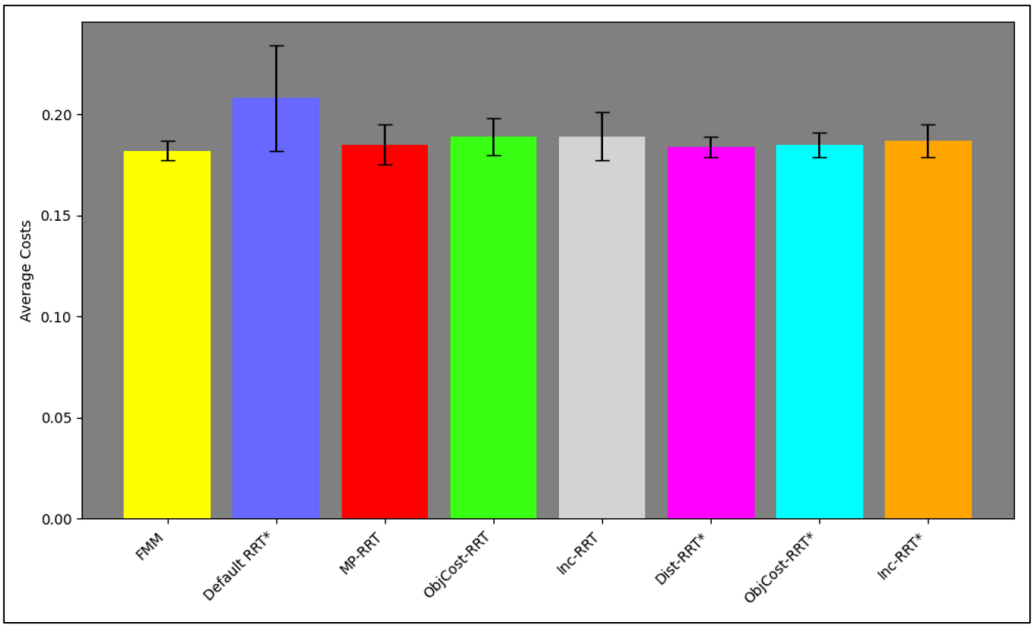


Figure 50: Averages of average costs over 10 sampled scenarios for $\lambda = 0.8$, with error bars showing standard deviations of average costs.

Table 17 displays averages of path lengths, accumulated costs and average costs computed over 10 randomly sampled scenarios with $\lambda = 0.8$. On average, Default RRT* produces the shortest path, and Dist-RRT* and FMM produces paths with lowest average costs. NH-FMM finds paths in 6 out 10 scenarios. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated cost and the corresponding standard deviation for Default RRT* is lower, compared to the sampling-based variants. The average cost and corresponding standard deviation is higher for Default RRT*, compared to FMM and the sampling-based variants. The accumulated costs, average costs and corresponding standard deviations of this table are shown in figures 49 and 50.

$\lambda = 0.8$	Length (m)	Acc. cost	Avg. cost	Acc. cost (σ_s)	Avg. cost (σ_s)
FMM	65.5	12.0	0.182	5.7	0.005
NH-FMM	6 out of 10 passed				
Default RRT*	65.2	14.1	0.208	7.5	0.026
MP-RRT	92.3	17.4	0.185	9.1	0.010
ObjCost-RRT	103.5	19.6	0.189	8.3	0.009
Inc-RRT	89.1	16.9	0.189	7.9	0.012
Dist-RRT*	97.9	18.2	0.184	10.8	0.005
ObjCost-RRT*	96.0	18.0	0.185	9.6	0.006
Inc-RRT*	91.5	17.3	0.187	8.6	0.008

Table 17

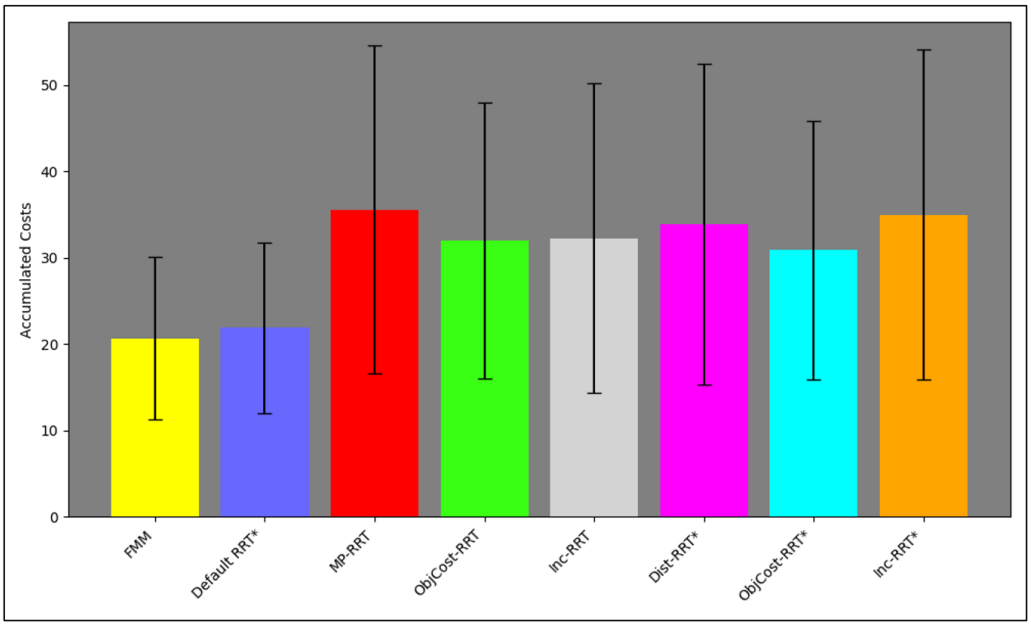


Figure 51: Averages of accumulated costs over 10 sampled scenarios for $\lambda = 0.6$, with error bars showing standard deviations of accumulated costs.

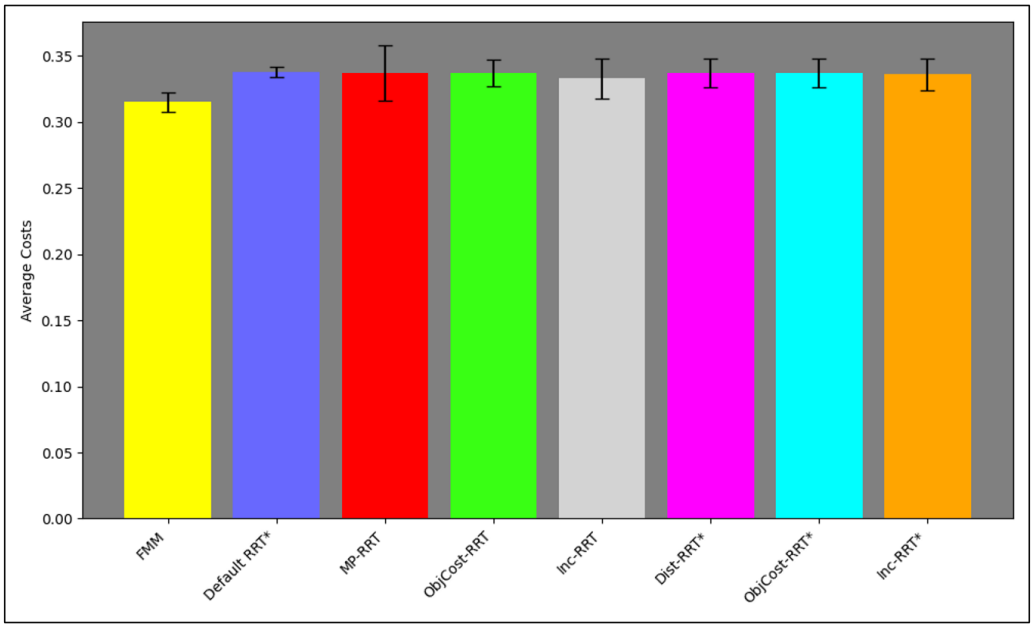


Figure 52: Averages of average costs over 10 sampled scenarios for $\lambda = 0.6$, with error bars showing standard deviations of average costs.

Table 18 displays averages of path lengths, accumulated costs and average costs computed over 10 randomly sampled scenarios with $\lambda = 0.6$. On average, Default RRT* produces the shortest path, and Inc-RRT and FMM produces paths with lowest average costs. NH-FMM finds paths in 2 out 10 scenarios. The lowest accumulated cost is given by FMM. It can be seen from this table that the accumulated cost and the corresponding standard deviation for Default RRT* is lower, compared to that of the sampling-based variants. The average cost is higher and the corresponding standard deviation is lower for Default RRT*, compared to FMM and the sampling-based variants. The accumulated costs, average costs and corresponding standard deviations of this table are shown in figures 51 and 52.

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost	Acc. cost (σ_s)	Avg. cost (σ_s)
FMM	65.6	20.7	0.315	9.4	0.007
NH-FMM	2 out of 10 passed				
Default RRT*	65.0	21.9	0.338	9.9	0.004
MP-RRT	103.7	35.6	0.337	19.0	0.021
ObjCost-RRT	94.0	32.0	0.337	16.0	0.010
Inc-RRT	95.7	32.3	0.333	17.9	0.015
Dist-RRT*	99.6	33.9	0.337	18.6	0.011
ObjCost-RRT*	90.8	30.9	0.337	15.0	0.011
Inc-RRT*	103.5	35.0	0.336	19.1	0.012

Table 18

5.2.7 Potentially Problematic Scenarios for the Sampling-Based Schemes

For the chosen and randomly sampled scenarios, no failures were observed for the motion primitive schemes. However, testing beyond these scenarios revealed that the motion primitive variants of RRT and RRT* can produce failure in certain scenarios, for example in planning scenarios featuring larger regions between the start and goal positions and goal regions situated in constrained environments. Figure 53 provides examples of such scenarios, where failures can occur for the proposed schemes. The red circles indicate approximate goal regions for these scenarios.

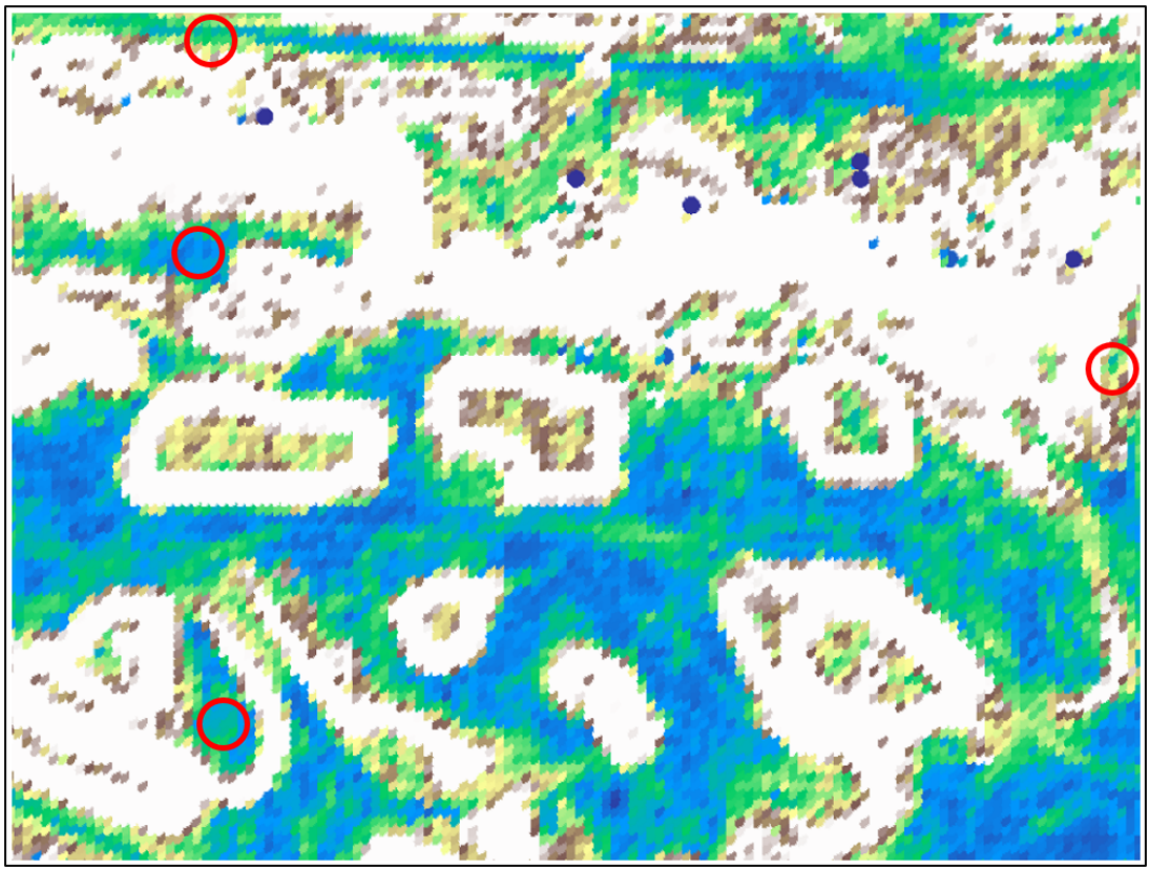


Figure 53: This figure provides some examples of potentially problematic goal regions for the sampling-based variants. The red circles indicate approximate goal regions, for which failures may occur for these algorithms.

5.3 Discussion

For the chosen and random scenarios, two main trends are observed in the data. The first trend concerns the scenarios when $\lambda = 1.0$. Compared to averages of accumulated costs for Default RRT*, in scenarios 3, 4, 5 and the randomly sampled scenarios (for $\lambda = 1.0$), the sampling-based variants obtain lower averages of accumulated costs. In Scenario 1, a majority of the sampling-based algorithms obtain lower accumulated costs compared to Default RRT*, whereas in Scenario 2, the sampling-based variants obtain higher accumulated costs on average, compared to Default RRT*. The second trend concerns $\lambda = 0.8$ and $\lambda = 0.6$. For these λ-values, Default RRT* consistently obtains lower accumulated costs on average, compared to the sampling-based variants (please refer to Appendix B). There are no obvious trends between the sampling-based variants with respect to averages of accumulated costs and average costs.

The first planning scenario consists of an open planning environment with three prominent regions between the start and goal positions. The first is partially constrained and is characterized by a region of higher inclinations and more variations in inclination values. In the second region, the terrain is characterized by less variations in inclination values, and lower inclination values compared to the first region. The third region consists of terrain characterized by less variations in inclination and lower inclination values compared to the first and second regions. Figure 54 gives a rough overview of these regions.

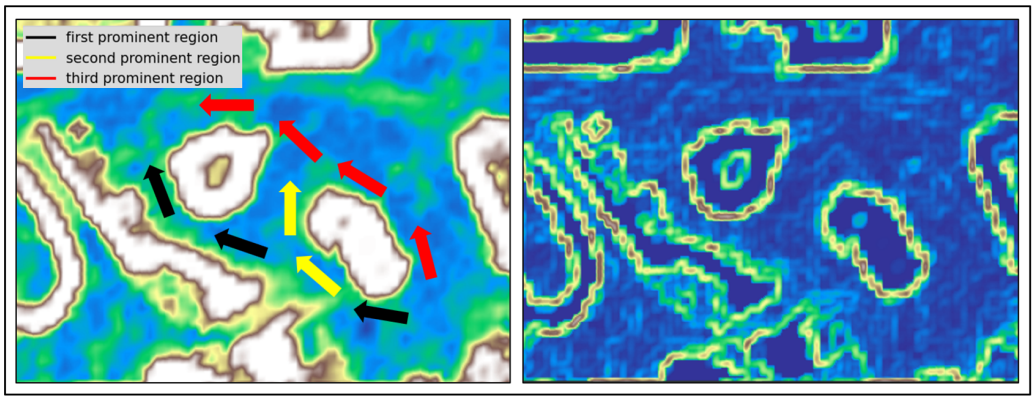


Figure 54: Rough overview of prominent alternative regions between the start and goal positions for Scenario 1. The arrows are used to represent rough intermediate directions to the goal. For example, the yellow and black arrows show that there is some overlap between the first and second regions. The map to the left is the mean of the inclination map, and the map to the right shows standard deviations. Please refer to Section 3.7 for the full maps and color scheme used to represent ranges of values in these maps. The colors range from dark blue to light blue, to green, to yellow, to brown (and to white for the mean map), respectively. The values increase in this order.

In Scenario 1, although a majority of the sampling based variants obtain lower accumulated cost compared to Default RRT* for $\lambda = 1.0$), it can be seen that Inc-RRT and Dist-RRT* obtain higher accumulated costs with respect to that of Default RRT*. Since there is no apparent trend in averages of accumulated costs between the sampling-based variants, a possible explanation for this observation is that the use of a limited, distinct set of motion primitives combined with clearance checks at every node, leads to a prioritization of safe distances from obstacles over path traversability in some scenarios. This is likely related to the constrained expansion and its effect on coverage of states in the search space, as it can be seen that NH-FMM produces lower accumulated costs on average compared to Default RRT* in this case. NH-FMM, which incorporates clearance checks as opposed to Default RRT* and FMM, consistently finds solutions for $\lambda = 1.0$), in this scenario. This shows that more optimal paths which satisfy the curvature and clearance requirements exist for Scenario 1, when $\lambda = 1.0$.

Scenario 2 is characterized by a partially constrained environment with a clear-cut path to the goal. In Section 5.2.2, it is seen that for $\lambda = 1.0$, the path chosen by FMM is close to the Default RRT* path. It is also seen that averages of accumulated costs for the sampling-based variants and NH-FMM are higher compared to that of Default RRT*. Therefore a straighter path is more traversable in this case. This indicates that unnatural swerves caused by use of a distinct motion primitive set, negatively affects traversability in scenarios characterized by a clear-cut path to the goal. Intuitively, a clear-cut path to the goal through a low-cost region is more traversable than a path with detours through the same region. This highlights a limitation of using a distinct motion primitive set to construct solution paths.

As there is no obvious trend in averages of accumulated costs for the sampling-based variants, the result that these algorithms obtain lower accumulated costs compared to Default RRT* for scenarios 3, 4 and 5, and for the randomly sampled scenarios, indicates that the collision-checking scheme (i.e., the clearance checks) is the significant contributor to more traversable paths for the sampling-based variants. Default RRT* simply finds the shortest paths and does not incorporate clearance checks. Hence, this algorithm is prone to choose paths closer to obstacles, when navigation around obstacles is required for a given planning scenario. Since Default RRT* obtains lower accumulated costs compared to the

sampling-based variants for λ-values equal to 0.8 and 0.6, this indicates that decreasing λ to 0.8 may result in an over-prioritization of higher admissible inclination values that forces FMM to choose paths that are closer to obstacles for the given inclination map. This map is characterized by higher inclinations and more variations in inclination values close to obstacles.

The results in sections 5.2.1, 5.2.2 and 5.2.4 show that when λ is decreased from 1.0 to 0.6, FMM chooses paths that are closer to the Default RRT* paths. In section 5.2.3 it is shown that when $\lambda = 0.6$, FMM is forced to choose a path through two narrow passages to a region of higher elevation in order to reach the goal. In this case, the sampling-based variants tend to avoid traversing the narrow passages, although the passages satisfy the clearance requirement (i.e., NH-FMM finds solutions paths in Scenario 3, for $\lambda = 0.6$). The finding that there is tendency among the sampling-based variants to avoid exploration of narrow passages is supported by the examples in Section 5.2.7. The results from scenarios 3, 4, 5 and the randomly sampled scenarios indicate that the tree expansions in the sampling-based variants are constrained by the motion primitive set and clearance checks, but that the clearance checks can contribute to more traversable paths when $\lambda = 1.0$, by ensuring that regions close to obstacles are avoided, given the characteristics of the inclination map. The results from the random simulations show that when λ is decreased, the absolute difference in standard deviations of accumulated costs and average costs between Default RRT* and FMM decreases (please refer to Section 5.2.6). Results from the random simulations also show that NH-FMM fails to produce solutions more frequently as λ is decreased.

The lack of an apparent trend in averages of accumulated costs and average costs between the sampling-based variants is also an indication that the proposed optimization scheme for Dist-RRT*, ObjCost-RRT* and Inc-RRT* does not lead to more traversable paths for the sampling-based motion primitive scheme. It is also seen that Dist-RRT* does not generally produce shorter paths on average, compared to the other sampling-based variants. With respect to the average costs, these values do not consistently follow the trend in accumulated costs, which appears to support the arguments made by Jaillet et al. [83] (please refer to Section 4.4). Since the motion primitive optimization scheme appears to be ineffective, the RRT variants are preferred among the sampling-based variants, because these produce solutions more efficiently (i.e., the RRT variants do not incorporate optimization steps).

6 Conclusions and Future Work

In this thesis, a sampling scheme for traversability mapping and a motion planning approach for terrain are presented. The sampling scheme takes inspiration from existing approaches, and provides a basis for constructing traversability maps. Inclination maps presented in this thesis, use a simple method to analyze point clouds to compute inclination estimates within indiviual cylinders, and this approach can be extended with other, more sophisticated approaches. For example, the incorporation of other metrics corresponding to surface roughness, the number of points in a cylinder and a variance measure of a set of angles or points, similar to the approach in [9], may enable a more accurate assessment of the terrain.

The proposed motion planning approach extends RRT and RRT* by employing a fixed set of motion primitives throughout tree-building that features a collision-checking scheme that is responsible for performing clearance-checks in minimum-enclosing circles surrounding the nodes in a tree. This approach is extended to account for an inclination-based cost field. Results indicate that the collision-checking scheme is responsible for producing more traversable paths among the sampling-based variants. Since no apparent trend is observed between the proposed algorithms, the RRT-variants are preferred as these offer less computation. This indicates that a more rigorous scheme for ensuring feasibility is required to produce more traversable paths. Using local optimization to improve on the solutions generated by algorithms employing the motion primitive scheme may have a positive effect on traversability, as such techniques can assist in reducing the unnatural swerves caused by expanding with a fixed set of primitives [8].

The conclusion that RRT-variants are preferred, highlights the decrease in efficiency that results from attempting to apply RRT* using a simple motion primitive scheme. A more rigiorous feasibility scheme based on Informed-RRT* may be a better alternative compared to using the default algorithm, as Informed-RRT* is shown to be more efficient and retains the same AO and convergence guarantees as the default RRT* algorithm by sampling a subset of states described by a prolate hyperspheroid to improve a solution [88]. As a means of potentially attaining higher efficiency for the overall scheme, the presented collision-checking approach may be exploited to analyze traversability on-demand by iteratively overlapping cylinders between nodes. This may reduce the amount of processing involved

in generating a complete grid map that is used for planning. However, global information may still provide useful information when evaluating traversability of terrain.

Results from the NH-FMM algorithm show that FMM does not guarantee feasibility for UGVs without neutral-turn and reverse motion capabilities, operating in terrain scenarios. This motivates the use of a sampling-based approach in UGV motion planning on terrain. On the other hand, when NH-FMM finds solution paths, the FMM solutions for the corresponding scenarios are preferred, because FMM finds the optimal path in terms of traversability. Therefore, a sampling-based approach may be used in combination with NH-FMM, when NH-FMM fails to establish further connections to the FMM path, as suggested in Section 4.4. In Section 5.2, Default RRT* is used as an ideal benchmark algorithm, alongside FMM, to describe the results, indicating that the incorporation of an algorithm purely optimizing on distance is useful for evaluation of a new scheme in this context. However, FMM can also be used to produce optimal distance paths. This can be achieved by setting the inverse speed function in the Eikonal update to a constant value. Hence, although FMM paths are only idealistic, the algorithm can provide useful benchmarks for comparison in the evaluation of a new scheme.

The performance-based comparison used for evaluation may be improved by adding the remaining accumulated cost and path length along a straight line to the goal from the final point of the path to the accumulated cost and path length, given the limitations of the motion primitive scheme involving the acceptance of a solution within 5 m from the goal. On the other hand, a limitation of this approach is that there may not be a clear path in a straight line from the final point on the path, and the cost along a straight line may not accurately reflect the cost that would be achieved if a path connected directly to the goal, given the intrinsic properties of the algorithms and the potential variations in cost within the radius from the goal. Furthermore, attempting to connect a primitive to the goal when it is close to a tree may not prove as successful in finding a path as reaching the goal within a given radius. To enable a fairer comparison when evaluating algorithms in terms of performance, the adoption of a more sophisticated feasibility scheme may include disallowing path end points that do not directly connect to goal points. A bi-directional approach may be useful for this purpose [89].

References

[1] N. Abcouwer, S. Daftry, T. del Sesto, O. Toupet, and M. Ono, "Machine Learning Based Path Planning for Improved Rover Navigation," *IEEE Aerospace Conference*, Mar. 2021.

[2] L. C. Santos, F. N. Santos, E. J. S. Pires, A. Valente, P. Costa, and S. Magalhaes, "Path Planning for ground robots in agriculture: a short review," *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE*, Apr 2020, doi: 10.1109/icarsc49921.2020.9096177.

[3] C. U. et al., "Autonomous Driving in Urban Environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, Aug. 2008.

[4] DARPA, "Grand challenge 2004 final report," Defense Advanced Research Projects Agency, Tech. Rep., Jul. 2004. [Online]. Available: https://www.esd.whs.mil/Portals/54/Documents/FOID/Reading%20Room/DARPA/15-F-0059_GC_2004_FINAL_RPT_7-30-2004.pdf

[5] S. Bruvoll, "Situation dependent path planning for computer generated forces," 2014, Norwegian Defence Research Establishment (FFI). [Online]. Available: https://publications.ffi.no/nb/item/asset/dspace:2439/14-01222.pdf

[6] S. M. Lavalle, *Planning Algorithms*, J. O'Kane, Ed. Cambridge University Press, 2006.

[7] S. T. et al., "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, Sep. 2006.

[8] M. Thoresen, N. H. Nielsen, K. Mathiassen, and K. Y. Pettersen, "Path Planning for UGVs Based on Traversability Hybrid A*," *IEEE Robotics and Automation Letters*, vol. 6, no. 2. Institute of Electrical and Electronics Engineers (IEEE), p. 1216–1223, Apr 2021, doi: 10.1109/lra.2021.3056028.

[9] N. Perez-Higueras, A. Jardon, A. Rodriguez, and C. Balaguer, "3d exploration and navigation with optimal-rrt planners for ground robots in indoor incidents," *Sensors*, vol. 20, no. 1, p. 220, December 2019.

[10] K. Mathiassen, M. Baksaas, S. A. Græe, E. A. Mentzoni, and N. H. Nielsen, "Making the milrem themis ugv ready for autonomous operations," in *Unmanned Systems Technology XXIII*, vol. 11758, 2021, p. 117580S.

[11] J. V. G. González, "Fast marching methods in path and motion planning: improvements and high-level applications," Ph.D. dissertation, Universidad Carlos III de Madrid, 2015.

[12] S. Luu, "Comparing the motion planning methods Hybrid A* and RRT for autonomous off-road driving of bicycle vehicles," Master's thesis, 2021.

[13] S. R. Lindemann and S. M. LaValle, "Current issues in sampling-based motion planning," in *Robotics research. The eleventh international symposium*. Springer, 2005, pp. 36–54.

[14] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics - Modelling, Planning and Control*, M. J. Grimble and M. A. Johnson, Eds. Springer, 2009.

[15] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4. Institute of Electrical and Electronics Engineers (IEEE), p. 566–580, Aug. 1996, doi: 10.1109/70.508439.

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

[17] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.

[18] A. Patel, "Introduction to A*," Apr. 2022. [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

[19] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[20] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," in *ICAPS*, vol. 5, 2005, pp. 262–271.

[21] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[22] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent unmanned ground vehicles*. Springer, 1997, pp. 203–220.

[23] A. Muhammad, M. A. Ali, and I. H. Shanono, "Path planning methods for mobile robots: A systematic and bibliometric review," *ELEKTRIKA-Journal of Electrical Engineering*, vol. 19, no. 3, pp. 14–34, 2020.

[24] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Robotics research*. Springer, 2007, pp. 239–253.

[25] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global planning on the mars exploration rovers: Software integration and surface testing," *Journal of Field Robotics*, vol. 26, no. 4, pp. 337–357, 2009.

[26] A. Valero-Gomez, J. V. Gomez, S. Garrido, and L. Moreno, "The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories," *IEEE Robotics & Automation Magazine*, vol. 20, no. 4, pp. 111–120, 2013.

[27] S. Garrido, M. Malfaz, and D. Blanco, "Application of the fast marching method for outdoor motion planning in robotics," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 106–114, 2013.

[28] Z. He, J. Wang, and C. Song, "A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures," *arXiv preprint arXiv:2108.13619*, 2021.

[29] C. Arismendi, D. Álvarez, S. Garrido, and L. Moreno, "Nonholonomic motion planning using the fast marching square method," *International Journal of Advanced Robotic Systems*, vol. 12, no. 5, p. 56, 2015.

[30] V. González, C. A. Monje, L. Moreno, and C. Balaguer, "Fast marching square method for uavs mission planning with consideration of dubins model constraints," *IFAC-PapersOnLine*, vol. 49, no. 17, pp. 164–169, 2016.

[31] P. Švestka and M. H. Overmars, "Motion planning for car-like robots using a probabilistic learning approach," *The International Journal of Robotics Research*, vol. 16, no. 2, pp. 119–143, 1997.

[32] M. B. Kobilarov and G. S. Sukhatme, "Near time-optimal constrained trajectory planning on outdoor terrain," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 1821–1828.

[33] S. M. LaValle, J. J. Kuffner, B. Donald *et al.*, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and computational robotics: new directions*, vol. 5, pp. 293–308, 2001.

[34] S. Bak, S. Bogomolov, T. A. Henzinger, and A. Kumar, "Challenges and tool implementation of hybrid rapidly-exploring random trees," in *International Workshop on Numerical Software Verification*. Springer, 2017, pp. 83–89.

[35] 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.

[36] R. Takemura and G. Ishigami, "Traversability-based RRT* for planetary rover path planning in rough terrain with LIDAR point cloud data," *Journal of Robotics and Mechatronics*, vol. 29, no. 5, pp. 838–846, 2017.

[37] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017.

[38] M. Du, J. Chen, P. Zhao, H. Liang, Y. Xin, and T. Mei, "An improved RRT-based motion planner for autonomous vehicle in cluttered environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4674–4679.

[39] M. Du, T. Mei, H. Liang, J. Chen, R. Huang, and P. Zhao, "Drivers' visual behavior-guided RRT motion planner for autonomous on-road driving," *Sensors*, vol. 16, no. 1, p. 102, 2016.

[40] D. J. Webb and J. v. d. Berg, "Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints," *arXiv preprint arXiv:1205.5088*, 2012.

[41] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *Autonomous Robots*, vol. 43, no. 7, pp. 1715–1732, 2019.

[42] N. A. Melchior, R. G. Simmons *et al.*, "Particle rrt for path planning with uncertainty." in *ICRA*. Citeseer, 2007, pp. 1617–1624.

[43] C. Jiang, Z. Hu, Z. P. Mourelatos, D. Gorsich, P. Jayakumar, Y. Fu, and M. Majcher, "R2-rrt*: reliability-based robust mission planning of off-road autonomous ground vehicle under uncertain terrain environment," *IEEE Transactions on Automation Science and Engineering*, 2021.

[44] N. A. Melchior, J.-y. Kwak, and R. Simmons, "Particle RRT for Path Planning in very rough terrain," in *NASA Science Technology Conference 2007 (NSTC 2007)*. Citeseer, 2007.

[45] S. U. Lee, R. Gonzalez, and K. Iagnemma, "Robust sampling-based motion planning for autonomous tracked vehicles in deformable high slip terrain," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2569–2574.

[46] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.

[47] E. Schmerling, L. Janson, and M. Pavone, "Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2574–2581.

[48] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[49] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[50] S. Potiris, A. Tompkins, and A. Goktogan, "Terrain-based path planning and following for an experimental mars rover," in *Australasian Conference on Robotics and Automation, Melbourne*, 2014, pp. 1–10.

[51] J. Jordan and A. Zell, "Real-time model based path planning for wheeled vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5787–5792.

[52] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," in *Proceedings, IEEE aerospace conference*, vol. 5. IEEE, 2002.

[53] A. Tahirovic and G. Magnani, "A roughness-based rrt for mobile robot navigation planning," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 5944–5949, Jan 2011.

[54] K. Iagnemma and S. Dubowsky, "Mobile robots in rough terrain: Estimation, motion planning and control with application to planetary rovers," *Springer Berlin/Heidelberg*, 2004.

[55] P. Papadakis, "Terrain traversability analysis methods for unmanned ground vehicles: A survey," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1373–1385, April 2013.

[56] Y. Ji, Y. Tanaka, Y. Tamura, M. Kimura, A. Umemura, Y. Kaneshima, H. Murakami, A. Yamashita, and H. Asama, "Adaptive motion planning based on vehicle characteristics and regulations for off-road ugvs," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 599–611, 2018.

[57] "Milrem robotics transport," https://milremrobotics.com/wp-content/uploads/2019/05/Milrem_Robotics_Transport_green-650x471.png, accessed: 2023-05-07.

[58] P. Morin and C. Samson, "Motion control of wheeled mobile robots." *Springer handbook of robotics*, vol. 1, pp. 799–826, 2008.

[59] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model," in *2006 2nd International Conference on Information & Communication Technologies*, vol. 1. IEEE, 2006, pp. 781–786.

[60] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[61] J. F. Canny, B. R. Donald, J. H. Reif, and P. G. Xavier, "On the complexity of kinodynamic planning," *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pp. 306–316, 1988.

[62] T. University, "Comp150-07: Intelligent robotics notes on configuration space," 2023, accessed: 2023-05-08. [Online]. Available: https://www.cs.tufts.edu/comp/150IR/hw/cspace.html

[63] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 33–55, 2016.

[64] J. Sethian, "Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science," no. 3, 1999.

[65] G. Peyre, "Dijkstra and Fast Marching Algorithms," Apr. 2010. [Online]. Available: https://www.numerical-tours.com/matlab/fastmarching_0_implementing/

[66] Y. Li, "Real-time motion planning of multiple agents and formations in virtual environments," Ph.D. dissertation, The Royal Institute of Technology (KTH), 2001.

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

[68] M. Stölzle, T. Miki, L. Gerdes, M. Azkarate, and M. Hutter, "Reconstructing occluded elevation information in terrain maps with self-supervised learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1697–1704, 2022.

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

[70] "Defence - milrem robotics," https://milremrobotics.com/defence/, accessed: 2023-05-07.

[71] "Høydedata og dybdedata," Norwegian Mapping Authority, 2023, accessed: May 15, 2023. [Online]. Available: https://kartverket.no/api-og-data/terrengdata

[72] "Estimating surface normals in a pointcloud," Point Cloud Library, 2023, accessed: May 1, 2023. [Online]. Available: https://pcl.readthedocs.io/en/latest/normal_estimation.html

[73] P. Fankhauser, "Grid map," ANYbotics, 2023, accessed: May 1, 2023. [Online]. Available: https://github.com/ANYbotics/grid_map

[74] E. Jelavic, D. Jud, P. Egli, and M. Hutter, "Towards autonomous robotic precision harvesting: Mapping, localization, planning and control for a legged tree harvester," *Field Robotics*, 2021.

[75] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[76] Y. Yang, J. Pan, and W. Wan, "Survey of optimal motion planning," *IET Cyber-Systems and Robotics*, vol. 1, no. 1, pp. 13–19, 2019.

[77] V. Vonásek, M. Saska, K. Košnar, and L. Přeučil, "Global motion planning for modular robots with local motion primitives," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2465–2470.

[78] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *Autonomous Robots*, vol. 43, pp. 1715–1732, 2019.

[79] L. Palmieri and K. O. Arras, "A novel rrt extend function for efficient and smooth mobile robot motion planning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 205–211.

[80] H.-M. Zhang, M.-L. Li, and L. Yang, "Safe path planning of mobile robot based on improved a* algorithm in complex terrains," *Algorithms*, vol. 11, no. 4, p. 44, 2018.

[81] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning," in *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2013, pp. 365–380.

[82] D. of Environment and Science, "Terrain slope," Queensland Government, 2023, accessed: May 12, 2023. [Online]. Available: https://wetlandinfo.des.qld.gov.au/wetlands/ecology/aquatic-ecosystems-natural/estuarine-marine/itst/terrain-slope/#:~:text=Slope%20is%20frequently%20used%20in,their%20structural%20macrobiota%5B11%5D.

[83] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, pp. 635–646, 2010.

[84] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, 2010.

[85] J. Gomez, "Fast marching method and fast marching square implementation," 2023, accessed: May 11, 2023. [Online]. Available: https://jvgomez.github.io/pages/fast-marching-method-and-fast-marching-square.html#source-code

[86] M. M. Rahman, "Motion planning algorithm: Rrt star python code," 2016, accessed: May 11, 2023. [Online]. Available: https://www.linkedin.com/pulse/motion-planning-algorithm-rrt-star-python-code-md-mahbubur-rahman

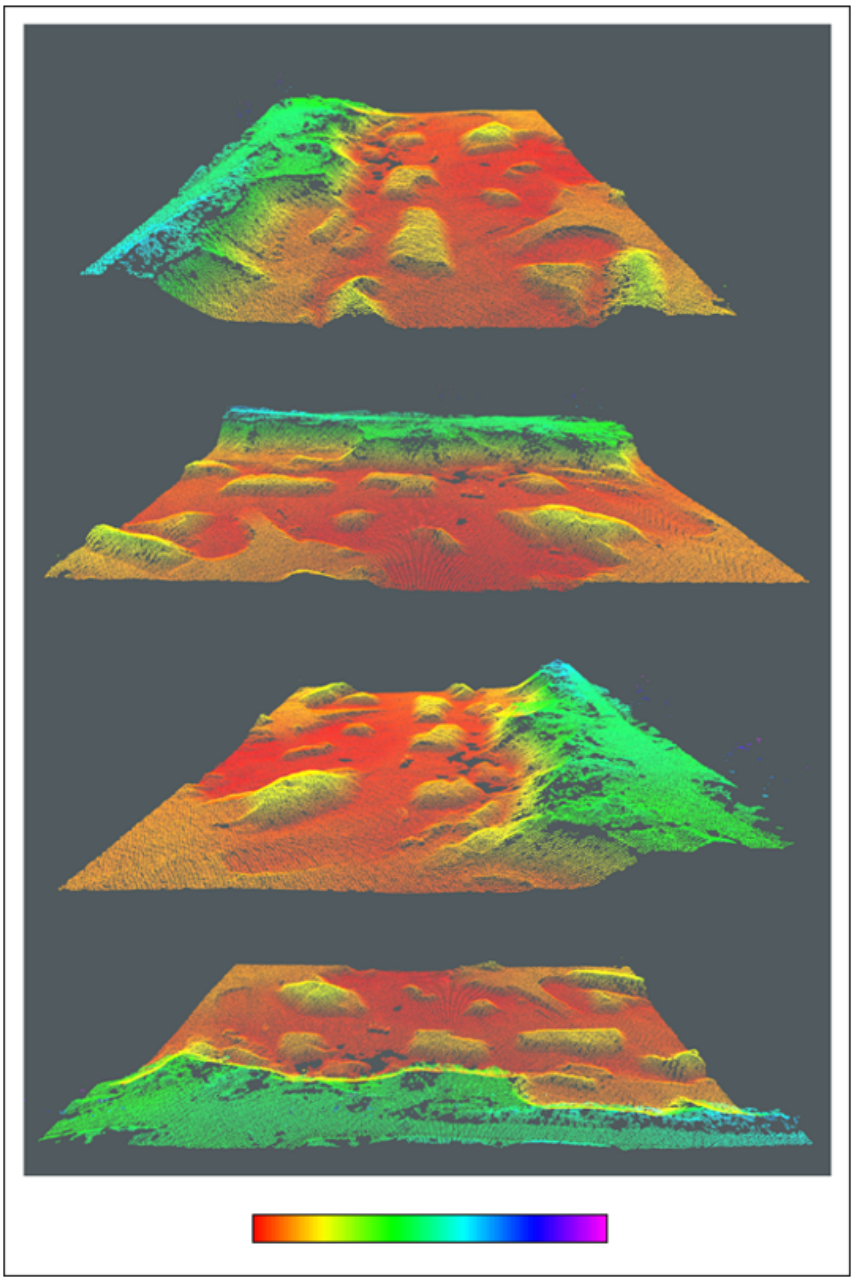
[87] M. M. Rahman, L. Bobadilla, and B. Rapp, "Sampling-based planning algorithms for multi-objective missions," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 709–714.

[88] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 2997–3004.

[89] F. Lamiraux, E. Ferré, and E. Vallée, "Kinodynamic motion planning: Connecting exploration trees using trajectory optimization methods," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4. IEEE, 2004, pp. 3987–3992.

Appendix A - Traversability Mapping

Four Different Perspectives of the Point Cloud Dataset



The color bar in the picture is a cropped HSV color palette, ranging from red to purple. The color red indicates points corresponding to a range of the lowest height values, while purple indicates points corresponding to the range of the greatest height values within the dataset. The purpose of including this picture is to give a rough high-level overview of the terrain data that was used to generate the traversability map for motion planning.

Inclination Map

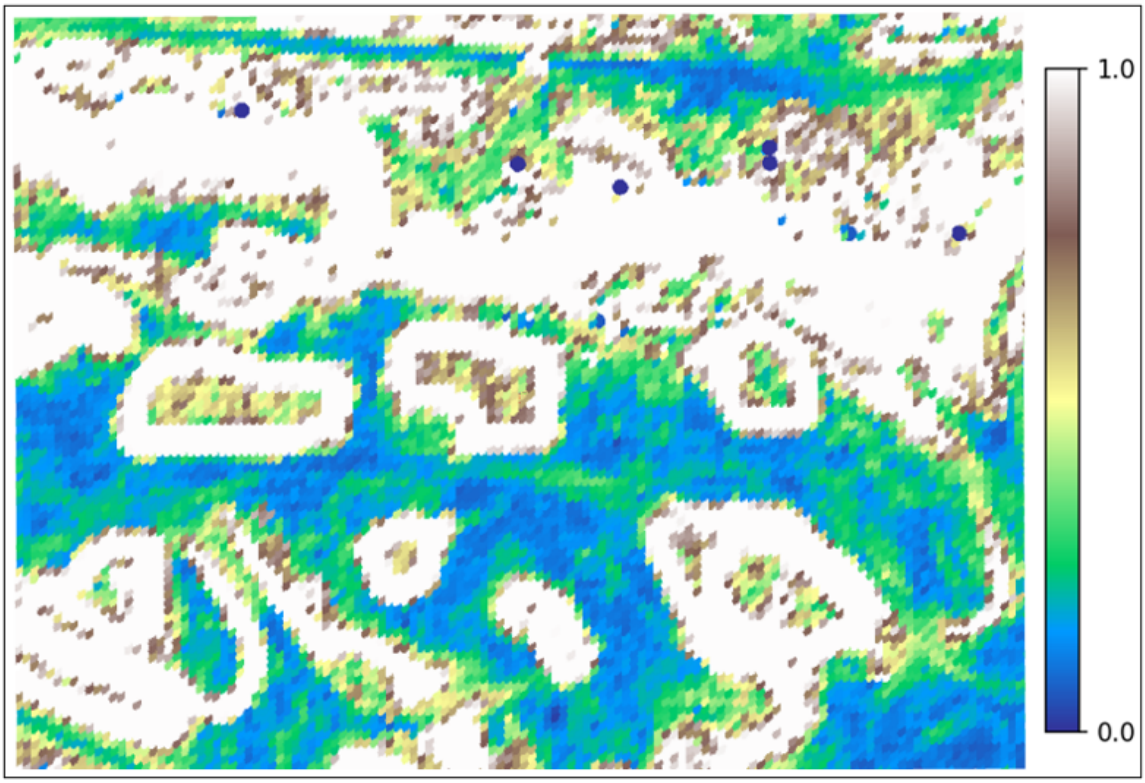


Figure 55: Triangle approach with $\alpha_{lim} = 11.8$.

Appendix B - Simulation Results

Overview of planning scenarios:

- Scenario 1 - Alternative paths to the goal, open environment.
- Scenario 2 - Obvious path to the goal, partially constrained environment.
- Scenario 3 - Longer, alternative paths to the goal, open environment.
- Scenario 4 - Longer, distinct, prominent path to partially constrained goal environment.
- Scenario 5 - Alternative paths to the goal, constrained starting environment.

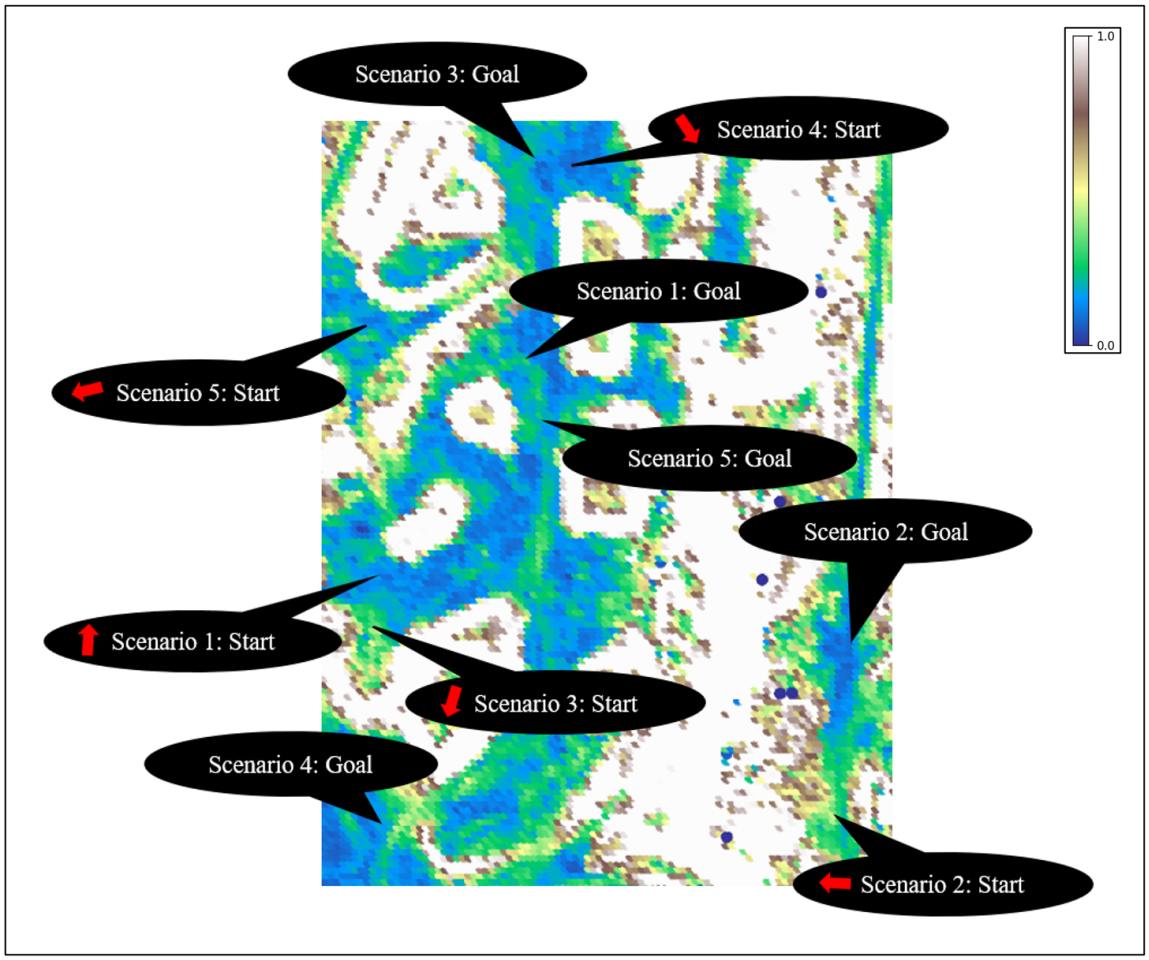


Figure 56: Rough high-level overview of chosen planning scenarios. The black arrows point to start and goal positions for the 5 chosen scenarios, and red arrows represent sampled orientations at the start positions.

Scenario 1: Path Lengths, Accumulated Costs and Average Costs

(Averages over 10 runs per table)

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	87.6	1.2	0.014
NH-FMM	90.0	1.3	0.014
Default RRT*	72.9	3.2	0.043
MP-RRT	100.7	2.4	0.024
ObjCost-RRT	101.5	2.9	0.029
Inc-RRT	111.8	3.9	0.036
Dist-RRT*	104.1	3.2	0.030
ObjCost-RRT*	106.7	2.7	0.025
Inc-RRT*	101.5	2.7	0.027

Table 1

$\lambda = 0.8$	Length (m)	Acc. cost	Avg. cost
FMM	72.6	13.1	0.181
NH-FMM	77.1	14.0	0.181
Default RRT*	72.4	13.8	0.190
MP-RRT	102.0	19.2	0.188
ObjCost-RRT	123.2	24.0	0.194
Inc-RRT	102.0	19.3	0.189
Dist-RRT*	100.5	19.0	0.189
ObjCost-RRT*	102.4	19.3	0.188
Inc-RRT*	99.4	18.9	0.190

Table 2

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	75.6	22.0	0.291
NH-FMM	-	-	-
Default RRT*	72.5	24.2	0.334
MP-RRT	105.0	36.6	0.349
ObjCost-RRT	107.7	37.3	0.345
Inc-RRT	100.7	35.3	0.351
Dist-RRT*	105.4	36.9	0.350
ObjCost-RRT*	99.0	34.5	0.348
Inc-RRT*	100.5	35.3	0.352

Table 3

Scenario 2: Path Lengths, Accumulated Costs and Average Costs

(Averages over 10 runs per table)

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	48.2	1.1	0.022
NH-FMM	53.6	2.2	0.040
Default RRT*	46.6	2.0	0.043
MP-RRT	59.8	9.3	0.155
ObjCost-RRT	63.0	10.8	0.168
Inc-RRT	61.0	9.2	0.150
Dist-RRT*	60.0	10.4	0.170
ObjCost-RRT*	61.3	11.0	0.179
Inc-RRT*	60.8	11.1	0.183

Table 4

$\lambda = 0.8$	Length (m)	Acc. cost	Avg. cost
FMM	46.6	8.2	0.175
NH-FMM	-	-	-
Default RRT*	46.5	8.3	0.179
MP-RRT	61.0	14.7	0.240
ObjCost-RRT	70.0	16.4	0.233
Inc-RRT	58.3	15.2	0.261
Dist-RRT*	60.8	14.2	0.233
ObjCost-RRT*	62.5	15.1	0.241
Inc-RRT*	63.4	15.6	0.245

Table 5

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	47.2	13.7	0.290
NH-FMM	-	-	-
Default RRT*	46.5	14.9	0.321
MP-RRT	59.5	19.7	0.331
ObjCost-RRT	64.3	20.4	0.317
Inc-RRT	60.4	19.6	0.325
Dist-RRT*	61.0	19.8	0.325
ObjCost-RRT*	58.9	19.0	0.323
Inc-RRT*	58.3	18.9	0.325

Table 6

Scenario 3: Path Lengths, Accumulated Costs and Average Costs

(Averages over 10 runs per table)

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	141.7	2.1	0.015
NH-FMM	154.2	2.8	0.018
Default RRT*	128.5	11.9	0.093
MP-RRT	175.4	7.1	0.040
ObjCost-RRT	166.4	5.4	0.033
Inc-RRT	172.9	6.4	0.037
Dist-RRT*	174.1	7.0	0.041
ObjCost-RRT*	175.4	6.9	0.040
Inc-RRT*	175.6	6.9	0.039

Table 7

$\lambda = 0.8$	Length (m)	Acc. cost	Avg. cost
FMM	131.1	23.8	0.182
NH-FMM	-	-	-
Default RRT*	128.6	27.9	0.217
MP-RRT	173.1	32.8	0.190
ObjCost-RRT	183.4	35.2	0.192
Inc-RRT	177.6	34.4	0.193
Dist-RRT*	173.9	33.3	0.191
ObjCost-RRT*	172.4	33.3	0.193
Inc-RRT*	183.1	35.5	0.194

Table 8

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	135.9	39.2	0.288
NH-FMM	156.4	47.0	0.300
Default RRT*	128.6	43.3	0.336
MP-RRT	172.9	59.0	0.341
ObjCost-RRT	178.9	60.9	0.341
Inc-RRT	167.7	57.1	0.341
Dist-RRT*	174.6	60.0	0.344
ObjCost-RRT*	177.4	61.3	0.345
Inc-RRT*	179.7	61.6	0.343

Table 9

Scenario 4: Path Lengths, Accumulated Costs and Average Costs

(Averages over 10 runs per table)

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	219.1	4.4	0.020
NH-FMM	-	-	-
Default RRT*	194.2	11.6	0.060
MP-RRT	246.3	11.4	0.046
ObjCost-RRT	231.3	9.0	0.039
Inc-RRT	247.6	10.8	0.043
Dist-RRT*	241.4	10.7	0.044
ObjCost-RRT*	245.9	10.5	0.043
Inc-RRT*	249.8	10.2	0.041

Table 10

$\lambda = 0.8$	Length (m)	Acc. cost	Avg. cost
FMM	194.7	35.7	0.183
NH-FMM	-	-	-
Default RRT*	194.0	38.1	0.197
MP-RRT	237.5	45.3	0.191
ObjCost-RRT	249.8	47.6	0.191
Inc-RRT	249.3	47.1	0.189
Dist-RRT*	245.3	46.8	0.191
ObjCost-RRT*	255.1	48.8	0.191
Inc-RRT*	268.2	51.9	0.193

Table 11

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	196.6	60.1	0.306
NH-FMM	-	-	-
Default RRT*	193.8	64.7	0.334
MP-RRT	259.6	88.1	0.339
ObjCost-RRT	242.5	82.8	0.341
Inc-RRT	239.5	81.9	0.342
Dist-RRT*	237.8	81.0	0.341
ObjCost-RRT*	242.3	83.0	0.343
Inc-RRT*	241.2	82.5	0.342

Table 12

Scenario 5: Path Lengths, Accumulated Costs and Average Costs

(Averages over 10 runs per table)

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost
FMM	94.1	3.7	0.039
NH-FMM	-	-	-
Default RRT*	72.8	10.6	0.146
MP-RRT	105.2	6.8	0.066
ObjCost-RRT	103.5	7.3	0.070
Inc-RRT	106.7	7.0	0.068
Dist-RRT*	106.9	7.3	0.069
ObjCost-RRT*	102.8	8.0	0.078
Inc-RRT*	117.4	8.3	0.074

Table 13

$\lambda = 0.8$	Length (m)	Acc. cost	Avg. cost
FMM	75.5	14.5	0.192
NH-FMM	-	-	-
Default RRT*	73.1	17.4	0.238
MP-RRT	109.9	21.5	0.197
ObjCost-RRT	106.5	20.7	0.195
Inc-RRT	111.4	21.9	0.196
Dist-RRT*	103.7	20.6	0.199
ObjCost-RRT*	104.3	20.5	0.197
Inc-RRT*	94.0	18.8	0.199

Table 14

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost
FMM	73.6	22.5	0.306
NH-FMM	-	-	-
Default RRT*	72.8	24.0	0.330
MP-RRT	96.6	31.2	0.322
ObjCost-RRT	113.7	37.0	0.323
Inc-RRT	96.8	31.2	0.322
Dist-RRT*	105.6	34.3	0.324
ObjCost-RRT*	110.1	35.7	0.323
Inc-RRT*	99.6	32.0	0.321

Table 15

Results from 10 Random Simulations: Path Lengths, Accumulated Costs, Average Costs and Standard Deviations

(Averages over 10 sampled scenarios)

$\lambda = 1.0$	Length (m)	Acc. cost	Avg. cost	Acc. cost (σ_s)	Avg. cost (σ_s)
FMM	74.4	1.4	0.019	0.8	0.002
NH-FMM	87.0	2.0	0.023	1.0	0.004
Default RRT*	65.3	5.8	0.077	4.6	0.043
MP-RRT	97.7	4.3	0.044	2.7	0.018
ObjCost-RRT	97.7	3.3	0.033	2.0	0.006
Inc-RRT	95.7	3.6	0.038	2.0	0.019
Dist-RRT*	95.7	3.1	0.032	1.9	0.008
ObjCost-RRT*	93.0	3.4	0.035	2.5	0.012
Inc-RRT*	100.2	3.5	0.035	1.9	0.009

Table 16

$\lambda = 0.8$	Length (m)	Acc. cost	Avg. cost	Acc. cost (σ_s)	Avg. cost (σ_s)
FMM	65.5	12.0	0.182	5.7	0.005
NH-FMM	6 out of 10 passed				
Default RRT*	65.2	14.1	0.208	7.5	0.026
MP-RRT	92.3	17.4	0.185	9.1	0.010
ObjCost-RRT	103.5	19.6	0.189	8.3	0.009
Inc-RRT	89.1	16.9	0.189	7.9	0.012
Dist-RRT*	97.9	18.2	0.184	10.8	0.005
ObjCost-RRT*	96.0	18.0	0.185	9.6	0.006
Inc-RRT*	91.5	17.3	0.187	8.6	0.008

Table 17

$\lambda = 0.6$	Length (m)	Acc. cost	Avg. cost	Acc. cost (σ_s)	Avg. cost (σ_s)
FMM	65.6	20.7	0.315	9.4	0.007
NH-FMM	2 out of 10 passed				
Default RRT*	65.0	21.9	0.338	9.9	0.004
MP-RRT	103.7	35.6	0.337	19.0	0.021
ObjCost-RRT	94.0	32.0	0.337	16.0	0.010
Inc-RRT	95.7	32.3	0.333	17.9	0.015
Dist-RRT*	99.6	33.9	0.337	18.6	0.011
ObjCost-RRT*	90.8	30.9	0.337	15.0	0.011
Inc-RRT*	103.5	35.0	0.336	19.1	0.012

Table 18